8. VIDITELNOST

Cíl Po prostudování této kapitoly budete umět

- určit viditelné a neviditelné hrany a stěny 3D objektů
-

Výklad

Odstraňování neviditelných hran patří k základním procesům 3D grafiky. Nepatří však k nejednodušším. Spíše naopak. Řada algoritmů, která se tímto problémem zabývá, je náročná jak na čas tak na prostor paměti, který je zapotřebí. Všechny algoritmy však vycházejí z **analýzy prostorového objektu.** (Jinak to snad ani nejde.) Pro jednoduchost budeme vycházet z předpokladu, že těleso (objekt) je "oplátován" rovinami.

Z rovnice roviny $\alpha : a\,x + b\,y + c\,z + d = 0$ bude pro jednotlivé body prostoru platit:

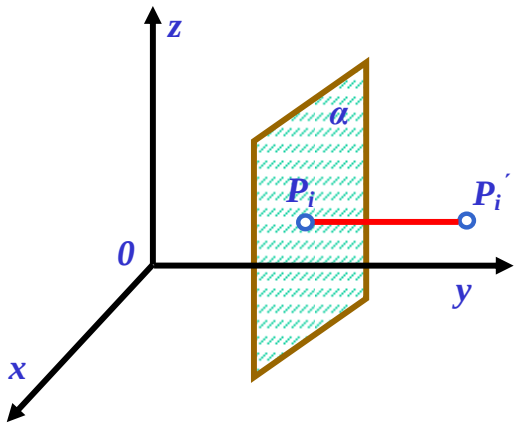
Je-li bod P_i (x_i, y_i, z_i) bodem roviny, potom platí $a\,x_i + b\,y_i + c\,z_i + d = 0 \ \ldots = q$.

Jestliže $q > 0$, leží bod "nad" rovinou α.

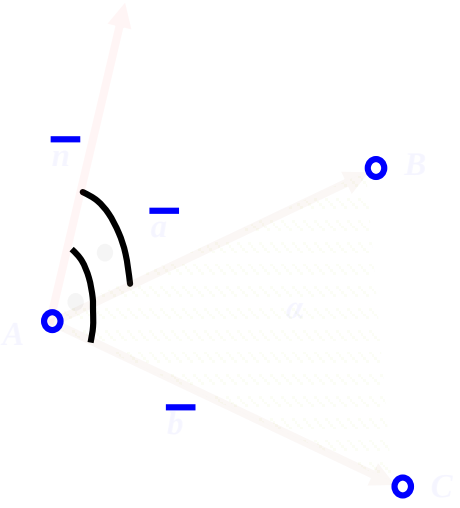
Je-li $q < 0$ leží bod P "pod" rovinou α. Koeficienty a, b, c jsou souřadnice normálového vektoru roviny $\alpha : a\,x_i + b\,y_i + c\,z_i + d = 0$.

Na obrázku 8.1 je v pravoúhlé souřadnicové soustavě naznačena rovina α a body P_i a P_i'.

Bod P_i je bodem roviny α a bod P_i' není bodem roviny α. V počítačové grafice však rovina je často zadána třemi body. Orientace roviny dané třemi body je určena normálou



Obr. 8.1



Obr. 8.2

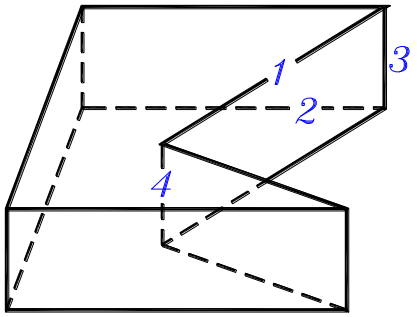
roviny. Souřadnice normálového vektoru $\overline{n}$ dostaneme kde vektorovým součinem vektorů $\overline{a}, \overline{b}$ této roviny.

Tento jednoduchý algoritmus není spolehlivý při konkávních - nevypuklých objektech. U těchto konkávních objektů existuje rovina, která "dělí" objekt. Tím se stane, že bod *P* prostoru je "nad" rovinou objektu, ale "nad" tímto bodem se nachází jiný bod objektu, který jej "zastiňuje". Je nutno použít více metod ke zkoumání celé skupiny objektů. Jedna z metod je vyjít z obrysu pokud tento obrys dokážeme určit. Potom bereme jednu hranu za druhou a určujeme jejich viditelnost. Tuto viditelnost určíme z rozdílnosti vzdálenosti "průsečíků" zkoumaných hran ve směru pohledu. V tomto směru nám pomůže **analýza průmětu prostorového objektu.** Hrany rozdělíme na neviditelné a viditelné.

