Algoritmy ořezávání

(Clipping Algorithms)

Habilitační práce

(Habilitation Thesis)

Prof.Ing.Václav Skala, CSc.

http://www.VaclavSkala.eu

Abstrakt

Algoritmy ořezávání a jejich implementace je jednou z klíčových operací v počítačové grafice a je nedílnou součástí systému zpracování geometrických entit. V této práci jsou uvedeny základní metody pro ořezávání přímek, úseček, mnohoúhelníků, oblastí a trojúhelníků v E^2 E^3.

Abstract

Clipping algorithms and their implementation is one of the key issues in processing of geometric entities. In this work fundamental algorithms for line, line segment, triangle, areaand polygon clipping in E^2 and E^3 are presented.

Algoritmy ořezávání – Habilitační práce
Clipping algorithms – Habilitation Thesis
Published & printed by: Vaclav Skala – UNION Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic
Year 2011
ISBN 978-80-86943-22-0

Electronic vision 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 kopie habilitační práce obhájené v roce 1990, tedy více než před 20 lety. 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 byla habilitace napsána

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

Odborné publikace autora (13 zahraničních publikací, 3 národní s mezinárodní účastí) od r.1985 do r.1989, které se vztahují k habilitační práci, jsou uvedeny v kap.5 a svědčí o odborné úrovni habilitačního tématu. Po podání habilitační práce pak vznikly další odborné publikace věnované metodám ořezávání („Line clipping“) a zájemce je 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

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. F.R.A.Hopgood
Rutherford Appleton
Laboratory, U.K.
a
Oxford Brookes
University, U.K.

Prof. Jack E. Bresenham
IBM Corp., USA
a
Winthrop University,
USA

Prof. David F.Rogers
U.S. Naval Academy,
USA

V Plzni dne: 2011-07-22 prof.Ing.Václav Skala, CSc.

Vysoká škola strojní a elektrotechnická v Plzni

Katedra informatiky a výpočetní techniky

H A B I L I T A Č N Í P R Á C E

V Plzni dne 26. září 1990

Doc. Ing. Václav Skala, CSc.

Předmluva

Přesto, že jsem byl v roce 1989 docentem jmenován v rámci řádného habilitačního řízení, okolnosti (vyhláška ministerstva školství, mládeže a tělovýchovy ČR o podmínkách a průběhu jmenování profesorů a habilitací docentů) mne nutí k tomu, že předkládám tuto habilitační práci.

Předkládaná habilitační práce je upravenou částí ze skript **Počítačová grafika I a II**, s těžištěm v řešení operací ořezávání. Vedle známých algoritmů obsahuje též algoritmy původní, které byly publikovány v zahraničním odborném tisku, viz seznam publikovaných prací.

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, např. řady SM 52/12.

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 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 zatím převážně používají a jsou založena na kreslení čar. Rozvoj počítačové grafiky v ČSFR lze dokumentovat na vzrůstajícím počtu systémů vhodných pro počítačovou grafiku. Patří k nim 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, a systémy IBM PC/XT/AT, resp. PS/2. Tyto systémy jsou v mnoha případech doplňovány různými grafickými zobrazovacími jednotkami s velkou rozlišovací schopností a akcelerátory. V krátké budoucnosti lze očekávat používání grafických pracovních stanic, které se vyznačují velkým pracovním výkonem, zejména pak počtem prováděných grafických operací. Patří k nim zejména pracovní stanice typů IRIS, SUN, APOLLO a VRX firmy Hewlett Packard. Se systémy je dodáváno nejen základní programové vybavení pro počítačovou grafiku (např. standardy GKS, PHIGS), ale i ucelené systémy orientované uživatelsky. Jsou to např. Eagle, AutoCAD, OrCAD, MOVIE.BYU atd. Vzhledem k tomu, že již byla definována norma ISO 7942 (GKS - Graphical Kernel System), ISO 8805 (GKS-3D) a ISO 9592 (PHIGS - Programmers Hierarchical Interactive Graphics System), je možné, aby tvůrci nových celků

již používali programové rozhraní GKS, resp. PHIGS pro nové aplikace, i když se poněkud sníží rychlost odezvy systému a zvýší se paměťové nároky. Při použití standardního rozhraní se docílí zejména u složitějších celků snazší přenositelnost na jiné výpočetní systémy, což je důležitým aspektem při jejich návrhu. I když v ČSFR není dosud úplná implementace GKS, resp. PHIGS, lze ji v krátké době očekávat.

1.1. Vymezení obsahu pojmu počítačová grafika

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 především ve výzkumu nových technologiích a v mnoha netradičních oblastech.

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.

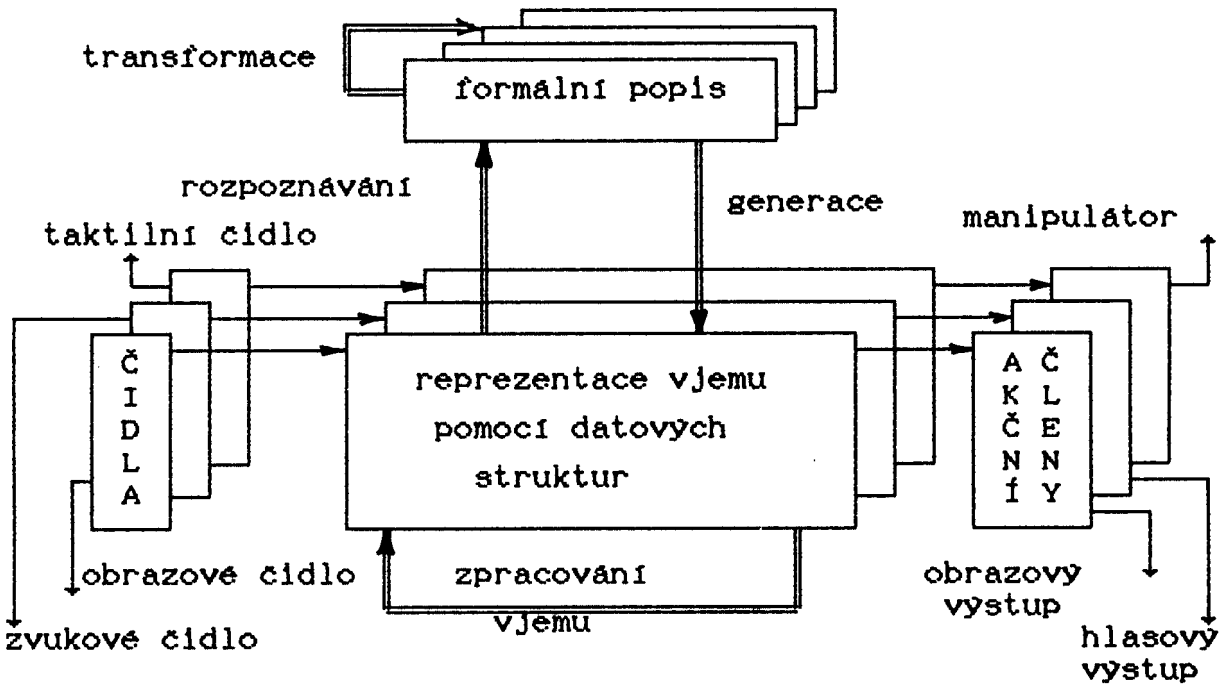
Některé typy úloh, zejména simulační, vyžadují superpočítače, kdy ani použití systému CRAY není postačující, např. z důvodů rychlosti odezvy systému. Pro efektivní použití a tvorbu nových grafických systémů je však nezbytné pochopit jednotlivé základní principy metod a algoritmů, i když bude neustále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

Vymezení oblasti počítačové grafiky je charakterizováno tabulkou 1.1, 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. Obr.1.1 pak naznačuje funkční začlenění počítačové grafiky. 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.

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



Obr. 1.1

Za obrazové čidlo lze považovat např. kameru, 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**. 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.

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 Marconi Radar Systems Ltd., SINGER, Link Miles (U.K.). 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 CT5 umožňuje reprezentovat 10 000 čtverečních mil při hustotě až 1.5 mil n-úhelníků na čtvereční

mili).

Je zrejmé, že techniky dálkového průzkumu Země umožňují i monitorování určité oblasti. Typickou ukázkou jsou snímeky oblasti Černobylu před havárií jaderné elektrárny. Takové monitorovací techniky lze použít v mnoha oblastech, zejména pak ve při použití např. kamer pracujících v infračervené části spektra. Jedno z nejznámějších center dálkového průzkumu Země je RAE Farnborough (U.K.). Jako téměř každodenní aplikaci počítačové grafiky lze vidět na obrazovce TV přijímače při předpovědi počasí.

Další aplikací počítačové grafiky lze nalézt v oblasti architektury, stavebnictví apod., kdy např. výstup z Polaroid kamery je 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 čisté urbanistické studie, kdy se např. budova "zasadí" do skutečné existující krajiny.

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.

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 musí být z technického hlediska velmi výkonné, ale musí 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, raketovém, leteckém a loďařském

průmyslu.

Další oblastí aplikace počítačové grafiky je oblast systémů usnadňující publikační činnost. Jednodušší systémy se nazývají textovými procesory, resp. WP systémy (Word Processor Systems), mezi které lze zařadit Word Perfect nebo Chi-Writer. 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 publikací rozsáhlých, 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 s obrázky.

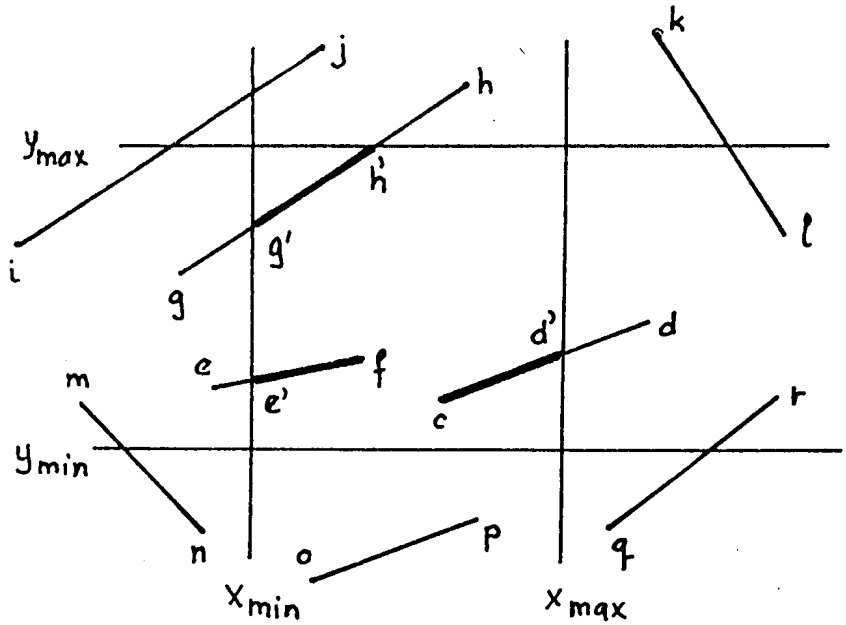
Jistě neposlední oblastí aplikace počítačové grafiky jsou nadstavby pro operační systémy, či jiné programové celky, které umožňují uživateli "příjemnou" komunikaci se systémem. Tyto systémy umožňují, aby i běžný uživatel výpočetní techniky mohl tuto techniku využívat velmi efektivně ve své profesi. Jako příklad uveďme alespoň prostředí GEM, známé především od počítačů řady ATARI, resp. OS-Shell na IBM-PC.

2. Orezávání

Velmi důležitou částí každého programového vybavení pro počítačovou grafiku je orezávání těch částí scény, které jsou mimo zobrazovací plochu. Nejjednodušším případem je orezávání v dvourozměrném prostoru vůči obdélníku zobrazovací plochy. Tento případ je v praxi nejčastější, zejména při zobrazování plošných objektů.

2.1. Dvourozměrné orezávání

Na obr.2.1.1 jsou uvedeny některé situace, které mohou nastat v případě, kdy je zobrazovací plocha ohraničena obdélníkem. Úkolem orezávání je odříznout ty části úseček, které jsou mimo obdélník, tj. zobrazit jen ty části, které jsou uvnitř obdélníka $< x_{min} , x_{max} > x < y_{min} , y_{max} >$.



Obr.2.1.1

Vzhledem k tomu, že operace orezávání se používá velmi často, je nutné, aby příslušný algoritmus velmi rychle rozhodl, zda je část úsečky uvnitř obdélníka a určil příslušné koncové body.

2.2. Cohen - Sutherlandův algoritmus

Algoritmus těží především z jednoduchého způsobu zakódování jednotlivých možností tak, aby jednotlivé případy mohly být snadno rozlišeny. Algoritmus začíná vždy určením čtyřbitového kódu pro každý koncový bod zkoumané úsečky, a to tak, že stavové slovo příslušející danému bodu nabývá hodnoty podle obr.2.2.1, kde hodnota 1 označuje hodnotu TRUE.

1 0 0 1	1 0 0 0	1 0 1 0
0 0 0 1	0 0 0 0	0 0 1 0
0 1 0 1	0 1 0 0	0 1 1 0

Obr.2.2.1

Podle polohy bodu vůči obdélníku ořezávání nabývají jednotlivé bity stavového slova následujících hodnot (bity číslovány zprava doleva):

bit 0 bod je vlevo od obdélníka
bit 1 bod je vpravo od obdélníka
bit 2 bod je pod obdélníkem
bit 3 bod je nad obdélníkem

V případě, že daný programovací jazyk nemá možnost nastavení i-tého bitu, lze nastavení jednotlivých bitů provést pomocí operace sčítání, a to:

```
if x < x_min then S := 1
             else if x > x_max then S := 2
                                else S := 0;
if y < y_min then S := S + 4
             else if y > y_max then S := S + 8;
```

Pro danou úsečku s koncovými body x_1 a x_2 označme S_1 stavové slovo příslušející bodu x_1 a S_2 stavové slovo příslušející bodu

x_2. Je zřejmé, že pokud $S_1 = 0$ a zároveň $S_2 = 0$, je daná úsečka zcela uvnitř daného obdélníka. Mají-li stavová slova S_1 a S_2 takovou hodnotu, že některý bit je v obou případech různý od nuly, pak daná úsečka nebude procházet daným obdélníkem, neboť bude celá nad obdélníkem (nebo pod ním, vlevo nebo vpravo od něho). Uvedený způsob kódování umožňuje rychlé určení takových případů, kdy je úsečka zcela vně, a bude tedy vypuštěna, anebo je zcela uvnitř a není nutné počítat průsečíky s hranicí okna.

V případě, že nenastává ani jedna z výše uvedených situací, je nutné najít průsečík s příslušnou hranou obdélníka, přičemž souřadnice průsečíku se určí takto

$(x_{min} , k . (x_{min} - x_1) + y_1)$, je-li bod vlevo, $k \neq \infty$

$(x_{max} , k . (x_1 - x_{max}) + y_1)$, je-li bod vpravo, $k \neq \infty$

$(q . (y_1 - y_{max}) + x_1 , y_{max})$, je-li bod nad, $q \neq \infty$

$(q . (y_{min} - y_1) + x_1 , y_{min})$, je-li bod pod, $q \neq \infty$

kde

$$k = \frac{y_2 - y_1}{x_2 - x_1} \qquad \text{pro} \qquad x_2 \neq x_1$$

resp.

$$q = \frac{x_2 - x_1}{y_2 - y_1} \qquad \text{pro} \qquad y_2 \neq y_1$$

Celý Cohen-Sutherlandův algoritmus může být realizován např. algoritmem 2.2.1. Pro jednoduchost algoritmu jsou zavedeny operátory **land** a **lor** pracující s jednotlivými bity jejich celočíselných operandů jako s logickými proměnnými, přičemž výsledek je ukládán opět do odpovídajících bitů celočíselného výsledku. V Turbo Pascalu lze snadno použít přímo operátory **and** a **or**.

```
var xmin, xmax, ymin, ymax: real;
{ globální proměnné určující velikost výřezu }
procedure CLIP ( x1 , y1 , x2 , y2: real);
label 99;
var x, y: real;
    c, c1, c2: integer;
procedure CODE ( x, y: real; var c: integer);
begin c:=0;
      if x < xmin then c:=1
                  else if x > xmax then c:=2;
      if y < ymin then c:=c+4
                   else if y > ymax then c:=c+8
end { CODE };
begin
   CODE(x1,y1,c1);
   CODE(x2,y2,c2);
   { zjištění stavového slova }
   if ( c1 land c2 ) = 0 then
   begin { úsečka může procházet obdélníkem }
      if ( c1 lor c2 ) <> 0 then
      { úsečka neleží celá v obdélníku }
      repeat
         if c1 = 0 then c := c2 else c := c1;
         { operátory land, lor provádějí operace and, or }
         { nad všemi bity typu integer; výsledek je opět }
         { typu integer - v T-PASCALU lze použít and, or }
         if (c land 1) <> 0 then { bod je vlevo }
         begin y := y1+(xmin-x1)*(y2-y1)/(x2-x1);
               x:=xmin
         end
         else if (c land 2) <> 0 then { bod je vpravo }
              begin y:=y1+(xmax-x1)*(y2-y1)/(x2-x1);
                    x:=xmax
              end
              else if (c land 4) <> 0 then { bod je pod }
              begin y:=ymin;
                    x:=x1+(ymin-y1)*(x2-x1)/(y2-y1)
              end
```

```
            else if (c land 8) <> 0 then { bod je nad }
                 begin y:=ymax;
                       x:=x1+(ymax-y1)*(x2-x1)/(y2-y1)
                 end;
        if c = c1 then
            begin x1:=x; y1:=y; CODE(x1,y1,c1) end
        else
            begin x2:=x; y2:=y; CODE(x2,y2,c2) end;
        if (c1 land c2) <> 0 then go to 99
      until (c1 lor c2) = 0;
      LINE (x1,y1,x2,y2)
   end;
99:
end { konec procedury CLIP };
```

Algoritmus 2.2.1

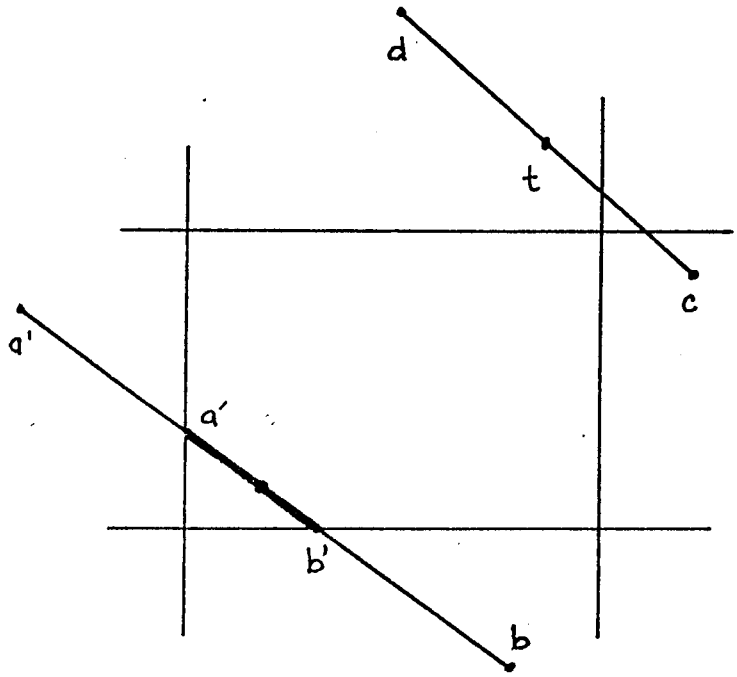
Nevýhodou Cohen-Sutherlandova algoritmu je nutnost používání reprezentace čísel s pohyblivou řádovou čárkou včetně operací násobení a dělení, viz např. [44].

