~~[illegible]~~ konference Algoritmy '91, JSMaF při SAV, 15.-19.4.1991, Štrbské Pleso, 1991

Algoritmy zobecněného ořezávání

Václav Skala
Katedra informatiky a výpočetní techniky, VŠSE, Plzeň

1. Úvod

Velmi důležitou částí každého programového vybavení pro počítačovou grafiku je ořezávání těch částí scény, které jsou mimo zobrazovanou oblast. Nejjednodušším případem je ořezávání v dvourozměrném prostoru vůči obdélníku zobrazovací plochy. Tento případ je v praxi nejčastější, zejména při zobrazování plošných objektů. Nicméně v technické praxi jsou zapotřebí i operace ořezávání nekonvexním n-úhelníkem či oblastí, tvořenou kruhovými oblouky.

2. Algoritmy zobecněného ořezávání

Algoritmy pro ořezávání nekonvexními n-úhelníky byly publikovány v dostupné literatuře [11], avšak uspokojivě neřešily problematiku singulárních případů, když ořezávaná úsečka či přímka prochází vrcholem nebo se jej dotýká nebo když hrana n-úhelníka leží na ořezávané přímce. S rozšířením aplikací počítačové grafiky vyvstává stále častěji požadavek na ořezávání úseček oblastmi, jejichž hranice jsou tvořeny nejen úsečkami, ale i kruhovými oblouky nebo částmi jiných kuželoseček. Oblasti samotné pak mohou obsahovat i díry. Předkládané algoritmy pro ořezávání nekonvexním n-úhelníkem či oblastí jsou založeny na parametrickém vyjádření úseček.

Předpokládejme, že hrany nekonvexního n-úhelníka jsou opět dány ve směru anebo proti směru hodinových ručiček, a že se vzájemně neprotínají. Pro jednoduchost předpokládejme, že pouze hrany sousední mají společný bod, sousední hrany neleží na společné přímce a vrcholy jsou navzájem různé. Uvedený algoritmus lze modifikovat i pro n-úhelníky nesplňující výše uvedené předpoklady.

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

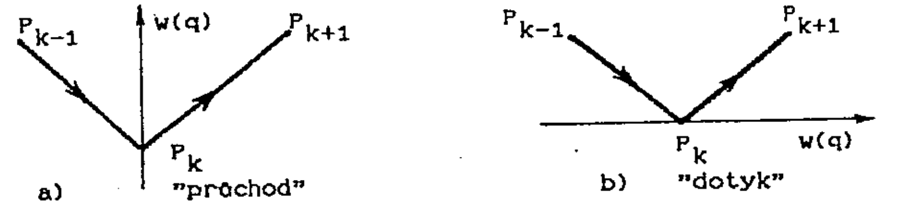
$$x(q) = x_r + (x_s - x_r) \,.\, q \qquad q \in (-\infty , \infty)$$

a souřadnice bodů $x(p)$ každé hrany n-úhelníka jako

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

Budeme zatím hledat průsečíky hran n-úhelníka s přímkou $v(q)$, na které leží ořezávaná úsečka P_rP_s. Kromě průsečíku přímky s hranou musíme rozlišit případy, kdy hrana n-úhelníka leží na přímce $v(q)$ a kdy přímka $v(q)$ prochází vrcholem. Při průchodu přímky $v(q)$ vrcholem mohou nastat dva případy, viz obr.1. V případě ad a) se generuje pouze jeden průsečík, zatímco v případě ad b) se generují dva totožné průsečíky. 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 $v(q)$.

$[\, s_1 \times s_2 \,]_z \cdot [\, s_3 \times s_2 \,]_z > 0$ $\qquad$ $[\, s_1 \times s_2 \,]_z \cdot [\, s_3 \times s_2 \,]_z < 0$



kde + , resp − v indexech značí součet, resp. rozdíl modulo n

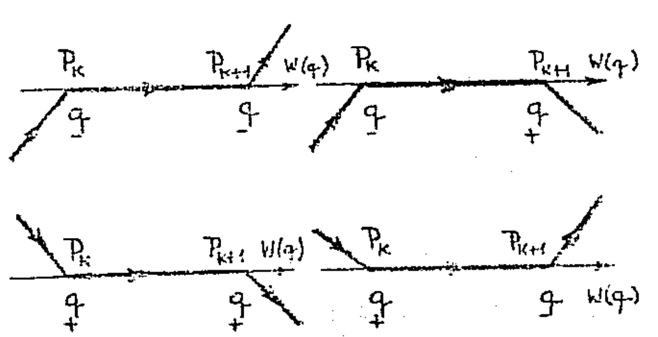
a $\quad s_1 = x_k - x_{k-1} \qquad s_2 = x_s - x_r \qquad s_3 = x_{k+1} - x_k$