K získání obrysových hran lze použít vlastnost normál rovin stěn. Nejdříve určíme viditelnost jednotlivých stěn podle normál těchto stěn. Stěna, kde její normála svírá ostrý úhel se směrem promítacího paprsku je viditelná (přivrácená). Je-li úhel promítacího paprsku a normály tupý, stěna je neviditelná (odvrácená). Hrany objektů je možné řešit jako průsečnice dvou rovin. Průsečnice dvou viditelných stěn jsou viditelné hrany. Průsečnice dvou neviditelných stěn jsou neviditelné hrany. Průsečnice viditelné stěny s neviditelnou stěnou tvoří obrysovou hranu. I tento způsob však závisí na konvexnosti resp. konkávnosti objektu. Hrany, které padnou "dovnitř obrysu" je nutno rozhodnout zjištěním vzdálenosti zdánlivého průsečíku těchto hran. (Bude dále zmíněno v následující části studijního materiálu).

Postup tedy rozdělme do bodů (Obr. 8.3):

1. Hrana dvou odvrácených ploch je neviditelná. (Hrana 2).
2. Hrana dvou přivrácených ploch je potencionálně **viditelná.** (Hrana 1).
3. Hrana tvořená přivrácenou a odvrácenou stranou je neviditelná pokud úhel normál je ostrý (hrana 4). Je-li tento úhel tupý, jde o hranu obrysovou. (Hrana 3).



Obr. 8.3

Dále je nutno prozkoumat potencionálně viditelné hrany. Určíme průsečíky těchto průmětů hran (pokud existují) a určujeme viditelnost úseků hran nezkřížených jinými hranami. Metody jsou různé. Například bodem hrany vedeme promítací paprsek a zjišťujeme, zda neprotíná některou plochu. Jestliže protíná, je úsečka nebo její část zacloněna jinou

plochou. Zde se s výhodou dá použít zpětného paprsku "zdánlivého" průsečíku hran. Tímto se budeme zabývat později.

Tento postup, kdy se orientujeme na hrany nebo stěny těles resp. objekty lze obecně charakterizovat algoritmem:

for každý objekt **do**

begin

Urči části (hrany resp. stěny) objektu, které nejsou zakryté ostatními objekty

Nakresli tyto nezakryté části objektu jejich barvou

end.

Druhou skupinou metod určování viditelnosti jsou metody zaměřené na **obraz** scény. Tyto metody zpracovávají viditelnost vybarvením každého pixelu obrazovky příslušnou barvou. Algoritmy, které zpracovávají viditelnost objektů 3D scény v prostoru obrazovky se nazývají **obrazově orientované.**

Obrazově formulovaný algoritmus viditelnosti:

for každý pixel **do**

begin

Urči objekt, který je nejblíže pozorovateli ve směru pohledu vedeného každým pixelem

if paprsek vedený pixelem prochází objektem **then**

Obarvi pixel barvou příslušnou k objektu

else

pixel obarvi barvou pozadí

end.

Problémy obou metod jsou v náročnosti na zpracování. Jde o čas i paměťovou náročnost. Metoda zaměřená na objekty mívá slabá místa v logickém vyhodnocování prostorových vztahů a tím dochází k chybnému řešení viditelnosti. Druhý způsob - tedy metody obrazově orientované - jsou bezpečnější, ale časově náročnější. Zde nezáleží na složitosti a počtu objektů v prostoru, ale na počtu pixelů, pro které vyhodnocování probíhá. Chybné vyhodnocení vede k nepřesnému zobrazení části obrazu. Ale nezpůsobí celkově chybné vyhodnocení, jak se to může stát u objektově orientovaných metod.

K vylepšení řešení viditelnosti je využíváno různých typů ***koherence*** (spojitost, souvislost). Jde o skutečnost, že části scény (blízké objekty, hrany, stěny apod.) vykazují stejné nebo podobné vlastnosti, které zkoumáme při zobrazování.

Jde o: **a) *objektovou koherenci*** - množinu objektů lze rozdělit na dvě podmnožiny, které leží zcela v jednom nebo ve druhém poloprostoru. Potom objekty, které nejsou z poloprostoru pozorovatele, nezmění viditelnost.

b) *Stěnová koherence*. Povrch tělesa je v určitém rozsahu koherentní. Vlastnosti zkoumané stěny lze zjistit inkrementálním výpočtem na základě vyhodnocení sousedních stěn.

c) *Hranová koherence.* Hrany mění viditelnost pouze v místech, kde se v průmětech hran kříží nebo pokud protínají viditelný povrch - stěnu.

d) *Implikovaná hranová koherence.* Hrana, která vznikne jako spojnice dvou průsečíků, které určují průsečnici dvou planárních stěn.

e) *Řádková koherence.* Množina viditelných objektů určených v jedné zobrazovací řádce se příliš neliší od následující řádky.

f) *Plošná koherence.* Skupina sousedních pixelů obvykle zobrazuje stejný povrch. "Podmnožinou" je "*řádková*"- *úseková koherence*, kdy zkoumáme pixely v jedné řádce.

g) *Hloubková koherence.* Přilehlé - sousední - stěny leží obvykle v obdobné vzdálenosti od pozorovatele. Odlišná tělesa promítaná do stejného místa mají obvykle různou hloubku. (Obdoba zdánlivého průsečíku průmětu hran.)

h) *Obrazová - snímková - koherence.* Snímky pořízené v krátkém časovém úseku jsou obvykle velmi podobné. Mohou vykazovat malé změny jak v barvě, tak ve viditelnosti objektů.