Vzhledem k tomu, že se ořezávání velmi často používá v posloupnosti operací až těsně před vlastním zobrazením na daném zařízení, kde jsou souřadnice určené v celočíselné reprezentaci, lze pro tento případ specifikovat speciální algoritmus.

2.3. Ořezávání půlením

Algoritmus ořezávání půlením je založen na principu půlení intervalu, který se často používá např. při řešení nelineárních rovnic. Pro zrychlení celého procesu se musí opět co nejrychleji zjistit, zda kreslená úsečka je zcela uvnitř či zcela vně daného obdélníka. K tomu se využije analogický postup jako u Cohen-Sutherlandova algoritmu. Ořezávání půlením je důležité zejména z toho důvodu, že nevyžaduje násobení a dělení, neboť dělení dvěma lze realizovat pomocí operace posuvu vpravo. Při programové realizaci je tento algoritmus pomalejší než předchozí, avšak při hardwarové realizaci je nepoměrně rychlejší. Je nutné poznamenat, že lze použít reprezentace s

pevnou desetinnou tečkou nebo reprezentace s pohyblivou řádovou čárkou.



Obr. 2.3.1

Počet binárních míst m reprezentující desetinnou část musí být alespoň tak velký, že:

$$2^m > \max (nx, ny),$$

kde nx, resp. ny je rozlišovací schopnost ve směru osy x, resp. osy y.

Z důvodu názornosti algoritmu je zaveden fiktivní datový typ **fixed** m, který reprezentuje hodnoty v pevné řádové čárce o m desetinných místech. Při skutečné realizaci se pak použije reprezentace celočíselná s hodnotami vynásobenými konstantou 2^m, tj. posunutí o m míst vlevo.

```
{ xmin, xmax, ymin, ymax - konstanty }
{ určující velikost výřezu }
procedure MIDCLIP ( xa , ya , xb , yb: integer );
{ xa, ya , xb, yb koncové body úsečky }
label 9;
const m = 10;
var x, y, x1, y1, x2, y2, c, c1, c2: fixed m;
{ definice proměnných s pevnou řádovou čárkou }
```

```
procedure CODE ( x, y: fixed m; var c: integer);
begin
   c:=0;
   if x < xmin then c:=1
               else if x > xmax then c:=2;
   if y < ymin then c:=c+4
               else if y > ymax then c:=c+8
end { CODE };

procedure INTER ( x1 , y1 , x2 , y2: fixed m; { krajní body }
             var x , y: fixed m { průsečík });
{ výpočet průsečíku s obdélníkem bod (x1,y1) }
{ je uvnitř, bod (x2,y2) je vně }
var c: integer;
begin
   while ( abs (x2-x1) > 2^-m ) or ( abs (y2-y1) > 2^-m ) do
   begin
      x := (x1+x2)/2; { realizovat pomocí shift right }
      y := (y1+y2)/2; { nalezení střední hodnoty }
      CODE (x,y,c);
      if c = 0 then begin x1 :=x ;  y1 := y end
               else begin x2 := x;  y2 := y end
   end;
   x := x1;     y := y1
end { INTER };
begin
   x1 := xa; x2 := xb;
   y1 := ya; y2 := yb;{ desetinná část reprezentována m bity }
   CODE (x1,y1,c1); CODE (x2,y2,c2);
   while not (( c1 land c2 ) <> 0) do
   begin
      if ( c1 lor c2 )=0 then { celá úsečka je uvnitř }
      begin
         LINE (xa,ya,xb,yb);
         go to 9
      end;
      { nalezení průsečíku s obdélníkem }
      if c1 = 0 then { bod (x1,y1) je uvnitř }
```

```
        begin
            INTER (x1,y1,x2,y2,x,y);
            LINE (xa,ya,x,y);
            go to 9
        end;
        if c2 = 0 then { bod (x2,y2) je uvnitr }
        begin
            INTER (x2,y2,x1,y1,x,y);
            LINE (x,y,xb,yb);
            go to 9
        end;
        x:=(x1+x2)/2; y:=(y1+y2)/2;
        CODE (x,y,c);
        if c = 0 then { bod (x,y) uvnitr }
        begin
            INTER (x,y,x1,y1,x1,y1);
            INTER (x,y,x2,y2,x2,y2);
            LINE (x1,y1,x2,y2);
            go to 9
        end;
        { bod x neni uvnitr }
        if ( c1 land c ) <> 0 then  { (x1,y1) , (x,y) neni videt }
            begin x1 := x;  y1 := y;  c1 := c end
        else { cast (x,y) , (x2,y2) neni videt }
            begin x2 := x;  y2 := y;  c2 := c end ;
        if ( abs (x2-x1) <= 2^-m ) and ( abs (y2-y1) <= 2^-m ) then
                    go to 9
        { odstraneni cyklu pro usecku (4,6),(6,4) }
        { je-li obdelnik <5,5> x <10,10>
    end { while };
9:  end { MIDCLIP } ;
```

Algoritmus 2.3.1

2.4. Algoritmus Liang-Barského

Ke Cohen-Sutherlandovu algoritmu navrhli Liang a Barski [85], [86] alternativní algoritmus, který vychází z parametrického vyjádření ořezávané úsečky, a to:

$$x(t) = x_r + (x_s - x_r).t \qquad t \in \langle 0, 1 \rangle$$

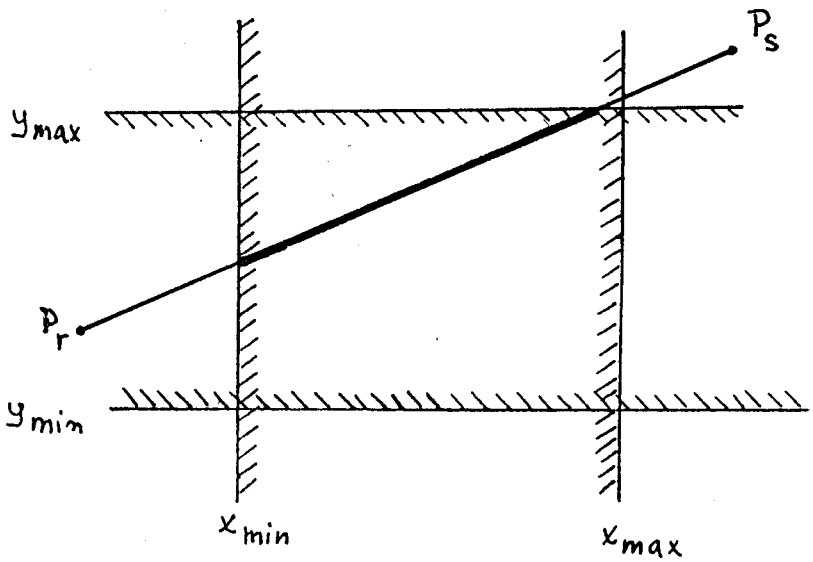
Algoritmus předpokládá, že daná úsečka bude postupně ořezávána vzhledem k jednotlivým polorovinám, tj. uvažují se postupně jen ty části úsečky, které leží ve vyšrafované polorovině.
Rovnici lze přepsat do tvaru

$$x(t) - x_r = (x_s - x_r) \cdot t$$

přičemž musí platit, že

$$x_{min} \leq x(t) \leq x_{max}$$

pro hledaný interval $t \in \langle t_1, t_2 \rangle$.



Obr.2.4.1

Uvedenou vektorovou rovnici lze přepsat do tvaru

$$p_k \cdot t \leq q_k \qquad k = 1,2,3,4$$

kde:

$$p_1 = -\Delta x \qquad q_1 = x_r - x_{min}$$

$$p_2 = \Delta x \qquad q_2 = x_{max} - x_r$$

$$p_3 = -\Delta y \qquad q_3 = y_r - y_{min}$$

$$p_4 = \Delta y \qquad q_4 = y_{max} - y_r$$

Nyní je zřejmé, že pro svislé či vodorovné úsečky je pro některá k hodnota $p_k = 0$. Je-li navíc i hodnota $q_k < 0$ pro toto k, pak úsečku lze odmítnout, neboť leží vně uvažované poloroviny. Celý postup je realizován algoritmem 2.4.1.

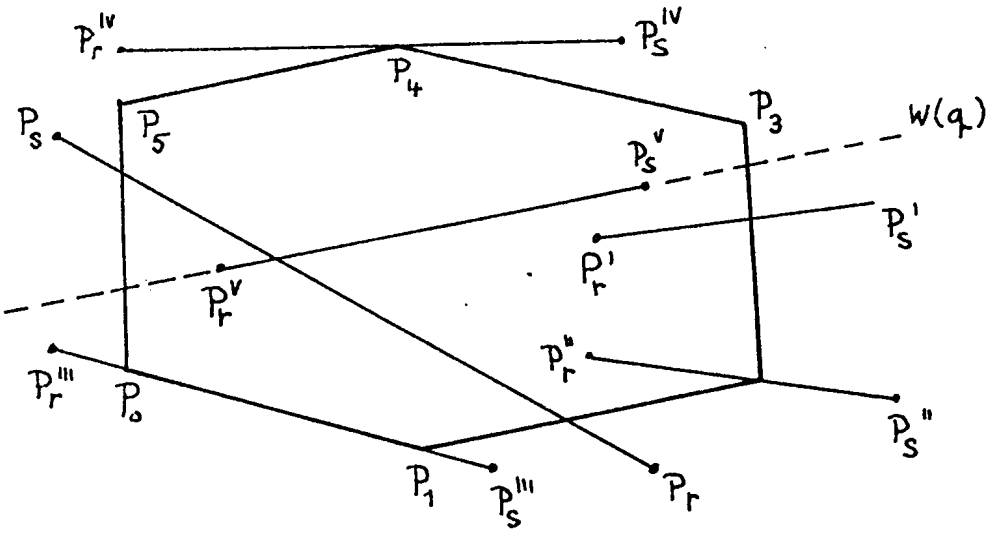
```
var xmin, xmax, ymin, ymax: real;
procedure LIANG-BARSKY ( var xr, yr, xs, ys: real );
var t1, t2, dx, dy: real;
function TEST ( p, q: real; var t1, t2: real ): boolean;
var r: real;
begin
   TEST: = true;
   if p < 0 then
   begin r := q / p;
         if r > t2 then TEST := false
                   else if r > t1 then t1 := r
   end
   else
   if p > 0 then
   begin
      r := q / p;
      if r < t1 then TEST := false
                else if r > t2 then t2 := r
   end
   else { úsečka je rovnoběžná k hraně  p = 0 }
      if q < 0 then TEST := false
end { TEST };
begin
   t1 := 0;  t2 := 1;
   dx := xs - xr;
   if TEST ( -dx, xr-xmin, t1, t2 ) then
     if TEST ( dx, xmax-xr, t1, t2 ) then
     begin dy := ys - yr;
        if TEST ( -dy, yr-ymin, t1, t2 ) then
          if TEST ( dy, ymax-yr, t1, t2) then
          begin
             if t1 > 0 then
             begin xr := xr + dx * t1;
```

```
                yr := yr + dy * t1
            end;
            if t2 < 1 then
            begin xs := xr + dx * t2;
                yr := yr + dy * t2
            end;
            LINE ( xr, yr, xs, ys )
        end
    end
end { LIANG-BARSKY };
{ (xr,yr),(xs,ys) jsou koncové body oříznuté úsečky }
```

Algoritmus 2.4.1

Výhodou Liang-Barského algoritmu je nízký počet operací násobení, dělení a též nezanedbatelná úspora vzniklá eliminací kódování pozice koncových bodů úsečky.

Dosud uvedené algoritmy ořezávání jsou omezeny na ořezávání obdélníkem, jehož hrany jsou rovnoběžné s osami souřadného systému. V technických aplikacích je však nutné ořezávání vůči konvexnímu n-úhelníku, což dosud uvedené algoritmy neumožňovaly.



Obr.2.5.1

2.5. Orezávání konvexním n-úhelníkem

Předpokládejme, že je dán konvexní n-úhelník, a to s orientací ve směru nebo proti směru hodinových ručiček, a uvažme různé situace, které mohou nastat, viz obr.2.5.1. Jednotlivé průsečíky přímky w resp. úsečky P_rP_s s konvexním n-úhelníkem získáme řešením soustavy lineárních rovnic, neboť pro ořezávanou úsečku platí rovnice

$$x(q) = x_r + (x_s - x_r) . q \qquad q \in \langle 0 , 1 \rangle$$

a pro hrany n-úhelníka platí

$$x(p) = x_i + (x_{i+1} - x_i) . p \qquad p \in \langle 0 , 1)$$

$$i = 0 , 1 , \ldots , n-1$$

kde operátor + (u indexů) značí sčítání modulo n.

Z důvodů jednoznačnosti, abychom v případě průchodu přímky vrcholem oblasti měli pouze jeden průsečík, je definiční interval pro parametr p polouzavřený. Vlastní algoritmus je založen na tom, že konvexní n-úhelník může mít nejvýše dva průsečíky s přímkou w (s výjimkou totožnosti s hranou), viz obr.2.5.1. Nejdříve se naleznou hodnoty parametru q odpovídající průsečíkům přímky s konvexním n-úhelníkem (označené t_i) a poté se určí společná část takto získané úsečky s úsečkou P_rP_s. Hodnoty t_1, t_2 získané algoritmem je nutné dosadit do příslušné rovnice pro x(q), abychom obdrželi souřadnice x, y koncových bodů oříznuté úsečky. Vlastní algoritmus byl částečně zjednodušen, neboť předpokládá, že neexistují dvě hrany n-úhelníka takové, které by ležely na společné přímce. V případě, že tato možnost může nastat, je nutné podmínku označenou ζ redukovat na podmínku i > n a sekvenci označenou ξ je nutné nahradit sekvencí jinou, viz algoritmus 2.5.2.

```
var i, j, k: integer;
    t: array [1..2] of real;
begin
   j := 0;  i := 0;  k := n - 1;
   repeat
      if existuje průsečík hrany xkxi a přímky w(q) tak,
         že p ∈ <0,1) then
```

```
      begin j:=j+1;
            t[j]:=q { ulozeni hodnoty q }
      end
      else
         if hrana x_k x_i lezi na w(q) then
         begin { sekvence ζ }
            t[1]:=hodnota q odpovidajici vrcholu x_k;
            t[2]:=hodnota q odpovidajici vrcholu x_i;
            j:=2 { konec sekvence ζ }
         end;
      k := i; i := i + 1
   until ( j = 2 ) or ( i > n ); { podminka ξ }
   if j <> 0 then
   begin if j = 1 then t[1]:=t[2] { primka w(q) se dotyka }
                  else if t[1] > t[2] then t[1] swap t[2];
         t[1] := max ( 0.0, t[1] ); { maximalni hodnota }
         t[2] := min ( 1.0, t[2] ); { minimalni hodnota }
          if t[1] ≤ t[2]  then LINE ( x(t[1]) , x(t[2]) )
   end
end;
```

Algoritmus.2.5.1.

```
if j = 0 then begin t_1 := q_1;  t_2 := q_2 end
         else begin if t_1 > t_2 then t_1 swap t_2;
                    if q_1 > q_2 then q_1 swap q_2;
                    t_1 := min ( q_1 ,t_1 );
                    t_2 := max ( q_2 ,t_2 )
              end;
```

Algoritmus 2.5.2.