Obr.1

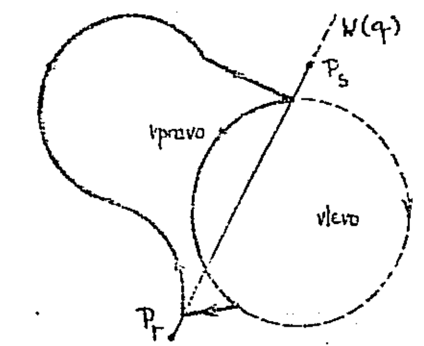
V případě, že na přímce $v(q)$ leží některá hrana, mohou nastat situace, které jsou znázorněny na obr.2. V těchto případech není možné ihned rozhodnout, jaké hodnoty parametru q mají být generovány. Z tohoto důvodu musí být generován speciální **atribut** parametru q, který je určen znaménkem souřadnice z vektorového součinu vektorů s_1 a s_2, resp. s_3 a s_2. Průsečík bude určen nejen hodnotou q, ale též hodnotou atributu, který je dán jako:

- _ (mezera) pro průsečík s hranou
- \+ nebo − znaménko 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 spolu s atributy. V následujícím kroku je nutné provést redukci získaných hodnot q podle tab.1; úplná tabulka včetně případů pro obecnější předpoklady viz [6]. Výsledkem je pak množina dvojic hodnot q, které určují úseky **přímky** $v(q)$ ležící uvnitř n-úhelníka. Pro určení úseků **úsečky** $P_r P_s$, 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 označenými částmi algoritmu 2.



Obr.2



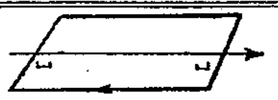
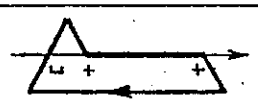
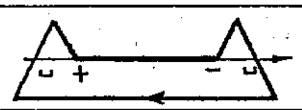
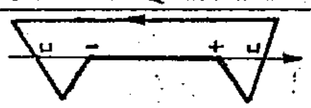
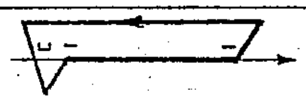
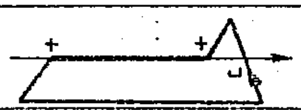
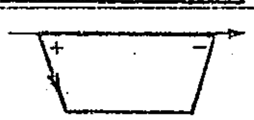
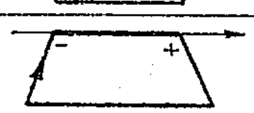
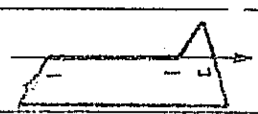
Obr.3