Použití resp. nepoužití příslušné koherence může mít významný vliv jak na časovou náročnost algoritmu, tak při nevhodném použití na správnost algoritmu.

8.1. Obrazově orientované algoritmy viditelnosti

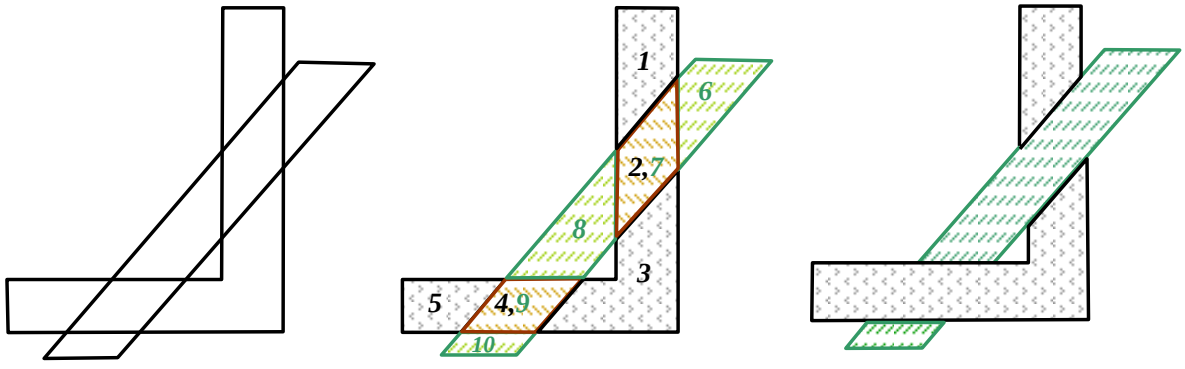
V této části se budeme zabývat algoritmy, které jsou nejčastěji používané. Jde o malířův algoritmus, Z-buffer, řádkový algoritmus a sledování paprsk. Pro přehlednost a možnost vzájemného porovnání stručně uvedeme dále algoritmy těchto metod.

8.1.1. Malířův algoritmus

Princip tohoto algoritmu spočívá v tom, že je jednotlivým útvarům, prvkům přidělena priorita kresby dle vzdálenosti od průmětny. Čím je prvek níže, tím má vyšší prioritu kresby. Je tedy kreslen nejdříve. Vlastní kresba potom probíhá tak, že jsou jednotlivé útvary překreslovány. Jde o techniku, kterou malíři používají. Ke zvýraznění "prostorovosti" kreslí nejprve pozadí obrazu a postupně na toto pozadí nanášejí další vrstvy a tím zlepšují prostorovost obrazu. Jistou modifikací tohoto způsobu je tzv. *obrácený malířův algoritmus.* Jeho princip spočívá v postupu, kdy kreslíme objekty odpředu dozadu tak, že kreslíme nejbližší části a z ostatních kreslíme postupně dle vzdálenosti jen ty, které neleží v již dříve kreslené části. Předpokladem obou způsobů je, že scéna je modelována hraniční reprezentací s rovinnými pláty, které se neprotínají.

Základním útvarem může být rovinná ploška, obvykle trojúhelník. Při zobrazování ploch se prostorové (křivočaré) čtyřúhelníky vytvořené na ploše parametrickými křivkami rozdělí úhlopříčkou na dva trojúhelníky. Podobně se soustavou trojúhelníků nahradí stěny zobrazovaných těles (hranol, jehlan, válec, kužel, apod.).

Důležitým předpokladem malířova algoritmu jsou třídící algoritmy, které jednoznačným způsobem určí pořadí kresby. Zde nastávají problémy v tom, že nelze vždy toto pořadí určit u kompletních objektů. V tom případě je nutno objekt rozložit na více objektů a třídění provést znovu. Na následujícím obrázku (Obr. 8.4 a)) je právě situace, kdy dva objekty se vzájemně zakrývají a nelze určit pořadí "hloubky" jednotlivých celých objektů.



a) b) c)

Obr. 8.4

Objekty je nutno rozdělit na vzájemně se nepřekrývající obrázky (Obr.8.4 b)). Tak vznikne celkem 10 plošek. Plošky 1,3,5 a 6,8,10 se nepřerývají a jsou vybarveny barvou příslušných

objektů. Plošky 2, 4 a 7, 9 se kryjí a je třeba rozhodnout, které jsou blíže pozorovateli a jsou tudíž vybarveny příslušnou barvou. Výsledek je na obrázku Obr. 8.4 c.