2.6. Cyrus-Beckův algoritmus pro ořezávání konvexním n-úhelníkem

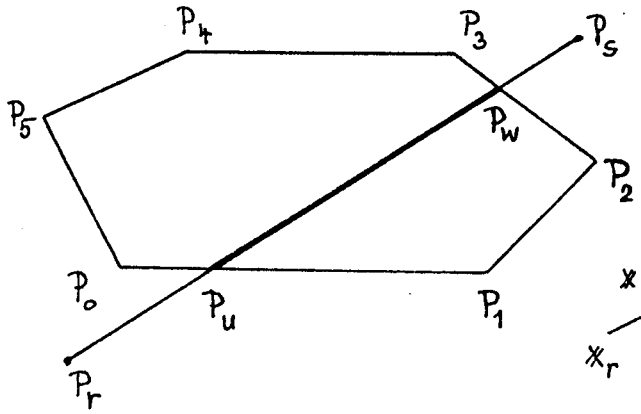
Cyrus-Beckův algoritmus je založen na znalosti normál hran konvexního n-úhelníka a jejich orientace. Je známo, že přímka může protínat hranice konvexního n-úhelníka nejvýše ve dvou bodech. Vyjádříme-li úsečku z bodu P_r o souřadnicích $\mathbf{x}_r$ do bodu P_s o souřadnicích $\mathbf{x}_s$ parametricky, dostáváme

$$\mathbf{x}(t) = \mathbf{x}_r + (\mathbf{x}_s - \mathbf{x}_t) \,.\, t \qquad 0 \le t \le 1$$

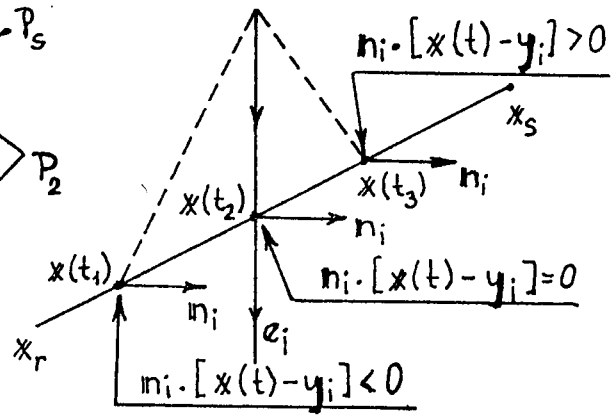
nebo po rozepsání:

$$x(t) = x_r + (x_s - x_r) \,.\, t$$

$$y(t) = y_r + (y_s - y_r) \,.\, t \qquad 0 \le t \le 1$$



Obr.2.6.1.



Obr.2.6.2.

Uvažujeme-li obecný konvexní n-úhelník, např. viz obr.2.6.1, vidíme, že se celý problém redukuje na nalezení dvou hran konvexní oblasti, které daná úsečka, resp. přímka, protíná, a na určení souřadnic průsečíků.

Je-li $\mathbf{e}_i$ orientovaná hrana konvexního n-úhelníka, $\mathbf{y}_i$ bod této hrany a $\mathbf{n}_i$ normála této hrany, pak směr výsledného vektoru $\mathbf{x}(t)-\mathbf{y}_i$ lze určit pomocí znaménka výsledku skalárního součinu, viz obr. 2.6.2, a to

je-li $\mathbf{n}_i \,.\, [\, \mathbf{x}(t)-\mathbf{y}_i \,] < 0$, pak vektor $\mathbf{x}(t)-\mathbf{y}_i$ směřuje ven z konvexní oblasti,

je-li $\mathbf{n}_i \cdot [\, \mathbf{x}(t) - \mathbf{y}_i \,] = 0$, pak vektor $\mathbf{x}(t) - \mathbf{y}_i$ je kolmý na normálu,

je-li $\mathbf{n}_i \cdot [\, \mathbf{x}(t) - \mathbf{y}_i \,] > 0$, pak vektor $\mathbf{x}(t) - \mathbf{y}_i$ směřuje dovnitř konvexní oblasti.

Dosadíme-li za $\mathbf{x}(t)$, pak dostáváme výraz

$$\mathbf{n}_i \cdot [\, \mathbf{x}_r + (\mathbf{x}_s - \mathbf{x}_r) \cdot t - \mathbf{y}_i \,] = 0$$

jako podmínku pro bod průsečíku s hranou, tj:

$$\mathbf{n}_i \cdot [\, \mathbf{x}_r - \mathbf{y}_i \,] + \mathbf{n}_i \cdot [\, \mathbf{x}_s - \mathbf{x}_r \,] \cdot t = 0$$

Označíme-li

$$\mathbf{d} = \mathbf{x}_s - \mathbf{x}_r \qquad \mathbf{w}_i = \mathbf{x}_r - \mathbf{y}_i$$

pak $\mathbf{d}$ určuje směr úsečky $\mathbf{x}_r\mathbf{x}_s$, zatímco vektor $\mathbf{w}_i$ je úměrný vzdálenosti počátečního bodu od hranice obrysu. Lze tedy psát

$$t \cdot \mathbf{n}_i \cdot \mathbf{d} + \mathbf{w}_i \cdot \mathbf{n}_i = 0$$

a tedy

$$t = - \frac{\mathbf{w}_i \cdot \mathbf{n}_i}{\mathbf{n}_i \cdot \mathbf{d}} \quad , \text{ je-li } \mathbf{d} \neq \mathbf{0} \quad \text{a} \quad i = 1, 2, \ldots , n$$

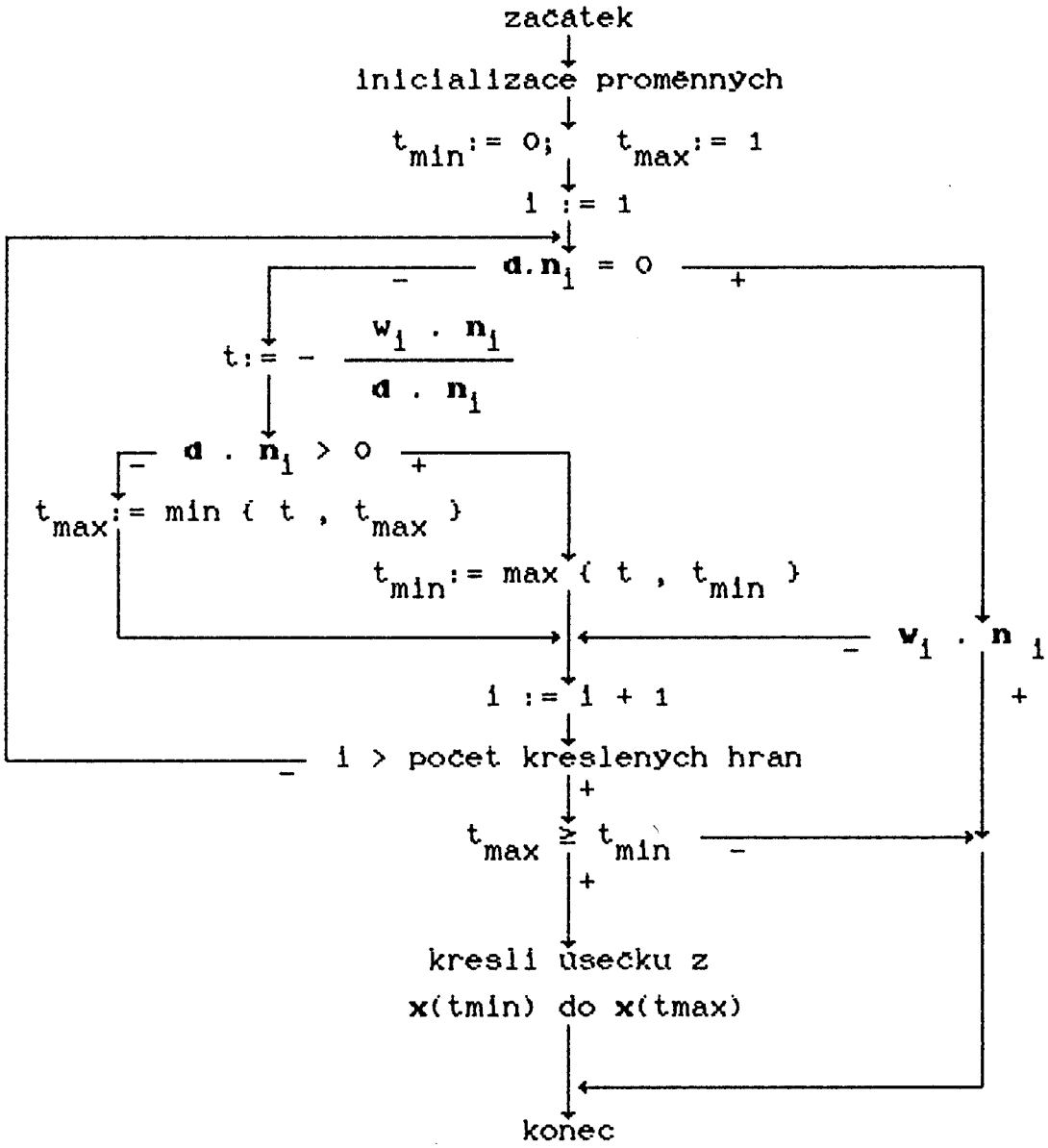
V případě, že $\mathbf{d} = \mathbf{0}$, je úsečka $\mathbf{x}_r\mathbf{x}_s$ nulové délky.

Je-li:

$$\mathbf{w}_i \cdot \mathbf{n}_i \begin{cases} < 0 & , \text{ bod je vně oblasti} \\ = 0 & , \text{ bod je na hranici oblasti} \\ > 0 & , \text{ bod je uvnitř oblasti} \end{cases}$$

Je zřejmé, že budeme uvažovat jen ty průsečíky, pro které bude platit, že $t \in \langle 0 , 1 \rangle$. Celý algoritmus je pak znázorněn na obr.2.6.3.

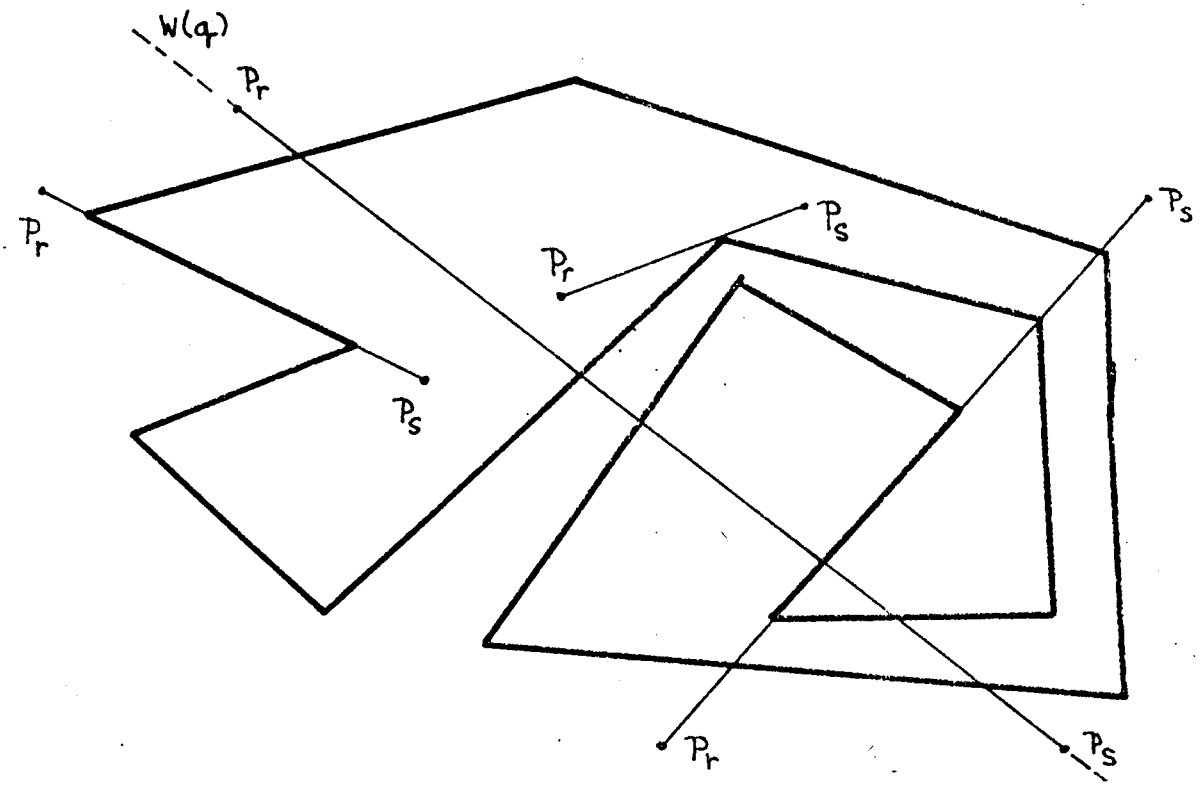
Nevýhodou uvedeného algoritmu je nutnost výpočtu normálového vektoru pro každou hranu dané oblasti v závislosti na orientaci konvexního n-úhelníka reprezentujícího danou oblast, podrobně viz [109]. Algoritmy založené na jiných přístupech lze nalézt např. v [80], [95].


začátek
inicializace proměnných
$t_{min} := 0$; $t_{max} := 1$
i := 1
$\mathbf{d} \cdot \mathbf{n}_i = 0$
−
+
$t := -\frac{w_i \cdot n_i}{\mathbf{d} \cdot \mathbf{n}_i}$
$\mathbf{d} \cdot \mathbf{n}_i > 0$
−
+
$t_{max} := \min \{ t , t_{max} \}$
$t_{min} := \max \{ t , t_{min} \}$
$w_i \cdot n_i$
−
i := i + 1
+
i > počet kreslených hran
−
+
$t_{max} \geq t_{min}$
−
+
kresli úsečku z
x(tmin) do x(tmax)
konec

Obr.2.6.3

2.7. Orezávání nekonvexním n-úhelníkem

Algoritmus pro orezávání nekonvexním n-úhelníkem je založen na parametrickém vyjádření úseček. Oproti algoritmu, který byl uveden v kap.2.5, však musí být řešeny jisté speciální případy. Uvažme situaci na obr.2.7.1 a několik možných případů orezávání.



Obr.2.7.1

Předpokládejme, že hrany nekonvexního n-úhelníka jsou opět dány ve směru anebo proti směru hodinových ručiček, a že se vzájemně neprotínají.

Pro jednoduchost předpokládejme:

- pouze hrany sousední mají společný bod,
- sousední hrany neleží na společné přímce,
- vrcholy jsou navzájem různé.

Dále uvedený algoritmus lze modifikovat i pro n-úhelníky nesplňující výše uvedené předpoklady.

Označme $x(q)$ souřadnice bodů přímky w a vyjádřeme je parametricky jako

$$\mathbf{x}(q) = \mathbf{x}_r + (\mathbf{x}_s - \mathbf{x}_r) \,.\, q \qquad q \in (-\infty , \infty)$$

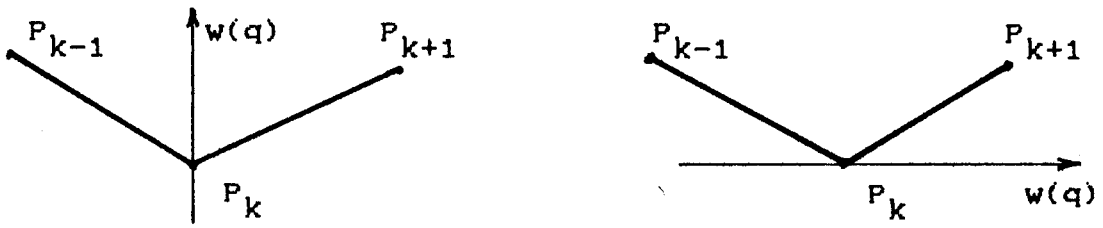
a souřadnice bodů $\mathbf{x}(p)$ každé hrany n-úhelníka jako

$$\mathbf{x}(p) = \mathbf{x}_i + (\mathbf{x}_{i+1} - \mathbf{x}_i) \,.\, p \qquad p \in \langle 0 , 1 \rangle$$
$$i = 0 , 1 , \ldots , n-1$$

Budeme zatím hledat průsečíky hran n-úhelníka s přímkou $\mathbf{w}$, na které leží ořezávaná úsečka P_rP_s. Je nutné poznamenat, že operace + u indexů značí operaci modulo n.

Kromě průsečíku přímky s hranou musíme rozlišit případy, kdy hrana n-úhelníka leží na přímce $\mathbf{w}(q)$ a kdy přímka $\mathbf{w}(q)$ prochází vrcholem. Při průchodu přímky $\mathbf{w}(q)$ vrcholem mohou nastat následující případy:

$[\mathbf{s}_1 \times \mathbf{s}_2]_z . [\mathbf{s}_3 \times \mathbf{s}_2]_z > 0$ $\qquad$ $[\mathbf{s}_1 \times \mathbf{s}_2]_z . [\mathbf{s}_3 \times \mathbf{s}_2]_z < 0$



a) "průchod" $\qquad$ b) "dotyk"

kde + v indexech značí součet modulo n
- v indexech značí rozdíl modulo n

a $\mathbf{s}_1 = \mathbf{x}_k - \mathbf{x}_{k-1}$ $\qquad$ $\mathbf{s}_2 = \mathbf{x}_s - \mathbf{x}_r$ $\qquad$ $\mathbf{s}_3 = \mathbf{x}_{k+1} - \mathbf{x}_k$

Obr.2.7.2

V případě ad a) se generuje pouze jeden průsečík, zatímco v případě ad b) se generují dva totožné body. V obou případech se průsečíky považují z hlediska dalšího zpracování za průsečíky s hranou. Ve skutečnosti se generují pouze hodnoty parametru q, které odpovídají poloze bodu P_k na přímce $\mathbf{w}(q)$.