Příkaz pro výběr podintervalů může být realizován např. algoritmem 1.

```
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 1

Až dosud v literatuře publikované algoritmy umožňovaly ořezávání 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 ořezávat úsečky vůči oblasti s hranicemi tvořenými oblouky.

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

Tabulka 1

Předpokládejme nyní, 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 ke spojnici počátečního bodu kruhového oblouku a jeho středu) má být vzata v úvahu. Pro zjednodušení algoritmu uvažme omezení, že všechny vrcholy mají navzájem různé souřadnice, žádný vrchol neleží na 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 předchozího algoritmu může mít přímka $w(q)$ s kruhovým obloukem dva průsečíky, což poněkud komplikuje řešení problému. V případě kruhového oblouku musíme totiž řešit následující soustavu rovnic vzhledem k proměnné q

$$x(q) = x_r + (x_s - x_r) . q \qquad q \in (-\infty , +\infty)$$

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

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 $a = (x_s - x_r)^2 + (y_s - y_r)^2 \qquad c = (x_r - x_u)^2 + (y_r - y_u)^2 - r^2$

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

V případě, že přímka $w(q)$ danou kružnici protíná nebo se jí dotýká, obdržíme řešením rovnice obecně dva reálné kořeny.

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

- je-li oblouk orientován doprava, bude bod $x(q_i)$ uvažován tehdy a jen tehdy, je-li $[s_1 \times s_2]_z < 0 \quad i=1,2$
- je-li oblouk orientován doleva, bude bod $x(q_i)$ uvažován tehdy a jen tehdy, je-li $[s_1 \times s_2]_z > 0 \quad i=1,2$

přičemž $x_k \neq x(q_i), \quad s_1 = x_{k+1} - x_k , \quad s_2 = x(q_i) - x_k$

Je zřejmé, že opět musí být řešeny speciální případy, kdy např. přímka $w(q)$ prochází vrcholem x_k. V těchto případech budou vyhodnoceny vektory s_1, 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. $s_3 = x_k - x_{k-1}$

V tabulce 2 jsou uvedena pravidla pro vyhodnocování možných situací, podrobnější ilustrace viz [5], [6]. Je-li oblouk orientován doprava, pak musí být změněno znaménko souřadnice z příslušného vektorového součinu. Pak můžeme definovat hodnoty proměných a , b pomocí sekvencí:

$a:=[s_1 \times s_2]_z$; $b:=[s_3 \times s_2]_z$;
if $x_{k-1}x_k$ je oblouk **then if** $a = 0$ **then** $a := - s_1 . s_2$
else if orientace oblouku je doprava **then** $a := -a$;
if $x_{k-1}x_k$ je oblouk **then if** $b = 0$ **then** $b := s_3 . s_2$
else if orientace oblouku je doprava **then** $b := -b$;

Celý postup ořezávání úsečky nekonvexní oblastí je možné realizovat např. algoritmem 2.

$[s_1 \times s_2]_z$	$[s_3 \times s_2]_z$	typ dotyk/průchod
< 0	< 0	průchod
< 0	> 0	dotyk
> 0	> 0	průchod
> 0	< 0	dotyk
< 0	= 0	**if** $s_3 . s_2 > 0$ **then** průchod **else** dotyk
> 0	= 0	**if** $s_3 . s_2 > 0$ **then** dotyk **else** průchod
= 0	< 0	**if** $s_1 . s_2 > 0$ **then** dotyk **else** průchod
= 0	> 0	**if** $s_1 . s_2 > 0$ **then** průchod **else** dotyk
= 0	= 0	**if** $s_1 . s_2 > 0$ **xor** $s_3 . s_2 > 0$ **then** průchod **else** dotyk

Tabulka 2

```
procedure Comp ( x_A, x_B: vector; var r: real; t: boolean );
begin if x_A x_B je lineární
      then begin s := x_B - x_A;  r := [ s x s_2 ]_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 v(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 Výpočet hodnoty ( q ); { předpokládá se, že  x_k = x(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 prosečík je uvnitř <x_k,x_i) then Generuj(q  s atributem ␣ )
           end
```

```
        else begin Výpočet hodnot ( q1 , q2 );
                     if průsečík existuje then Generuj( q1*, q2* s atributem ␣ )
              (* značí, že průsečík leží na požadované straně oblouku xkx1 )
              end;
    k := i;  i := i + 1;
end ( while );
SORT( získané hodnoty q );  REDUKUJ ( hodnoty q podle tabulky );
VYBER ( podintervaly jako <qj,qj+1> ∩ <0,1>  ∀ j );
```

Algoritmus 2

3. Závěr

Pro skutečné použití kuželoseček jako grafických primitiv jsou nezbytné též algoritmy jejich ořezávání konvexními a nekonvexními oblastmi. Popis těchto operací lze nalézt např. v [9], resp. [10]. Tyto algoritmy ve spojení s Weiler-Athertonovým algoritmem ořezávání obecné nekonvexní oblasti oblastí též nekonvexní poskytují možnosti pro zavedení kuželoseček jako základních grafických primitiv. Na tomto místě je nutné zdůraznit, že aproximace kuželoseček lineárními úseky může vést k podstatnému snížení rychlosti prováděných grafických operací, zejména pak při ořezávání a geometrických transformacích.

4. Literatura

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

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

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

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

[5] Skala,V.: Algorithms for 2D Line Clipping, in [3], 1989, pp.355-367.

[6] Skala,V.: Počítačová grafika I, skripta VŠSE Plzeň, 1990

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

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

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

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

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