Algoritmus je možno rozdělit do bodů:

1.Porovnej jednotlivé plochy z hlediska jejich z-tových souřadnic. Plocha s menší z-tovou souřadnici bude kreslena první.

2. Jestliže se plochy nepřekrývají, potom na pořadí kresby nezáleží.

3. Jestliže se neprotínají, lze určit pořadí jejich kreseb.

4. Pokud se protínají, je nutno určit jejich průsek a zpětným paprskem určit průsečík (viz. předcházející metoda) a pomocí zdánlivého průsečíku určit pořadí kreslení ploch.

Algoritmus malířova algoritmu.

Vstup: seznam polygonů *{P1, P2, . . . , Pn}*

pole *Intenzita [x,y]* nastavit na intenzitu pozadí}

Setřiď polygony podle hloubky;

for každý polygon P od nejhlubšího k nejbližšímu **do**

for každý vnitřní pixel[*x*,*y*] průmětu P **do**

Intenzita [*x*,*y*] = intenzita P v bodě (*x*,*y*);

Zobraz pole Intenzita;

Algoritmus obráceného malířova algoritmu.

Vstup: seznam polygonů {P1,P2,...,Pn}

pole Maska[*x*,*y*] inicializované O

pole Intenzita[*x*,*y*]

Setřiď polygony podle hloubky;

for každý polygon P od nejhlubšího k nejbližšímu **do**

for každý vnitřní pixel[*x*,*y*] průmětu P **do**

if Maska[*x*,*y*] = 0 **then**

begin

Intenzita [*x*,*y*] = intenzita P v bodě(*x*,*y*);

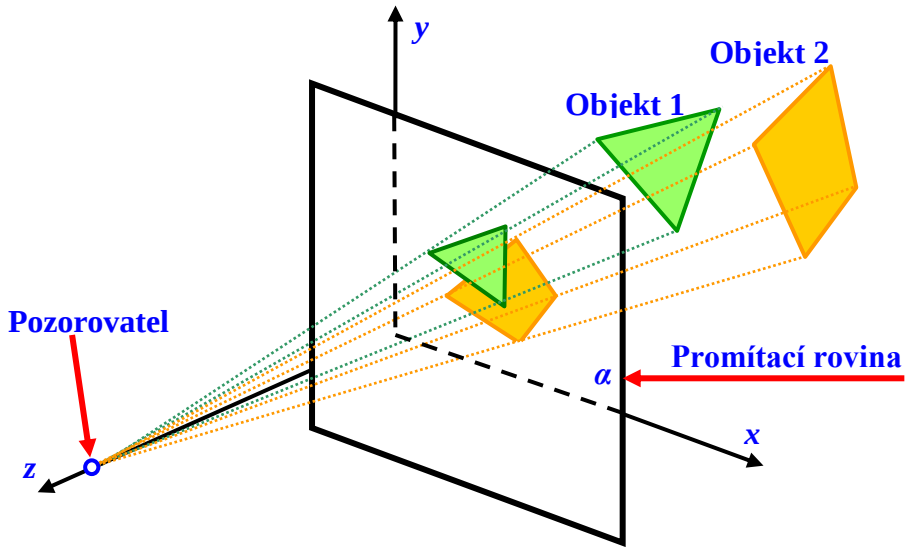
Maska[*x*,*y*] = 1

end;

Zobraz pole Intenzita;

8.1.2. Z-buffer

Jde o jeden z nejznámějších a nejbezpečnějších algoritmů. Ovšem za cenu rychlosti. Tak, jak do obrazové paměti ukládáme barvu (resp. jas) každého bodu - tedy pixelu, tak do paměti **hloubky** ukládáme jeho z-tovou souřadnici. Tedy z-tovou souřadnici průsečíku paprsku vedeného z tohoto zobrazovaného bodu do scény s promítací rovinou. Viz obrázek.
Výhodou tohoto algoritmu je jeho jednoduchost, která vlastně řeší všechny možné komplikace složitých tvarů, jejich překrývání dotyky a průsečíky. Výpočtová složitost je dána počtem zpracovaných n-úhelníků. Cenou je však **paměťová náročnost** a s tím související i **čas** potřebný ke zpracování.



Obr. 8.5

Na obrázku 8.5 je pozorovatel v bodě na ose z. Pokud je pozorovatel v nevlastním bodě, jsou promítací paprsky rovnoběžné a vzdáleností jednotlivých objektů od zobrazovací roviny jsou shodné s velikosti z-tové souřadnice zobrazovaného objektu. Pro středové promítání jsou ukládány hodnoty o barvě a intenzitě podle vzdáleností d_1, d_2,.. zobrazovaných objektů O_1, O_2, ... Je-li objektem O rovinná plocha, je možné využít pro výpočet barvy a intenzity vnitřních pixelů lineární interpolaci.
Označíme $I(x,y)$ intenzitu nebo barvu na pozici bodu (x_p, y_p) a $Z(x,y)$ vzdálenost nejbližšího bodu zpracované scény od pozorovatele, který se promítne do bodu (x_p, y_p).