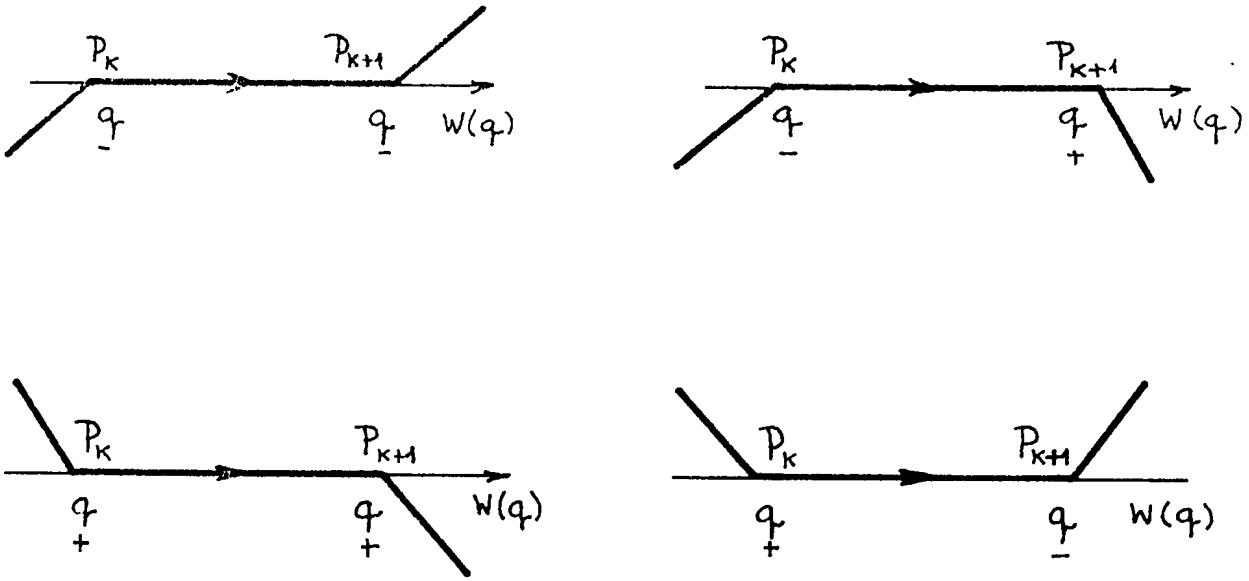
V případě, že na přímce w(q) leží některá hrana, mohou nastat situace, které jsou znázorněny na obr.2.7.3. V těchto případech není možné ihned rozhodnout, jaké hodnoty parametru q mají být generovány. Z tohoto důvodu musí být generován speciální atribut parametru q, který je určen znaménkem souřadnice z vektorového součinu vektorů $\mathbf{s}_1$ a $\mathbf{s}_2$, resp. $\mathbf{s}_3$ a $\mathbf{s}_2$.

Průsečík bude určen nejen hodnotou q, ale též hodnotou atributu, který je dán jako

␣ (mezera) pro průsečík s hranou

\+ nebo - znaménkem souřadnice z vektorového součinu $[s_1 \times s_2]$, resp. $[s_3 \times s_2]$ pro "dotyk" nebo "průchod" vrcholem.

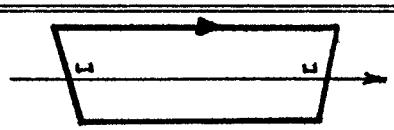
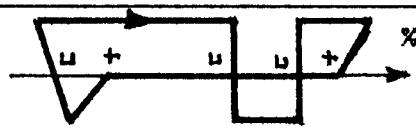
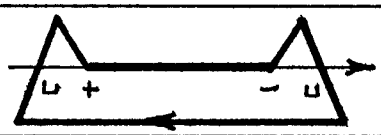
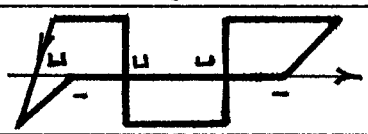
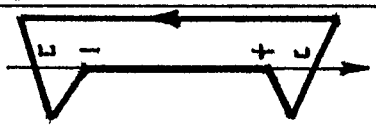
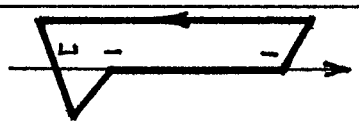
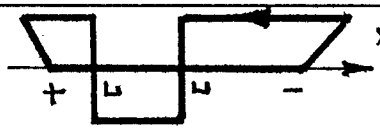
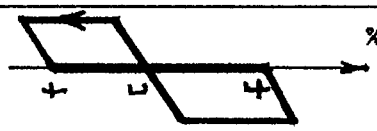
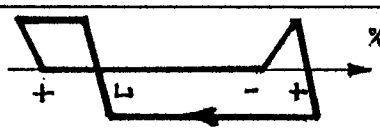
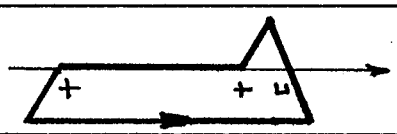
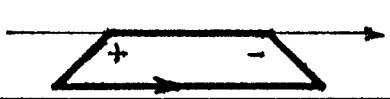
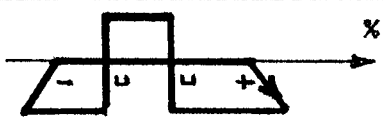
Po nalezení všech průsečíků, včetně určení jejich atributů, musí být získaná množina hodnot q setříděna opět spolu s atributy. V následujícím kroku je nutné provést redukci získaných hodnot q podle tab.2.7.1. Výsledkem je pak množina dvojic hodnot q, které určují úseky přímky w(q) a které leží uvnitř n-úhelníka. Pro určení úseků úsečky P_rP_s, které leží uvnitř n-úhelníka, je nutné vyhodnotit průnik intervalu <0,1> s jednotlivými intervaly, které jsou dány po sobě jdoucími dvojicemi hodnot q. Celý postup může být realizován algoritmem 2.7.1.

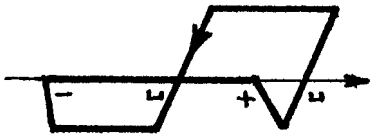
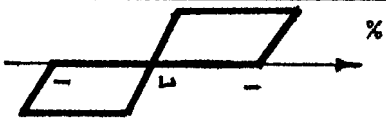
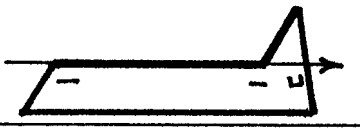


Obr.2.7.3

```
k:=n-1; i:=0;
s1 := xk - xk-1;   s2 := xs - xr; { s a x jsou vektory }
while i < n do
begin
   s3:= xi - xk;  { operace s vektory }
   Vypočti hodnotu (q);
   if vrchol xk leží na přímce w(q)
   then
   begin
      if [s3 x s2]z = 0
      then { hrana xkxi leží na přímce w(q) }
         Generuj (q s atributem sign ( [s1 x s2]z)
      else if [s1 x s2]z = 0
           then { hrana xk-1xk leží na přímce w(q) }
             Generuj (q s atributem sign ( [s3 x s2]z)
           else if [s1 x s2]z . [s3 x s2]z < 0
                then { typ dotyk }
                   Generuj ( q ,q s atributem ⌣ )
                else { typ průchod }
                  Generuj ( q s atributem ⌣ )
   end
   else
   if existuje-li průsečík přímky w(q) s hranou <xk,xi)
        then Generuj (q s atributem ⌣ );
   s1:=s3; k:=i; i:=i+1
end;
SORT( získané hodnoty q );
REDUKUJ ( hodnoty q podle tabulky );
VYBER ( podintervaly jako <qj,qj+1> ∩ <0,1>  ∀ j );
```

Algoritmus 2.7.1

atributy q_i	q_{i+1}	q_{i+2}	situace	činnost
␣	␣	*		uloz(q_i,q_{i+1}); i:=i+2
␣	+	␣		% uloz(q_i,q_{i+2}); i:=i+2 změň atribut q_i na +
␣	+	+		uloz(q_i,q_{i+2}); i:=i+3
␣	+	-		uloz(q_i,q_{i+2}); i:=i+2 změň atribut q_i na .
␣	-	␣		% uloz(q_i,q_{i+2}); i:=i+2 změň atribut q_i na -
␣	-	+		uloz(q_i,q_{i+2}); i:=i+2 změň atribut q_i na ␣
␣	-	-		uloz(q_i,q_{i+2}); i:=i+3
+	␣	␣		% uloz(q_i,q_{i+2}); i:=i+2 změň atribut q_i na +
+	␣	+		% uloz(q_i,q_{i+2}); i:=i+3
+	␣	-		% uloz(q_i,q_{i+2}); i:=i+2 změň atribut q_i na ␣
+	+	*		uloz(q_i,q_{i+1}); i:=i+1 změň atribut q_i na ␣
+	-	*		uloz(q_i,q_{i+1}); i:=i+2
-	␣	␣		% uloz(q_i,q_{i+2}); i:=i+2 změň atribut q_i na -

-	⊔	+	 % ulož(q_i,q_{i+2}); i:=i+2 změň atribut q_i na ⊔
-	⊔	-	 % ulož(q_i,q_{i+2}); i:=i+3
-	+	*	 ulož(q_i,q_{i+1}); i:=i+2
-	-	*	 ulož(q_i,q_{i+1}); i:=i+1 změň atribut q_i na ⊔

* znamená všechny případy, tj. ⊔ + -

% případy, kdy se hrany nebo vrcholy dotýkají, nebo se hrany protínají

Tabulka 2.7.1

Příkaz pro výběr podintervalů může být realizován např. algoritmem 2.7.2.

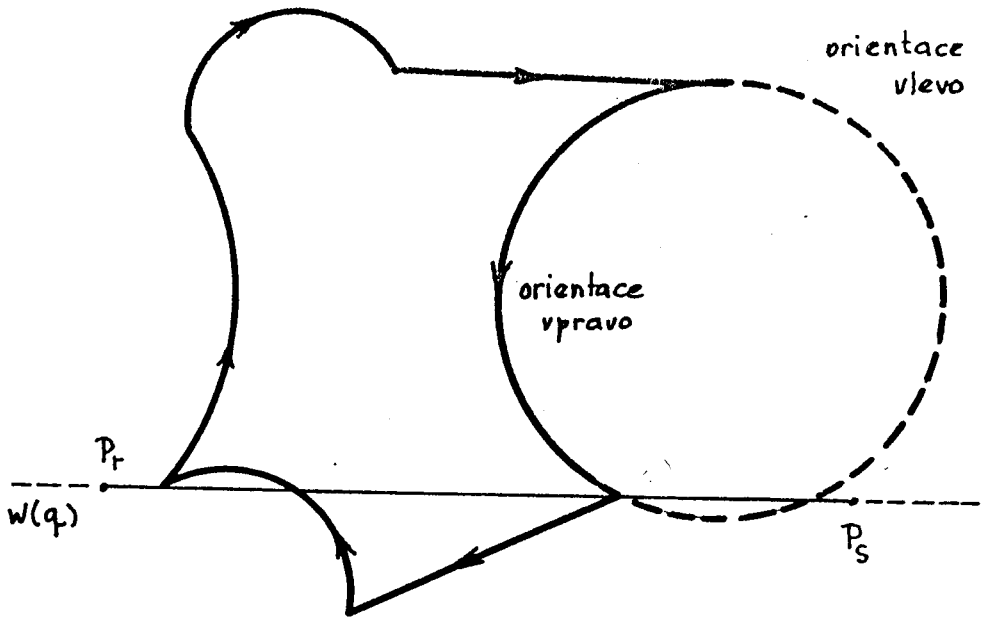
```
i:=1;
while i ≤ počet průsečíků - 1 do
begin if max ( 0 , q_i ) ≤ min ( 1 , q_{i+1} )
      then ulož ( max ( 0 , q_i ) , min ( 1 , q_{i+1} ) );
      i:=i+2
end;
```

Algoritmus 2.7.2

Je zřejmé, že uvedený algoritmus je rozšířením algoritmu z kap.2.5, přičemž je nutné použít redukci a třídění vzhledem k tomu, že n-úhelník je nekonvexní (podrobně viz. [64], [117]).

2.8. Orezávání nekonvexních oblastí

Až dosud uváděné algoritmy umožňovaly orezávání úseček či přímek vzhledem k n-úhelníkům, tj. vzhledem k oblastem, jejichž hranice byly tvořeny úsečkami. Nicméně existuje poměrně široká třída úloh, kde je vhodné orezávat úsečky vůči oblasti s hranicemi tvořenými oblouky.



Obr.2.8.1

Předpokládejme, že oblast je dána posloupností vrcholů ve směru nebo proti směru hodinových ručiček. Není-li hrana lineární, pak kromě poloměru a pozice středu oblouku je dána i informace o tom, která část kružnice (pravá nebo levá vzhledem k počátečnímu bodu kruhového oblouku) má být vzata v úvahu. Pro zjednodušení algoritmu uvažme následující omezení

- všechny vrcholy mají navzájem různé souřadnice,
- žádný vrchol neleží hraně nebo kruhovém oblouku,
- dvě hrany se nedotýkají, pokud nejsou sousední,
- dvě sousední hranice oblasti, tj. hrany či oblouky, mohou mít pouze vrchol jako společný bod.

Na rozdíl od algoritmu z kap.2.7 může mít přímka $\mathbf{w}(q)$ s kruhovým obloukem dva průsečíky, což poněkud komplikuje řešení problému. Postup pro nalezení všech průsečíků je obdobný, avšak v případě kruhového oblouku musíme řešit následující soustavu rovnic vzhledem k proměnné q

$$\mathbf{x}(q) = \mathbf{x}_r + (\mathbf{x}_s - \mathbf{x}_r) . q \qquad q \in (-\infty , +\infty)$$

$$(x - x_u)^2 + (y - y_u)^2 - r^2 = 0$$

kde (x_u, y_u) je střed kružnice a r je její poloměr.

Řešením obdržíme kvadratickou rovnici pro q

$$aq^2 + bq + c = 0$$

kde
$$a = (x_s - x_r)^2 + (y_s - y_r)^2$$

$$b = 2[(x_r - x_u).(x_s - x_r) + (y_r - y_u).(y_s - y_r)]$$

$$c = (x_r - x_u)^2 + (y_r - y_u)^2 - r^2$$

V případě, že přímka $\mathbf{w}(q)$ danou kružnici protíná nebo se jí dotýká, obdržíme řešením rovnice obecně dva reálné kořeny, a to:

$$q_{1,2} = \frac{- b \pm \sqrt{b^2 - 4ac}}{2 a}$$

Nyní je nezbytné určit, které průsečíky leží na části kružnice tvořící hranici dané oblasti. Pomocí testu lze zjistit, zdali průsečík leží vpravo či vlevo od spojnice počátečního a koncového bodu kruhového oblouku. To znamená, že

- je-li oblouk orientován doprava, bude bod $\mathbf{x}(q_i)$ uvažován tehdy a jen tehdy, je-li $[\mathbf{s}_1 \times \mathbf{s}_2]_z < 0 \quad i=1,2$
- je-li oblouk orientován doleva, bude bod $\mathbf{x}(q_i)$ uvažován tehdy a jen tehdy, je-li $[\mathbf{s}_1 \times \mathbf{s}_2]_z > 0 \quad i=1,2$

přičemž $\mathbf{x}_k \neq \mathbf{x}(q_i)$, $\mathbf{s}_1 = \mathbf{x}_{k+1} - \mathbf{x}_k$, $\mathbf{s}_2 = \mathbf{x}(q_i) - \mathbf{x}_k$

Je zřejmé, že opět musí být řešeny speciální případy, kdy např. přímka $\mathbf{w}(q)$ prochází vrcholem $\mathbf{x}_k$. V těchto případech budou vyhodnoceny vektory $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{s}_3$ takto:

- pro oblouk $s_1 = [\ y_k - y_u\ ,\ x_u - x_k\ ]^T$, kde (x_u, y_u) je střed kružnice

  pro hranu $s_1 = [\ x_k - x_{k-1}\ ,\ y_k - y_{k-1}\ ]^T$,

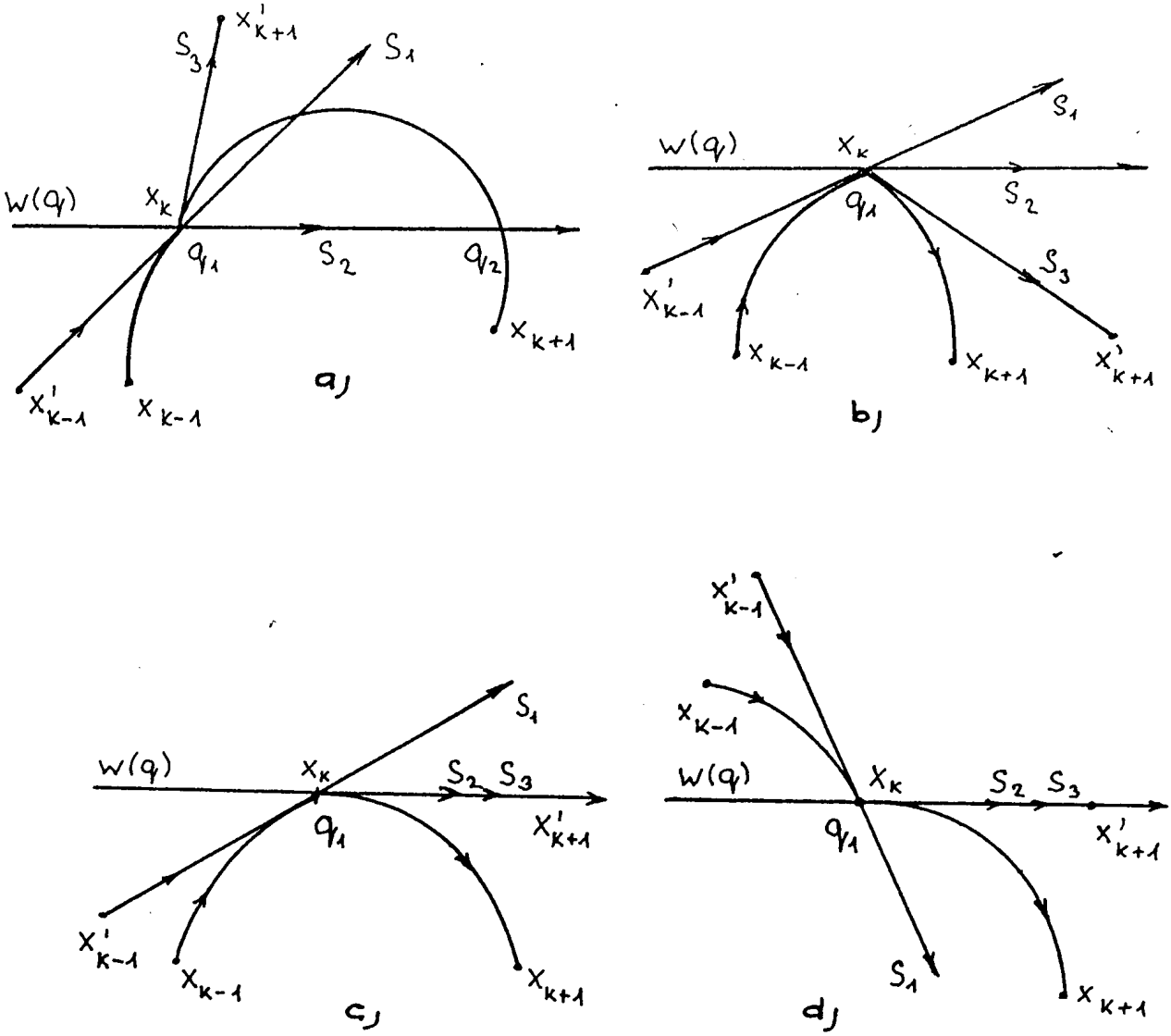
  tj. $s_1 = x_k - x_{k-1}$

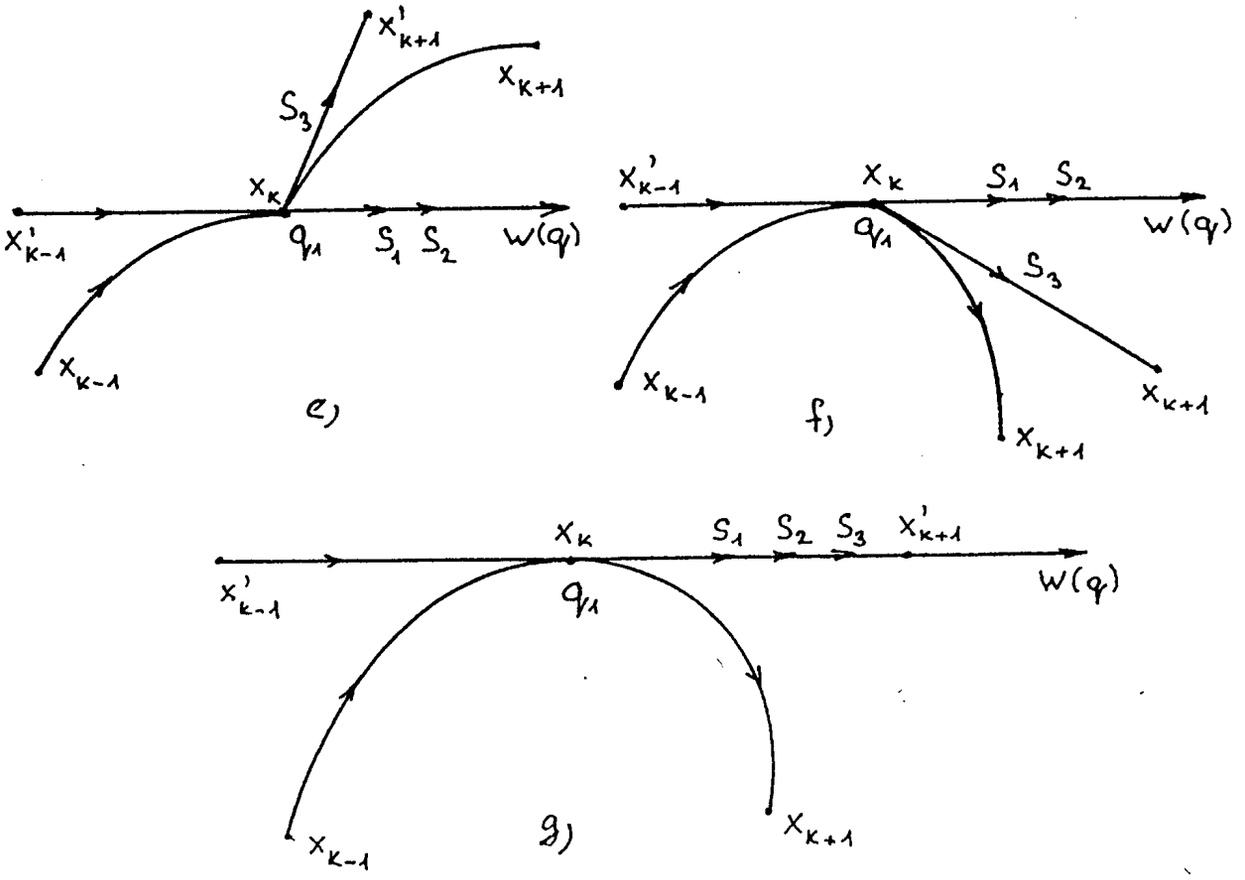
- pro oblouk $s_3 = [\ y_k - y_w\ ,\ x_w - x_k\ ]^T$, kde (x_w, y_w) je střed kružnice

  pro hranu $s_3 = [\ x_k - x_{k-1}\ ,\ y_k - y_{k-1}\ ]^T$,

  tj. $s_3 = x_k - x_{k-1}$

Některé možné situace jsou uvedeny na obr.2.8.2 a v tabulce 2.8.1. pak pravidla pro jejich vyhodnocení.





Obr.2.8.2

Je-li oblouk orientován doprava, pak musí být změněno znaménko souřadnice z příslušného vektorového součinu. Pak můžeme definovat hodnoty proměnných a , b pomocí sekvencí:

$a := [\mathbf{s}_1 \times \mathbf{s}_2]_z$;

if $\mathbf{x}_{k-1}\mathbf{x}_k$ je oblouk
  then if a = 0
    then a := - $\mathbf{s}_1 . \mathbf{s}_2$
    else if orientace oblouku je doprava
      then a := -a;

```
b:=[s3 x s2]z;
if xk-1xk je oblouk
   then if b = 0
      then b :=-s3. s2
      else if orientace oblouku je doprava
              then b := -b;
```

Cely postup orezávání úsečky nekonvexní oblastí je možné realizovat např. algoritmem 2.8.1.

$[s_1 \times s_2]_z$	$[s_3 \times s_2]_z$	situace na obr.2.8.2	typ dotyk/průchod
< 0	< 0	a	průchod
< 0	> 0	b	dotyk
> 0	> 0	+ a	průchod
> 0	< 0	+ b	dotyk
< 0	= 0	c	if-$s_3.s_2$>0 **then** průchod **else** dotyk
> 0	= 0	d	if-$s_3.s_2$>0 **then** dotyk **else** průchod
= 0	< 0	e	if-$s_1.s_2$>0 **then** dotyk **else** průchod
= 0	> 0	f	if-$s_1.s_2$>0 **then** průchod **else** dotyk
= 0	= 0	g	if-$s_1.s_2$>0 **xor** $s_3.s_2$>0 **then** průchod **else** dotyk

\+ pro opačnou orientaci přímky w(q)

Tabulka 2.8.1

```
procedure Comp ( xA, xB: vector; var r: real; t: boolean );
begin
   if xAxB je lineární
   then begin s := xB - xA;  r := [ s x s2]z end
   else
```

```
   begin s :=[ y_k - y_w , x_w - x_k ]^T;
         r := [ s x s_2]_z;
         if r = 0
         then if t then r := s . s_2
                   else r := - s . s_2
         else if oblouk orientován doprava
              then r := -r
   end
end { Comp };
{ tělo vlastního algoritmu }
k := n - 1;  i := 0;
s_2 := x_s - x_r;
while i < n do
begin
   if x_k leží na přímce w(q) then
   begin Comp ( x_k , x_i , a , true );
         Comp ( x_k-1 , x_k , b , false );
         if x_k x_i je lineární then
         begin { předpokládá se, že  x_k = x(q) }
            Výpočet hodnoty ( q );
            if a*b > 0
            then Generuj( q s atributem ⌣ )
            else if a*b < 0
                 then Generuj( q, q s atributem ⌣ )
                 else if a = 0
                      then Generuj( q s atributem sign b )
                      else Generuj( q s atributem sign a )
         end
         else { předpokládá se, že  x_k = x(q_1) }
         begin
            Výpočet hodnot ( q_1 , q_2 );
            if a*b > 0
            then Generuj( q_1 s atributem ⌣ )
            else if a*b < 0
                 then Generuj( q_1, q_1 s atributem ⌣ )
                 else if a = 0
                      then Generuj( q_1 s atributem sign b )
```

```
                    else Generuj( q_1 s atributem sign a );
            Generuj( q_2* s atributem ⌣ );
        end
    end
    else if x_k x_i je lineární then
        begin Výpočet hodnoty ( q );
              if průsečík je uvnitř <x_k,x_i>
              then Generuj( q  s atributem ⌣ )
        end
        else begin Výpočet hodnot ( q_1 , q_2 );
                   if průsečík existuje
                   then Generuj( q_1*, q_2* s atributem ⌣ )
 (* značí, že průsečík leží na požadované straně oblouku x_k x_i )
             end;
   k := i;  i := i + 1;
end ( while );
SORT( získané hodnoty q );
REDUKUJ ( hodnoty q podle tabulky );
VYBER ( podintervaly jako <q_j,q_j+1> ∩ <0,1>  ∀ j );
```

Algoritmus 2.8.1

Uvedený algoritmus je možné modifikovat i pro eliptické oblouky a i pro obecný případ kuželosečky f(x,y)=0, kde:

$$f(x,y) = ax^2 + by^2 + 2cxy + 2dx + 2ey + g$$

přičemž směrový vektor **s** je dán:

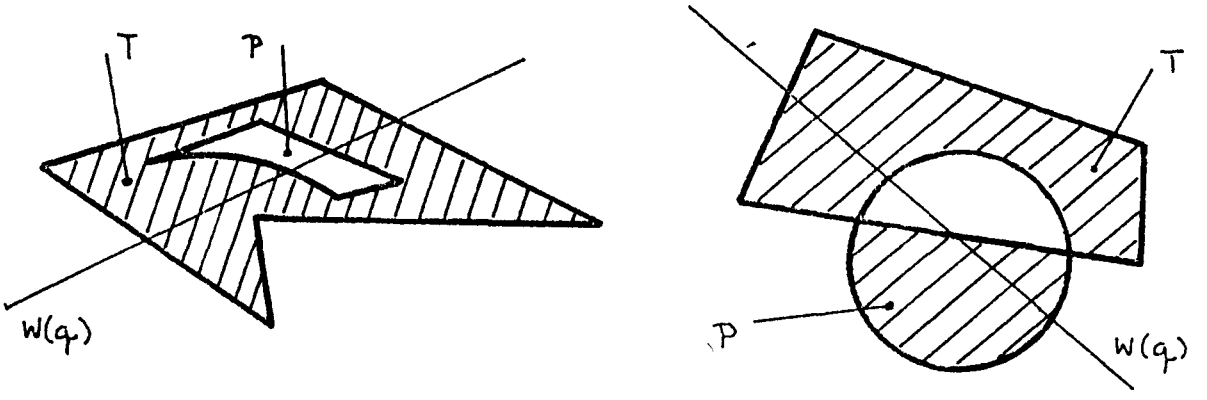
$$\mathbf{s} = [\ f_y\ ,\ -f_x\ ]^T$$

Uvedený algoritmus je možné použít i v případě oblastí, které mají "díry" s tím, že algoritmus se aplikuje na vnější hranici a od získaných dvojic hodnot q se odečtou intervaly dvojic hodnot q získaných aplikací algoritmu na hranice děr.

V případě modifikace algoritmu pro šrafování je vhodné nejdříve pootočit oblast tak, aby šrafy byly rovnoběžné s osou x, resp. osou y, spočítat průsečíky šrafy s hranicí oblasti a pak provést zpětné pootočení. Tím se podstatně zjednoduší výpočty jednotlivých průsečíků úsečky s hranicí oblasti.

2.9. Orezávání úseček složenými oblastmi

Dosud uvedené algoritmy řešily problematiku orezávání úsečky obecné nekonvexní oblasti, jejíž hranice byla tvořena úsečkami a kruhovými oblouky. V technické praxi se však vyskytují i útvary, které obsahují např. díry. Je zřejmé, že tento případ lze jednoduše vyřešit pomocí množinové operace rozdílu, kdy od oblasti T reprezentující oblast bez díry odečteme množinovou oblast reprezentující díru P, viz obr.2.9.1 Výslednou oblast C budeme nazývat složenou oblastí.



C = T - P

a

C = T Δ P

b

Obr.2.9.1

Stejně jako operace rozdílu lze použít i ostatních množinových operací

- sjednocení, ozn. ∪
- průniku, ozn. ∩
- rozdílu, ozn. - , resp. symetrického rozdílu, ozn. Δ

Je zřejmé, že v krajním případě je možné nahlížet na

- konvexní n-úhelník jako na sjednocení trojúhelníků,
- nekonvexní n-úhelník jako na rozdíl konvexní obálky a vhodných konvexních n-úhelníků, resp. jako na sjednocení konvexních n-úhelníků.

Vzhledem k řešené otázce orezávání vzniká otázka, jakým způsobem je nutné modifikovat dosud uvedené algoritmy, aby je bylo možné

použít i v případě množinových operací s oblastmi.

Předpokládejme, že P , T jsou oblasti a že složená oblast C je dána jako

$$C = T \ \mathbf{op} \ P$$

kde: $\mathbf{op} \in \{ \cup , \cap , - , \Delta \}$

Jsou-li části přímky **w**(q) oříznuté oblastí C dány množinou **c** hodnot dvojic parametrů q:

$$\mathbf{c} = \left\{ < c_k , c_{k+1} > \right\}_{k=1}^{L}$$

části přímky **w**(q) oříznuté oblastí T dány množinou **t** hodnot dvojic parametrů q:

$$\mathbf{t} = \left\{ < t_i , t_{i+1} > \right\}_{i=1}^{N}$$

části přímky **w**(q) oříznuté oblastí P dány množinou **p** hodnot dvojic parametrů q:

$$\mathbf{p} = \left\{ < p_j , p_{j+1} > \right\}_{j=1}^{M}$$

Pak platí, že:

$$\mathbf{c} = \mathbf{t} \ \mathbf{op} \ \mathbf{p}$$

Lze zjistit, že v případě nutnosti lze i složité oblasti definovat jako výsledek množinových operací nad konvexními n-úhelníky a kruhy, což však povede k částečnému nárůstu požadavků na výpočetní systém v důsledku provádění operací nad množinami dvojic hodnot parametrů q. Realizace operace sjednocení nad množinami dvojic hodnot parametrů q je naznačena algoritmem 2.9.1.