Algoritmus můžeme popsat:

1. Nastav $I(x, y)$ $\forall x, y$ na barvu pozadí a $\forall$x, y nastav $Z(x, y)$ na největší možnou vzdálenost.
2. Pro všechny pixely, na které se zobrazí daný n-úhelník:
   a) vypočti z-tovou souřadnici resp. vzdálenost *d*,
   b) je-li hodnota $z < Z(x,y)$, potom nastav $Z(x,y)$ na hodnotu z a nastav pixel $I(x,y)$ na hodnotu jasu či barvy odpovídající bodu (x,y).

Paměťová náročnost:

r_x ... resp. r_y rozlišovací schopnost ve směrech osy x resp. y.

n_i ... počet bytů nutných k reprezentaci jedné hodnoty barvy či intenzity

n_r ... počet bytů nutných k reprezentaci jedné hodnoty v Z-bufferu.

Celková kapacita paměti je dána vztahem

$$M = r_x \,.\, r_y \,.\, (n_i + n_r)$$

Pro $r_x = r_y = 1024$, $n_i = 1$ a $n_r = 5$ dostaneme

$$M = 1024 \,.\, 1024. \,(1 + 5) = \text{ 6MB.}$$

Urychlení algoritmu lze dosáhnout vhodnou matematickou úpravou výpočetních algoritmu. Například využití $\Delta x = 1$ a úpravy výpočtu tak, aby druhá proměnná *y* byla konstantní. Nebo opačně. (Sloupcová resp. řádková výpočetní metoda.)

Při zavedení souřadného systému z obrázku Obr. 8.5 jde o vyjádření z-ové souřadnice pro každou hodnotu (x,y). Předpokládejme, že pro každý n-úhelník máme k dispozici rovnice odpovídající roviny ve tvaru:

$$ax + by + cz + d = 0.$$

Pro $c \neq 0$ lze psát:

$$z = - (ax + by + d) / c$$

Při řádkové konverzi, kdy hodnota *y* je pro jeden daný řádek konstantní, lze pro souřadnice prvního pixelu (x_1, y_1, z_1) určit souřadnici z takto:

$$z = - (a (x_1 + \Delta x) + by + d) / c \,.$$

Je-li $\Delta x = 1$, potom dostaneme:

$$z = z_1 - a/c.$$

Příklad 1.
Aplikujeme uvedený postup na kvadratickou plochu. Např. pro kulovou plochu platí:

$$(x - x_s)^2 + k^2 + (z - z_s)^2 - r^2 = 0,$$

kde $k = y - y_s$ je konstanta pro daný řádek.

Potom pro body x_0 a x_1 lze psát:

$$(x_0 - x_s)^2 + k^2 + (z_0 - z_s)^2 - r^2 = 0$$
$$(x_1 - x_s)^2 + k^2 + (z_1 - z_s)^2 - r^2 = 0$$

Je-li $\Delta x = 1$, tj. $x_1 - x_0 = 1$, potom po odečtení rovnic dostaneme, že $z_1^2 = z_0^2 + 2x_0 - 1$.

Pro krok n + 1 dostaneme: $z_{n+1}^2 = z_n^2 + 2x_n - 1$

Vhodnou metrikou se lze vyhnout funkci odmocniny.

Algoritmus Z-bufferu.

```
procedure Z-buffer;
var pz:integer;
begin
    for každý pixel obrazovky do
        begin
            Pixel(x,y).Barva:=Barva pozadí;
            Pixel(x,y).Hloubka := Maximální hloubka {- Z-buffer }
        end;
    for každý plát do
        for každý vnitřní pixel(x,y) promítnutého plátu do
            begin
                Nová hloubka := Hloubka plátu v místě pixelu(x,y);
                Nová barva:= Barva plátu v místě pixelu(x,y);
                if Pixel(x,y).Hloubka >= Nová hloubka then
                    begin
                        Pixel[x,y].Barva:=Nová barva;
                        Pixel[x,y].Hloubka:=Nová hloubka
                    end
            end
end;
```

8.1.3. Řádkový algoritmus viditelnosti polygonů

Řádkový (Watkinsův) algoritmus je založen na využívání řádkové a hloubkové koherence.

Algoritmus řádkového algoritmu.

Setřiď všechny hrany polygonů podle *y*-souřadnic.

Vytvoř seznam aktivních hran pro 1. řádku.

for každou řádku obrazu **do**

Urči segmenty polygonů promítnuté na dané řádce. Po setřídění koncových bodů segmentů v řádce podle *x*-souřadnic jsou určeny úseky mezi setříděnými body.

Každý segment polygonu rozděl na části odpovídajícím úsekům.

for všechny úseky na řádce **do**

begin

Podle hloubky urči část segmentu viditelnou v daném úseku.

if úseky různých polygonů se protínají **then**

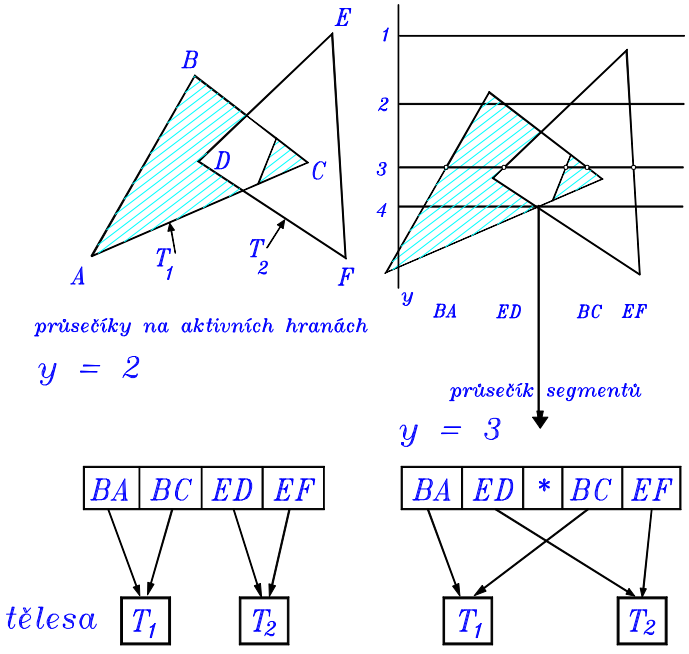
Urči jejich průsečík a rozhodni o vzájemné viditelnosti.

Vykresli úsek odpovídající barvou.

end;

Oprav seznam aktivních hran pro přechod na další řádku.

Na následujících obrázcích je příklad použití řádkového algoritmu při řešení viditelnosti.



Obr. 8.6

2.obrazová řádka ($y = 2$) protíná objektů T_1 a T_2 - trojúhelníky *ABC* a *EFG* po řadě na hranách *BA*, *BC*, *ED* a *EF*. Na této řádce se průměty trojúhelníků nepřekrývají a není tedy nutné porovnávat hloubku úseků. Viditelné úseky jsou *BA*-*BC*

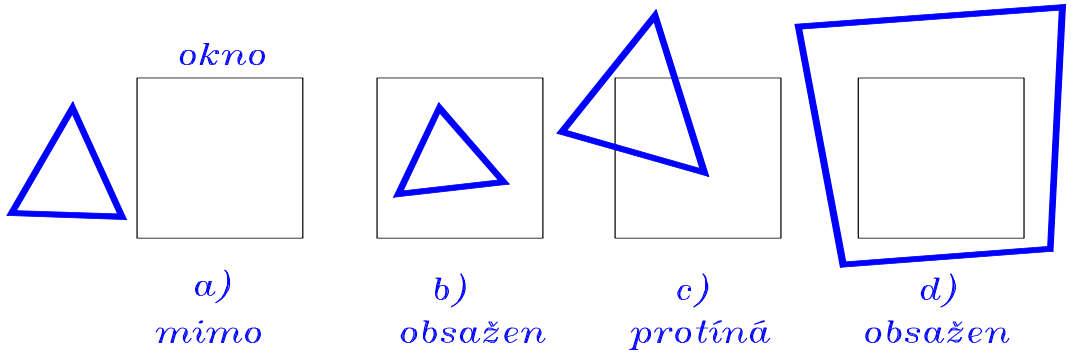
vybarveny barvou objektu T_1 a úseky *ED-EF* barvou objektu T_2. Na 3.řádce se segmenty dvou různých objektů překrývají. Je nutno řešit viditelnost v rozsahu překrytí *ED-BC*. Ve zbývajících úsecích 2.řádky (úseky *BA-ED* a *BC-EF*) jde o jeden objekt. Na překrytém úseku *(ED-BC)* je nutno určit hloubkovým testem průsečík, který je viditelný. Viditelnost úseků je *ED*-průsečík - barva T_1; *BC*-průsečík - barva T_2 .