$i:=1$; $j:=1$; $k:=1$; $c_k:=+\infty$; $c_{k+1}:=-\infty$;

while $(i \leq N)$ **and** $(j \leq M)$ **do**
begin
  if $p_j < t_i$
  then if $\langle t_i, t_{i+1} \rangle \blacksquare \langle c_k, c_{k+1} \rangle \neq \phi$
    then begin $c_k := \min(c_k, t_i)$; $c_{k+1} := \max(c_{k+1}, t_{i+1})$;
      $i := i+2$
    end

```
        else begin k:=k+2; c_k:=+∞; c_{k+1}:=-∞; end
    else if <p_j,p_{j+1}> ■ < c_k,c_{k+1}> ≠ φ
        then begin c_k:= min (c_k,p_j); c_{k+1}:= max (c_{k+1},p_{j+1});
                   j:=j+2
             end
        else begin k:=k+2; c_k:=+∞; c_{k+1}:=-∞; end
end { while };
while i ≤ N do
    if <t_i,t_{i+1}> ■ < c_k,c_{k+1}> ≠ φ
    then
    begin c_k:= min (c_k,t_i); c_{k+1}:= max (c_{k+1},t_{i+1});
          i:=i+2
    end
    else begin k:=k+2; c_k:=+∞; c_{k+1}:=-∞; end;
while j ≤ M do
    if <p_j,p_{j+1}> ■ < c_k,c_{k+1}> ≠ φ
    then begin c_k:= min (c_k,p_j); c_{k+1}:= max (c_{k+1},p_{j+1});
               j:=j+2
         end
    else begin k:=k+2;
               c_k:=+∞; c_{k+1}:=-∞;
         end;
```

Algoritmus 2.9.1

Operátor ■ je definován jako ∪ (sjednocení) s tím, že interval $(+\infty,-\infty)$ se považuje za prázdný interval označený ϕ. Jde vlastně o sjednocení intervalů hodnot q získaných oříznutím přímky w(q) oblastí T a P.

Je-li oblast složená, pak je nezbytné řešit též otázku určení hranice oblasti. Tato úloha není již úlohou zcela triviální. Pokud budeme používat následující řešení, i když neefektivní, dostaneme pouze hrany ohraničující oblast výslednou, které však nejsou uspořádány ve směru nebo proti směru pohybu hodinových ručiček.

Pak v případě operace

$$A \; \mathbf{op} \; B \qquad \mathbf{op} \in \{ \cup , \cap , - , \Delta \}$$

je postup pro operaci:

- sjednocení: vezmi všechny hrany oblasti B a proveď "vnější" ořezávání oblastí A; vezmi všechny hrany oblasti A a proveď "vnější" ořezávání oblastí B
- průnik: vezmi všechny hrany oblasti B a proveď "vnitřní" ořezávání oblastí A; vezmi všechny hrany oblasti A a proveď "vnitřní" ořezávání oblastí B
- rozdíl: vezmi všechny hrany oblasti B a proveď "vnitřní" ořezávání oblastí A; vezmi všechny hrany oblasti A a proveď "vnější" ořezávání oblastí B
- symetrický rozdíl: lze realizovat pomocí výše uvedených operací

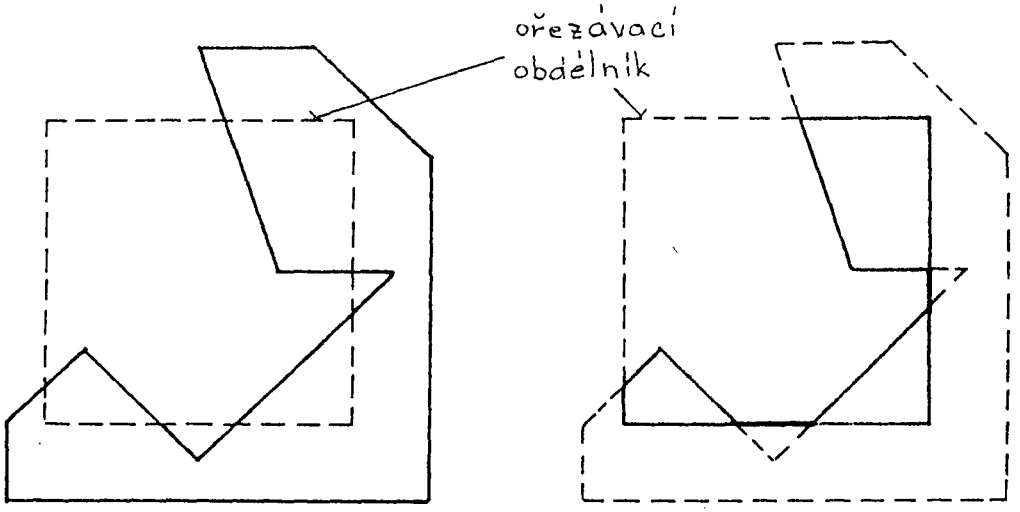
Je zřejmé, že takto lze definovat jen obrys oblastí, které vzniknou jednou množinovou operací z nesložených oblastí, přičemž navíc je nutné dosud uvedené algoritmy rozšířit o ořezávání kruhového oblouku nekonvexních oblastí, což vede k podstatnému zvýšení složitosti dosud uvedených algoritmů.

Vzhledem k tomu, že je požadováno, aby výsledkem po množinové operaci s n-úhelníky, resp. oblastmi, byl opět n-úhelník, resp. oblast, je nutné zabývat se takovými algoritmy ořezávání n-úhelníků, resp. oblastí, jejichž výsledkem je opět n-úhelník, resp. oblast.

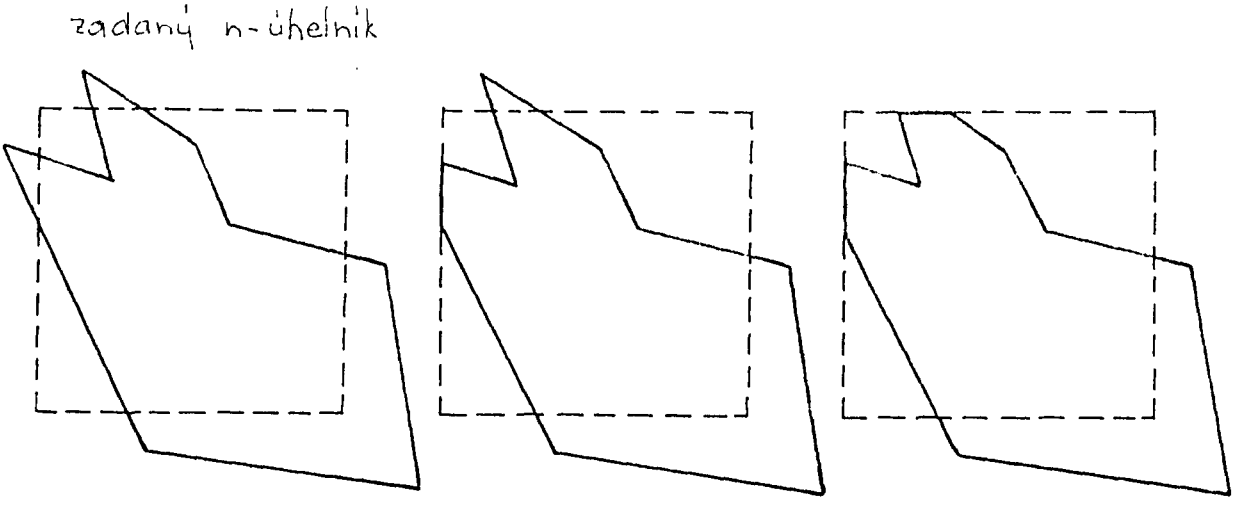
2.10. Sutherland-Hodgmanův algoritmus pro ořezávání

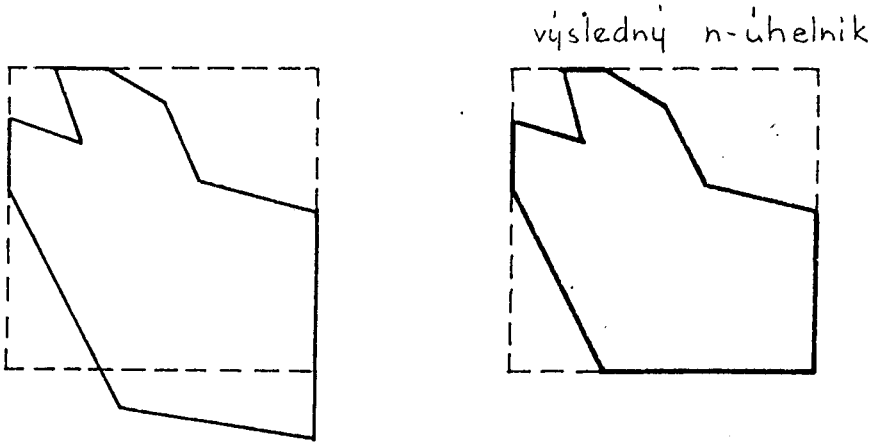
Na rozdíl od dříve uvedených algoritmů pro ořezávání přímek či úseček obdélníkem, konvexním či nekonvexním n-úhelníkem nebo oblastí algoritmy určené pro ořezávání n-úhelníka n-úhelníkem vytvářejí jako výstup opět n-úhelník, resp. n-úhelníky. Na obr.2.10.1 je uveden výsledek ořezávání nekonvexního n-úhelníka obdélníkem, kdy se původní n-úhelník rozpadl na tři nové n-úhelníky.

Základní myšlenkou Sutherland-Hodgmanova algoritmu (dále jen S-H algoritmus) je ořezávání n-úhelníka vůči jedné hraně či ořezávací rovině, přičemž se postupně provádí ořezávání hran n-úhelníka vůči jednotlivým hranám obdélníka, viz obr.2.10.2.



Obr.2.10.1





Obr.2.10.2

Dále uvedený algoritmus je založený na strategii postupného zkoumání jednotlivých hran n-úhelníka vzhledem k hranici oblasti, která musí být konvexní.

Pro snadnost pochopení algoritmu byly vynechány sekvence příkazů zajišťující ošetření chyb, výjimečných stavů a též zápis procedur INSIDE, INTERSECT a OUTPUT.

Procedura OUTPUT(q: **vertex**) zajistí uložení vrcholu q do výstupního pole out v reprezentujícího vrcholy vytvářeného n-úhelníka.

Funkce INTERSECT (s, p: **vertex**; q: **boundary**): **vertex** určí průsečík hrany spojující vrcholy s a p s hranicí q dané oblasti, viz obr.2.10.3.

Procedura INSIDE (s: **vertex**; q: **boundary**): **boolean** nabývá hodnoty **true**, je-li bod s uvnitř ořezávané oblasti. V případě, že hranice oblasti je zadána orientovaně, lze využít vektorového součinu k určení polohy bodu s vůči hraně q. Obecně platí, viz obr.2.10.4., že označíme-li

$$\mathbf{v} = \mathbf{a} \times \mathbf{c} \qquad\qquad \mathbf{w} = \mathbf{a} \times \mathbf{b}$$

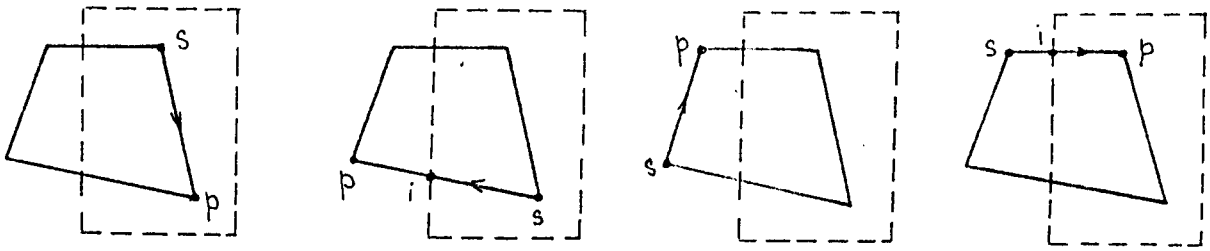
pak

- je-li $\mathbf{v}_z > 0$, tj. z-ová složka vektoru **v**, pak bod P_4 je vně ořezávacího n-úhelníka
- je-li $\mathbf{v}_z < 0$, tj. z-ová složka vektoru **v**, pak bod P_3 je uvnitř ořezávacího n-úhelníka

```
const max=100; { maximální počet vrcholů n-úhelníka }
type vertex = array [1..2] of real; { typ pro vrchol }
     boundary = array [1..2] of vertex; { typ pro hrany }
     polygon = array [1..max] of vertex; { typ pro n-úhelník }

procedure CLIP (in: integer; { počet vrcholů daného n-úhelníka }
                in v: polygon; { vrcholy vstupního n-úhelníka }
                var out: integer;
                { počet vrcholů výsledného n-úhelníka }
                var out v: polygon;
                { vrcholy výsledného n-úhelníka }
                clip boundary: boundary
                { hrana, vůči které se ořezává });
var i, p, s: vertex;     j: integer;
begin
   out:=0; { nastavení počtu vrcholů výsledného n-úhelníka }
   s:=in v[in]; { s a p reprezentují vrcholy n-úhelníka }
   for j:=1 to in do
   begin
      p:=in v[j];
      if INSIDE (p,clip boundary)
                { s a p odpovídají vrcholům na obr.2.10.3 }
      then { případ 1 a 4 }
           if INSIDE(s,clip boundary)
           then OUTPUT(p) { případ 1 }
           else begin { případ 4 }
                      i:=INTERSECT(s,p,clip boundary);
                      OUTPUT(i); OUTPUT(p)
                end
      else { případ 2 a 3 }
         if INSIDE(s,clip boudary) { případ 2 }
         then begin i:=INTERSECT(s,p,clip boundary);
                    output(i)
         end; { žádná akce pro případ 3 }
      s:=p; { vezmi další dvojici vrcholů }
   end { od j }
end { CLIP };
```

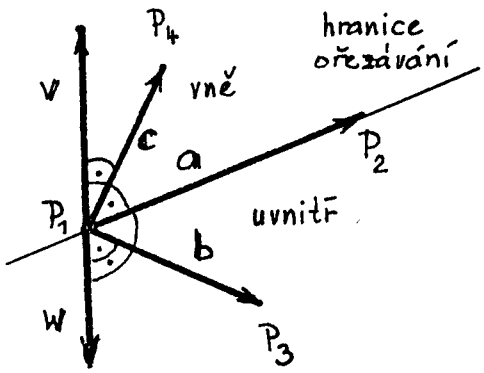
Algoritmus 2.10.1



ulož p ulož i nic neukládej ulož i,p

Obr.2.10.3

Výše uvedený algoritmus lze modifikovat tak, že je rekurzivně volán, a tím lze ušetřit dočasně přidělenou paměť, viz [125]. Algoritmus může být též realizován jako hardwarový procesor využívající "pipe-line" bez požadavků na vnější prostor. Nevýhodou Sutherland-Hodgmanova algoritmu je, že v případě oříznutí nekonvexního n-úhelníka vznikají hrany, které by správně neměly být zahrnuty do výsledného n-úhelníka. Tato skutečnost je dána tím, Sutherland-Hodgmanův algoritmus neposkytuje možnost "roztržení" n-úhelníka oříznutím na více n-úhelníků, viz obr.2.10.1.



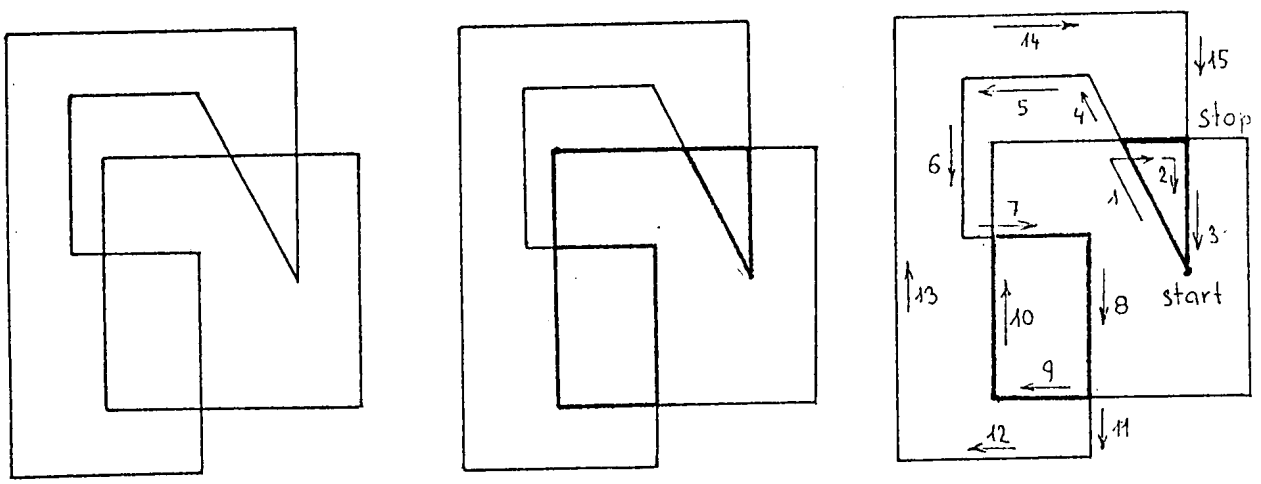
$\mathbf{v} = \mathbf{a} \times \mathbf{c}$ $v_z > 0$ bod P_4 je vně

$\mathbf{v} = \mathbf{a} \times \mathbf{b}$ $v_z < 0$ bod P_3 je uvnitř

Obr.2.10.4

2.11. Weiler-Athertonův algoritmus

Výše uvedený S-H algoritmus vyžaduje, aby n-úhelník, který tvoří oblast oříznutí, byl konvexní. V mnoha případech je takový předpoklad téměř nesplnitelný. Je pochopitelné možné nekonvexní n-úhelník rozdělit na několik konvexních n-úhelníků, provést oříznutí S-H algoritmem a pak je opět složit dohromady. Uvažme jednoduchý případ, viz obr.2.11.1.a.



ořezávaný n-úhelník	oblast oříznutí	výsledek oříznutí S-H algoritmem
a)		b)
		postup a výsledek oříznutí
		c)

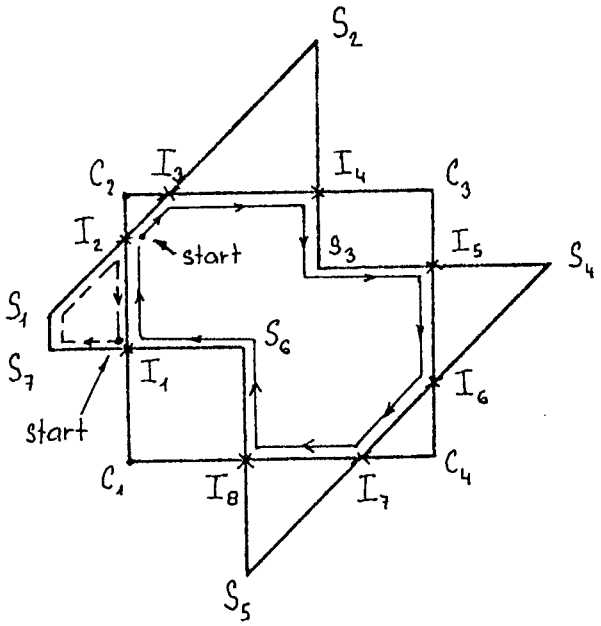
obr.2.11.1

Výsledek po oříznutí S-H algoritmem je na obr.2.11.1.b. Z obrázku je zřejmé, že kromě očekávaných částí ořezávaného n-úhelníka jsou ve výsledném tvaru ještě obsaženy části některých hran n-úhelníka ořezávacího.