8.1.4. Warnockův algoritmus

Při použití kombinovaných metod, na příklad malířova algoritmu, se setkáváme s tím, že pro přesnější analýzu je třeba rozdělit zobrazovanou scénu na více kreslicích ploch. Tento postup je v literatuře označován jako **Warnockův algoritmus.** Jeho princip spočívá v tom, že kreslící plochu dělíme tak dlouho, až je splněna jedna z následujících podmínek:

1. Všechny plošky leží mimo zónu - zůstane barva pozadí. Obr. 8.7 a).

2. Oblast obsahuje právě jeden celý n-úhelník. Daná oblast se vyplní barvou a zbytek - pozadím. (Obr. 8.7b)



Obr. 8.7

3. Oblast protíná právě jeden n-úhelník.

Daná část se vyplní barvou, zbytek pozadím. (Obr. 8.7c)

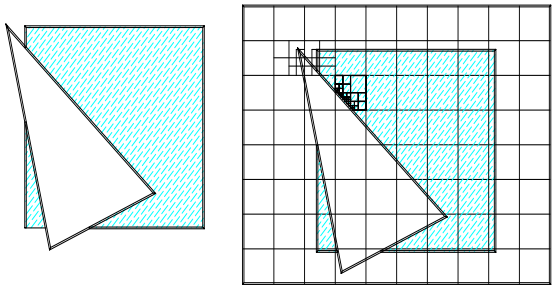
4. Pokud zobrazovaná část je celá uvnitř jednoho n-úhelníka. potom se celá oblast zobrazí barvou nejbližšího n-úhelníka, který oblast obklopuje. (Obr. 8.7d)

5. Pokud nenastane jeden z vyjmenovaných případů - oblast se rozdělí.

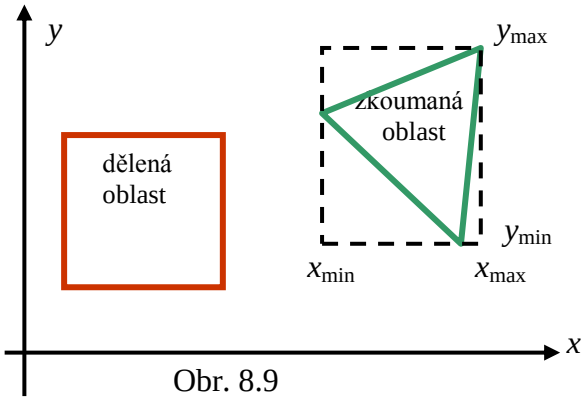
Dělení oblasti probíhá čtvercově resp. obdélníkově. Dle zadané oblasti.

Na následujícím obrázku Obr. 8.8 je naznačen postup při dělení obrazovky. Na tomto místě je možno připomenou záznam minima a maxima zobrazovaných objektů (n-úhelníků, trojúhelníků a pod.). Těchto hodnot je možno použít při realizaci této metody dělení

obrazovky. Viz obrázek Obr. 8.9. Jsou porovnávány minima a maxina x, y souřadnic dělené a zkoumané oblasti.



Obr. 8.8



Obr. 8.9

8.2. Plovoucí horizont

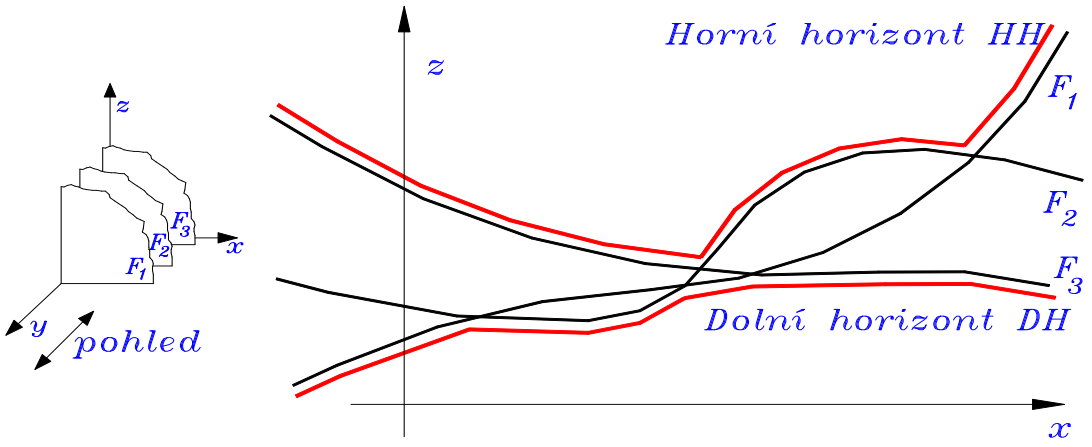
Pro vykreslování ploch, které jsou "popsány" jako "řezy" (řezných) křivek - což jsou řezy rovinami rovnoběžnými se souřadnicovými rovinami, lze použít pro zobrazení metodu tzv. **plovoucího horizontu.** Ploch zobrazujeme jako funkci dvou proměnných (nejčastěji vyjádřenou v explicitním tvaru)

$$z = F(x,y), \qquad \text{pro } x \in \langle x_a, y_b \rangle \quad \text{a} \quad y \in \langle y_c, y_d \rangle .$$

Řezy rovnoběžnými rovinami potom budou:

$$z_1 = F(x, y_j), \quad \text{a } z_2 = F(x_i, y) ,$$

kde i = 1, 2, . . ., n a j = 1, 2 , . . ., m .