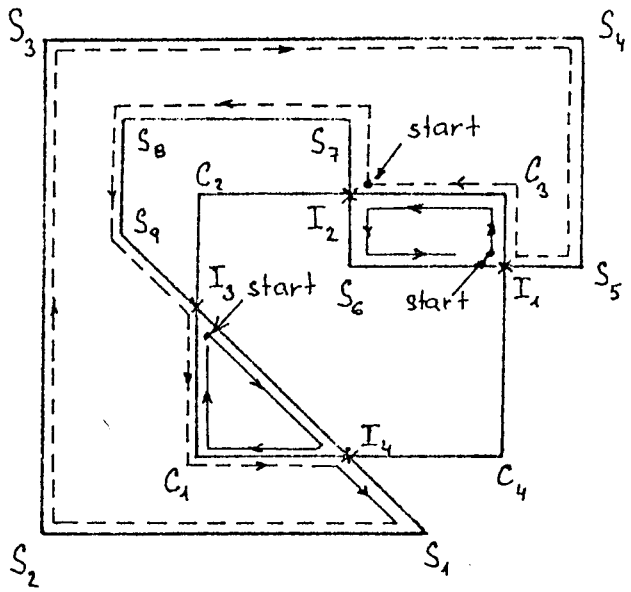
Weiler-Athertonův algoritmus (dále jen W-A algoritmus) umožňuje oříznutí obecně nekonvexního n-úhelníka oblastí, která je opět nekonvexním n-úhelníkem. Oba n-úhelníky mohou obsahovat

i vnitřní díry. Výsledek oříznutí W-A algoritmem je na obr.2.11.1.c. Z obrázku vyplývá, že výsledek může obsahovat několik samostatných n-úhelníků.

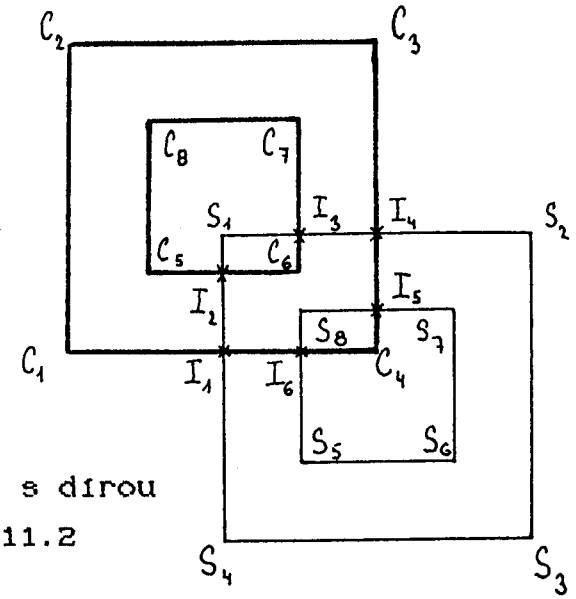
W-A algoritmus vyžaduje, aby n-úhelníky byly zadány pomocí seznamu vrcholů ve směru hodinových ručiček, přičemž vrcholy děr je nutné zadat ve směru opačném, tj. předpokládá se, že vnitřek n-úhelníka je vždy vpravo od orientované hrany. Pro názornost uvažme tři jednoduché charakteristické možnosti, viz obr.2.11.2.



Jednoduchý nekonvexní n-úhelník



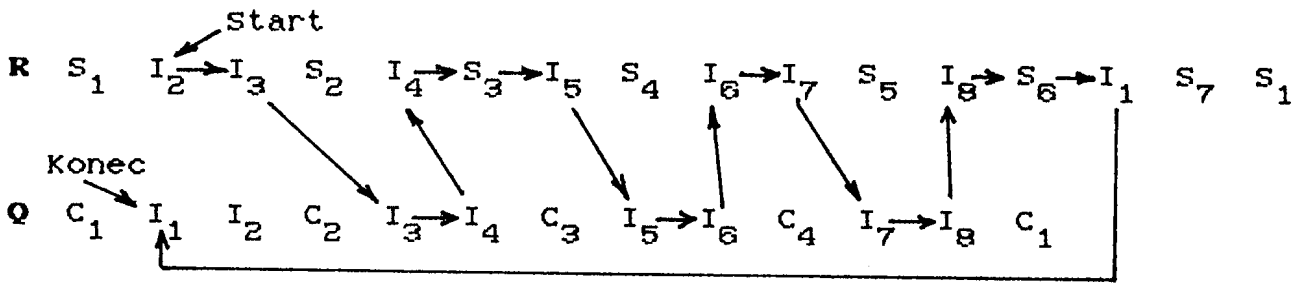
n-úhelník obklopující ořezávací n-úhelník



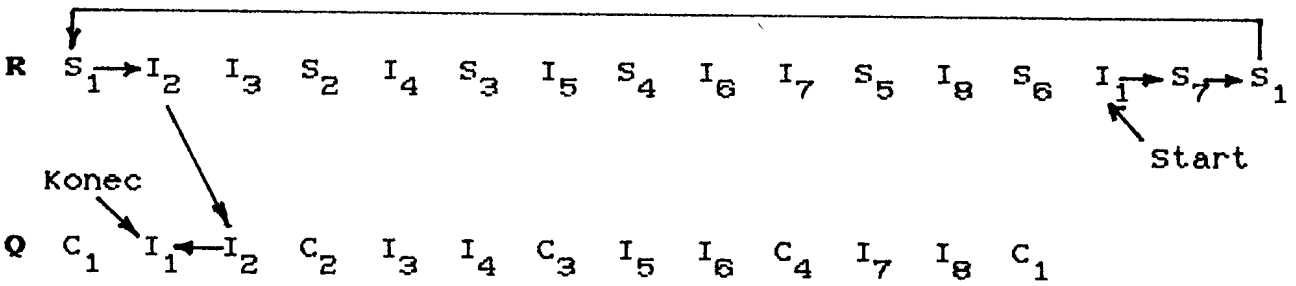
n-úhelník s dírou

Obr.2.11.2

Uvažme nejdříve případ z obr.2.11.2.a. Průsečíky jednotlivých hran n-úhelníků označené symbolem I_k jsou zařazeny do seznamu vrcholů jednotlivých n-úhelníků tak, abychom neporušili jejich orientaci, přičemž je nutné vytvořit vzájemné odkazy jednotlivých průsečíků I_k v seznamu vrcholů. Označíme-li **R** uspořádaný seznam vrcholů S ořezávaného n-úhelníka a průsečíků I_k, **Q** uspořádaný seznam vrcholů C n-úhelníka tvořícího oblast ořezávání a průsečíků I_k, pak dostáváme dva seznamy s příslušnými odkazy, viz obr.2.11.3.a.



a) Výsledný n-úhelník



b) Jedna z odříznutých částí původního n-úhelníka

Obr.2.11.3

Zároveň je nutné vytvořit seznam vstupních průsečíků I_2, I_4, I_6 a I_8, tj. bodů, kde hrana ořezávaného n-úhelníka vstupuje do n-úhelníka definujícího oblast ořezávání, a seznam výstupních průsečíků I_1, I_3, I_5 a I_7, tj. bodů, kde hrana ořezávaného n-úhelníka vystupuje z n-úhelníka definujícího oblast ořezávání. K vytvoření n-úhelníka, který vznikne oříznutím, je nutné začít od průsečíku, který je ve vstupním seznamu, a procházet seznamy **R** a **Q**, jak je naznačeno na obr.2.11.3.a. Výsledkem je seznam vrcholů:

I_2 I_3 I_4 S_3 I_5 I_6 I_7 I_8 S_6 I_1 I_2

V případě, že proces je započat od jiného vstupního průsečíku, je výsledek stejný (vrcholy jsou pouze cyklicky zaměněny). Průsečíky I_2, I_4, I_6 a I_8 musejí být odstraněny ze seznamu vstupních průsečíků, tj. seznam je nyní prázdný.

K vytvoření n-úhelníků zbylých po ořezávání je nutné začít od průsečíků z výstupního seznamu, tj. např. od průsečíku I_1 s tím, že seznam **Q** je prohlížen v opačném směru, viz obr.2.11.3.b. Výsledkem je pak seznam

I_1 S_7 S_1 I_2 I_1

přičemž průsečík I_1 musí být odstraněn z výstupního seznamu. Začneme-li od I_3, I_5 nebo I_7 dostáváme seznamy

I_3 S_2 I_4 I_3 $\quad$ I_5 S_4 I_6 I_5 $\quad$ I_7 S_5 I_8 I_7

přičemž je opět nutné odstranit příslušné průsečíky z výstupního seznamu.

Poněkud složitější situace nastává v případě, který je uveden na obr.2.11.2.b, kdy výsledkem oříznutí n-úhelníka jsou dva výsledné n-úhelníky. Postup je stejný jako v předešlém případě, viz obr.2.11.4.a, s tím, že po vytvoření prvního seznamu

I_1 S_6 I_2 C_3 I_1

a odstranění průsečíku I_1 ze vstupního seznamu je nutné ještě vytvořit seznam odpovídající vstupnímu bodu I_3

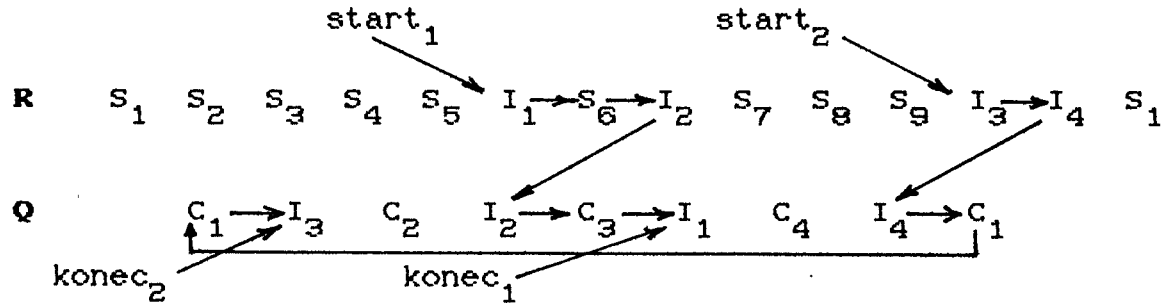
I_3 I_4 C_1 I_3

a vypuštění průsečíku I_3 ze vstupního seznamu.

Obdobným způsobem je vytvořen n-úhelník, který je zbylou částí původního n-úhelníka, s tím, že je nutné začít od libovolného prvku výstupního seznamu a seznam **Q** je prohlížen v opačném směru. Výsledkem je pak seznam:

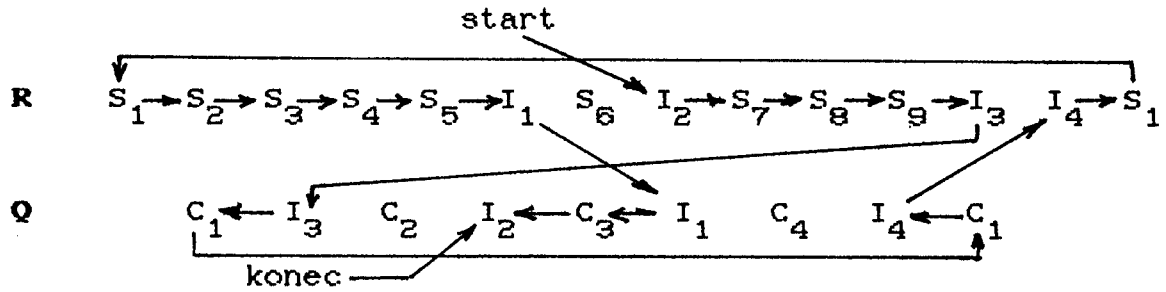
I_2 S_7 S_8 S_9 I_3 C_1 I_4 S_1 S_2 S_3 S_4 S_5 I_1 C_3 I_2

přičemž je nutné odstranit prvky I_2 a I_4 z výstupního seznamu, tj. seznam je nyní prázdný.



vstupní seznam: I_1, I_3 výstupní seznam: I_2, I_4

a) výsledné n-úhelníky

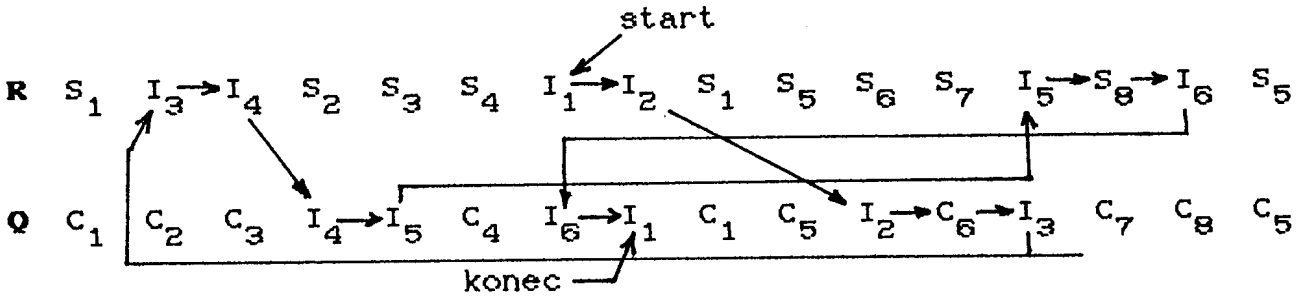


vstupní seznam: I_1, I_3 výstupní seznam: I_2, I_4

b) zbylá část

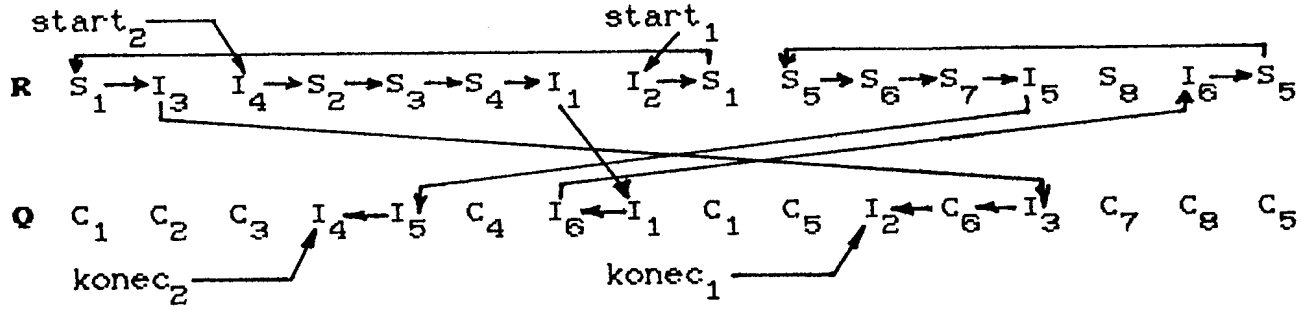
Obr.2.11.4

Uvažme nyní poslední případ z obr.2.11.2, kdy ořezávaný n-úhelník obsahuje díru a oblast ořezávání je tvořena též n-úhelníkem s dírou. Seznamy vytvoříme podobným způsobem jako v předchozích případech, viz obr.2.11.5.



vstupní seznam : I_1, I_3 , I_5 výstupní seznam: I_2, I_4, I_6

a) výsledný n-úhelník



vstupní seznam : I_1, I_3, I_5 výstupní seznam: I_2, I_4, I_6

b) zbylé části

obr.2.11.5

Postup je opět obdobný předchozím případům s tím rozdílem, že pro každý n-úhelník, který tvoří vnitřní či vnější hranici, je nutné vytvořit kruhový seznam. Je-li průsečík I_1 vzat jako první bod, pak výsledný seznam definující výsledek oříznutí je

$$I_1 \quad I_2 \quad C_6 \quad I_3 \quad I_4 \quad I_5 \quad S_8 \quad I_6 \quad I_1$$

a zbytek po oříznutí je definován seznamy

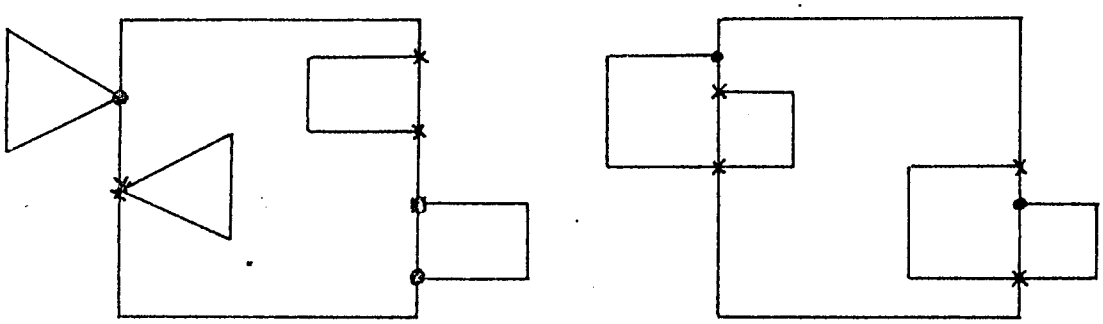
$$I_2 \quad S_1 \quad I_3 \quad C_6 \quad I_2$$

a

$$I_4 \quad S_2 \quad S_3 \quad S_4 \quad I_1 \quad I_6 \quad S_5 \quad S_6 \quad S_7 \quad I_5 \quad I_4$$

Je pochopitelné, že v případě úplné specifikace W-A algoritmu je nutné též řešit speciální případy, kdy se zadané n-úhelníky dotknou buď vrcholem, nebo hranami, viz obr.2.11.6. Do seznamů se pak zařadí jen ty průsečíky, které jsou označené *, zatímco

průsečíky označené • se nezařadí, podobně jako v kap.2.7.



Obr.2.11.6

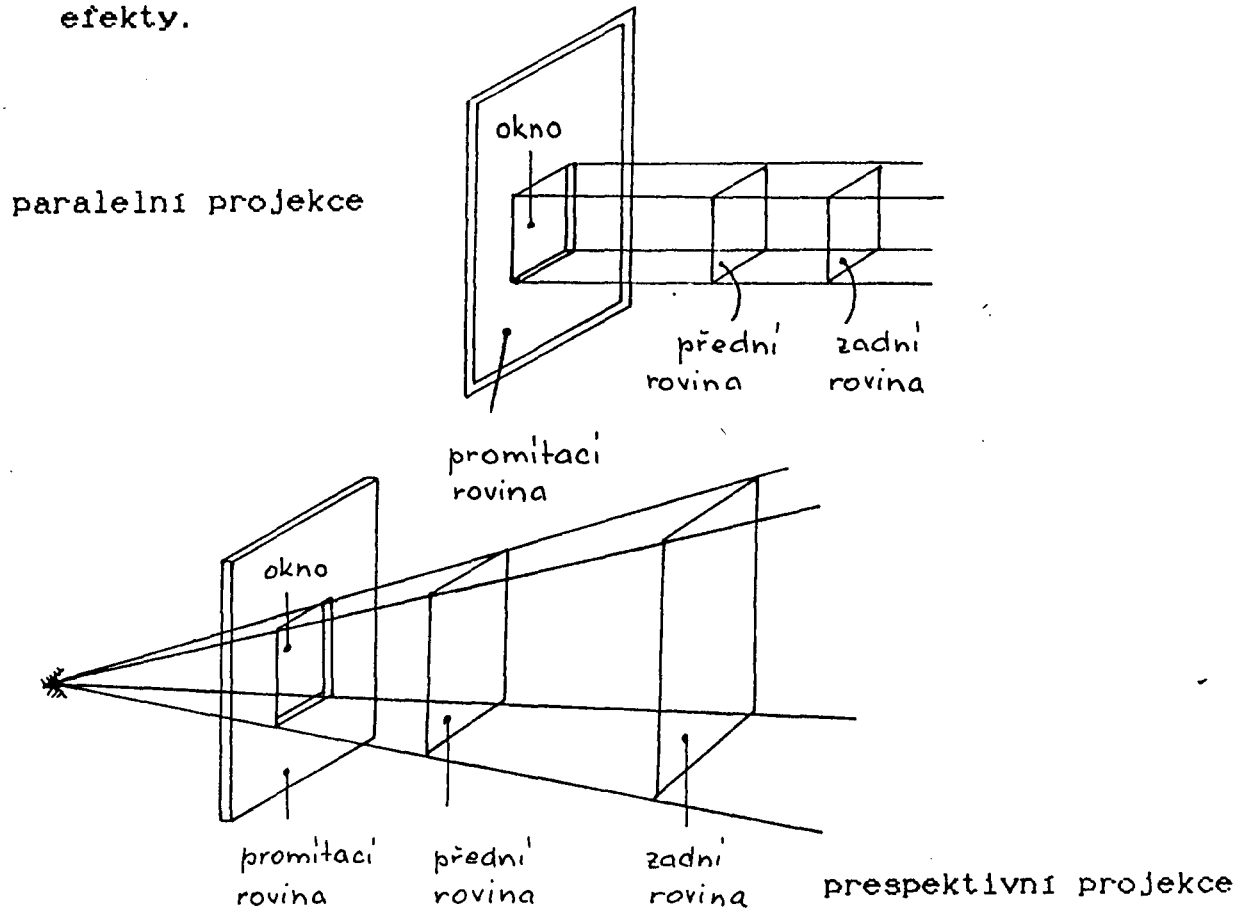
Nevýhodou uvedených algoritmů je nutnost orientace hran n-úhelníků, což může znamenat částečné omezení v praktických aplikacích. Orientaci je pak možné určit tak, že se např. vyhledá vrchol, který je nejvíce vpravo a podle hodnoty souřadnice y následujícího vrcholu se určí orientace n-úhelníka. V případě, že n-úhelník je konvexní, lze rozhodnout o orientaci na základě vektorového součinu směrových vektorů dvou po sobě jdoucích hran, které neleží na stejné přímce.

Jistou výhodou W-A algoritmu je, že jej lze modifikovat i pro případ použití kruhových oblouků, případně i jiných kvadratických křivek.

2.12. Orezávání v trírozměrném prostoru

Dosud uvedené algoritmy řešily problematiku orezávání úseček či n-úhelníka obecné vůči rovinné orezávací oblasti. Při zobrazování trírozměrných objektů je nutné používat modifikované orezávání v prostoru z několika důvodů, a to:

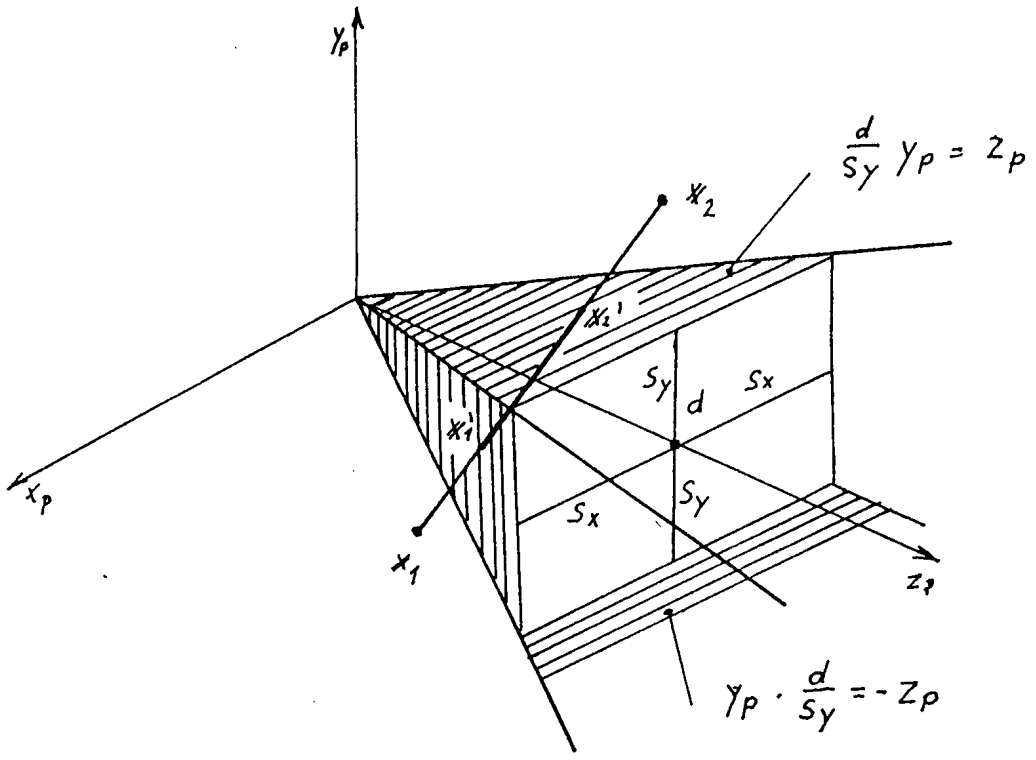
- objekt by mohl přesáhnout plochu, na kterou jej chceme zobrazovat (podobně jako při orezávání v rovině),
- objekty či jejich části, které jsou mimo zobrazovanou oblast tzv. pyramidu pohledu, by při projekci vytvářely nežádoucí efekty.



Obr.2.12.1

Z těchto důvodů bylo zavedeno orezávání vůči prostorové oblasti, která v případě používání perspektivní projekce má tvar pyramidy, viz obr.2.12.1.b. Pyramida pak reprezentuje viditelnou oblast pro pozorovatele. Obvykle je přední rovina orezávání oblasti v pozici pozorovatele a zadní rovina pak nekonečně daleko. Vzhledem k tomu, že okno může být různě velké, viz

obr.2.12.2, je vhodné nejdříve transformovat souřadnice tak, že ořezávání probíhá v "jednotkové" pyramidě, a poté aplikovat inverzní transformaci souřadnic.



obr.2.12.2

Jednotková pyramida je pak dána polorovinami:

$z \geq x$ $z \leq -x$ $z \geq y$ $z \leq -y$ $z \geq 0$

Celý proces ořezávání pak může být vyjádřen pomocí transformace

$$\mathbf{x'} = \mathbf{U}^{-1} \cdot \mathbf{Clip} \cdot \mathbf{U} \cdot \mathbf{x}$$

přičemž operace změny měřítka je reprezentována maticí **U** tak, že

$$\mathbf{U} = \begin{bmatrix} z_r/a & 0 & 0 & 0 \\ 0 & z_r/b & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

kde z_r určuje polohu promítací roviny na ose z,
a, b jsou pak rozměry okna.

Operátor **Clip** je pak popsán algoritmem 2.12.3 a je vlastně modifikací Cohen-Sutherlandova algoritmu, viz kap.2.2. Bity kódovací oblasti mají následující význam:

bit 0 - bod je vlevo od pyramidy, tj. x < -z
bit 1 - bod je vpravo od pyramidy, tj. x > z
bit 2 - bod je pod pyramidou, tj. y < -z
bit 3 - bod je nad pyramidou, tj. y > z

Je zřejmé, že na rozdíl od algoritmu z kap.2.2 musí být též určena odpovídající hodnota souřadnic y a z.

```
procedure CLIP3D ( x1, y1, x2, y2, z1, z2: real );
label 99;
var x, y, z, t: real;
    c, c1,c2: integer;

procedure CODE3 ( x, y, z: real; var c: integer );
begin c := 0; { hodnoty 1,2,4,8 odpovídají příslušným kódům }
      if x < -z then c:=1 else if x > z then c:=2;
      if y < -z then c:=c+4 else if y > z then c:=c+8;
end { CODE3 };
begin { zjištění stavových slov }
  CODE3(x1,y1,z1,c1);  CODE3(x2,y2,z2,c2);
  if (c1 land c2) = 0 then
  begin { úsečka může procházet pyramidou }
    if (c1 lor c2) <> 0 then
    begin { úsečka neleží celá v pyramidě }
      repeat
        if c1 = 0 then c:=c2 else c:=c1;
        if (c land 1) <> 0 then
        begin { bod je vlevo }
```

```
          t:=(z1+x1)/((x1-x2)-(z2-z1));
          z:=t*(z2-z1)+z1;  x:=-z;   y:=t*(y2-y1)+y1
        end
        else if (c land 2) <> 0 then
             begin { bod je vpravo }
               t:=(z1-x1)/((x2-x1)-(z2-z1));
               z:=t*(z2-z1)+z1;  x:=z;  y:=t*(y2-y1)+y1
             end
             else if (c land 4) <> 0 then
                  begin { bod je pod }
                    t:=(z1+y1)/((y1-y2)-(z2-z1));
                    z:=t*(z2-z1)+z1;
                    x:=t*(x2-x1)+x1;  y:=-z
                  end
                  else if (c land 8) <> 0 then
                       begin { bod je nad }
                         t:=(z1-y1)/((y2-y1)-(z2-z1));
                         z:=t*(z2-z1)+z1;
                         x:=t*(x2-x1)+z1;  y:=z
                       end;
        if c=c1 then
          begin x1:=x; y1:=y; z1:=z; CODE3(x1,y1,z1,c1)
          end
        else
          begin x2:=x; y2:=y; z2:=z;
                CODE3(x2,y2,z2,c2)
          end;
        if (c1 land c2)<>0 then goto 99 { usecka lezi vne }
      until (c1 lor c2)
    end;
    { nyni je nutne usecku zobrazit }
    LINE 3D(x1,y1,z1,x2,y2,z2)
  end;
99:
end { CLIP3D } ;
```

Algoritmus 2.12.1

3. Závěr

Je mi milou povinností poděkovat všem, kteří přispěli ke vzniku skript Počítačová grafika I a II, ať už vytvořením podmínek (zapůjčením počítačů k ověřování algoritmů) či stimulací k vlastní práci. Chtěl bych zejména poděkovat ing.I.Kolingerové za její konkrétní a podnětné připomínky, bývalým i současným studentům zaměření Počítačová grafika, kteří byli neocenitelnými pomocníky při ověřování algoritmů a tvorbě demonstračních programů. Bez jejich pomoci by nemohla ani vzniknout tato předkládaná disertační práce.

V předkládané práci jsem se pokusil zachytit rozvoj algoritmů počítačové grafiky v oblasti ořezávání. Pro další studium 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.+

b) sborníky konferencí

EUROGRAPHICS$^{\alpha}$
SIGGRAPH$^{\alpha}$ (USA)
Computer Graphics International$^{\alpha}$

Publikace označené *, + je možné vypůjčit přes knihovnu Západočeské univerzity (publikace označené + netvoří celý komplet), publikace označené α bude pravděpodobně možné si vypůjčit v krátké budoucnosti.

Za základní publikace 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.

4. 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.

[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.

[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 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.

[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 vypocetni technice, Matematicky seminar 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.

5. Publikační činnost (pouze od r. 1985)

Skala,V.: An Interesting Modification to the Bresenham Algorithm, NATO ASI 1985, in Fundamental Algorithms for Computer Graphics (Ed. R.A.E.Earnshaw), Springer Verlag, 1985.

Skala,V.: Hidden-Line, Hidden-Surface, Hidden-Contouring Problem. Problem Solutions by the Modified Bresenham's algorithm, Electronic Displays'87, London, 1987.

Skala,V.: Hidden-Line, Hidden-Surface and Hidden-Contouring Problem Solutions by the modified Bresenham's Algorithm, NATO ASI, 1987.

Skala,V.: An Intersecting Modification to the Bresenham Algorithm, Computer Graphics Forum, Vol.6., No.4, 1987.

Skala,V.,Tesar,R.,Zdráhal,K.: Device for the Power Factor Measurement, CS Patent No.239283, 1987.

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.

Skala,V.: Algorithms for 2D Line Clipping, ACM-SIGGRAPH Workshop Lisboa Local Group, Lisboa, Portugal, 1988.

Skala,V.: An Interesting Modification to the Bresenham Algorithm, ACM-SIGGRAPH Workshop Lisboa Local Group, Lisboa, Portugal, 1988.

Skala,V.: Basic Graphics Instructions for Drawing Surfaces with the Visibility Solution, Microsystem'88 International Conference, Bratislava (in Czech), 1988.

Skala,V.: Hidden-Line, Hidden-Surface, Hidden-Cortouring Problem Solutions by the Modified Bresenham's Algorithm, International Symposium on Algorithms'89, The House of Technology, Bratislava, 1989.

Skala,V.: Algorithms for 2D Line Clipping, in New Advances in Computer Graphics (Ed.R.A.Earnshaw,B.Wyvill) Proceedings of Computer Graphics International'89, Springer Verlag, 1989.

Skala,V.: A Unifying Approach to the Line Clipping Problem Solution, BISYCP'89, Beijing, 1989.

Skala,V.: Hidden-Line, Hidden-Surface Problem Solutions by the modified Bresenham's Algorithm, 4th Conference on Computer Graphics'89, Smolenice, Czechoslovakia.

Skala,V.: Algorithms for 2D Line Clipping, in EUROGRAPHICS'89 Proceedings (Ed.W.Hansmann,F.R.A.Hopgood,W.Strasser), North Holland, 1989.

Skala,V.: A Unifying Approach to the Visibility Solution, Iranian Journal of Science and Technology, accepted for publication, 1989.

Skala,V.: A Unifying Approach to the Line Clipping Problem Solution, SIAM Conference on Geometric Design, November 1989, TEMPE, AZ, USA.

Skala,V.: General Conics Clipping - Problem Solution, YUGRAPH'90, June 20-22, 1990, Dubrovnik, Yugoslavia.

Skala,V.: Algorithms for Clipping Quadratic Arcs, Computer Graphics International'90, June 27-29, 1990, Singapore.

6. Obsah