Obr. 8.10

Nejdříve vykreslíme 2 řezy v jednom směru. Při kresbě třetí křivky - řezu vytvoříme **horizont** tak, že bude platit:

Horní Horizont := max (horní horizont, (x,y_j))

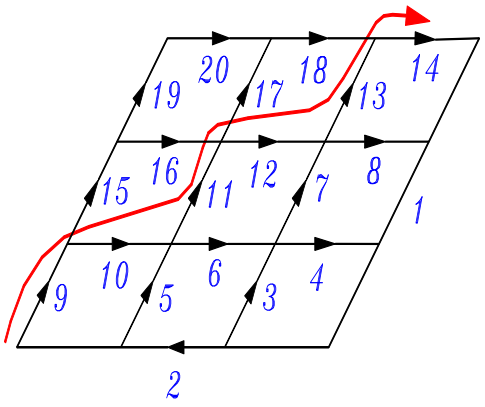
Dolní Horizont := min (dolní horizont, (x,y_j))

Vykreslování provedeme pomocí modifikovaného Bresenhamova algoritmu pro vykreslení křivky.

Postup:

1. Inicializujeme masku horizontu
   - nastavení Dolní hranice masky .....dolní okraj kreslící plochy;
   - nastavení Horní hranice masky .....horní okraj kreslící plochy.
2. Kresba mimo hranice nastavené masky.
3. Přestavení horizontů.
4. Zpět k bodu 2. až do vyčerpání intervalu $<y_c, y_d>$.

Po vykreslení řezů v jednom směru, budeme kreslit ve druhém směru. Při tomto postupu může dojít k chybám (zcela jistě dojde), je vhodné počítat a kreslit plochu v obou směrech "současně". I když i tato metoda -"zig-zag" - může mít svá úskalí, je přesnější. Postup volíme tak, že počítáme od "nejbližšího" rohu plochy k "oku" pozorovatele.



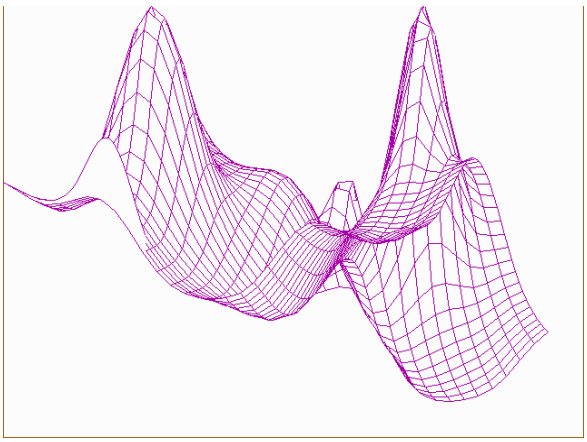
Obr. 8.11

Postup vykreslování horizontů je naznačen na obrázku číslo 8.11.

Pořadí vykreslování horizontů je:

1 a 2;
3 a 4;
5, 6, 7 a 8;
9, 10, 11, 12, 13 a 14;
15, 16, 17 a 18;
19 a 20.

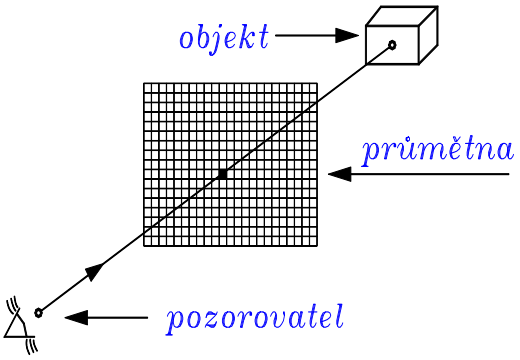
Na obrázku 8.12 je zobrazena plocha, kde neviditelné křivky plochy jsou potlačeny metodou plovoucího horizontu.



Obr. 8.12

8.3. Metoda zpětného sledování paprsku

Princip této metody spočívá ve sledování paprsků, které přicházejí ze scény k pozorovateli. A to tak, že paprsek, který se "odrazí" od pozorovaného objektu do oka pozorovatele protne promítací rovinu. Tento průsečík je potom zobrazen na stínítku obrazovky. Tato metoda (ray-tracing) je stejně tak, jako metoda Z-bufferu náročná na čas a paměť. Rozlišovací schopností obrazovky je dán počet paprsků, které musíme sledovat. Každý pixel, kterým paprsek prochází, je "vybarven" dle plochy, kterou protne. Jestliže jde o středové promítání je nutno aplikovat transformační rovnice této perspektivy. Je tedy zřejmé, že tato metoda resp. její rychlost je závislá na dobré analýze objektů na scéně. Zjistit, které paprsky vůbec stínítko protínají, v jaké vzájemné poloze se objekty nacházejí a pod. Při geometrické interpretaci jde tedy o průsečíky paprsků jdoucích středem promítání (okem pozorovatele) s plochou - objektem pozorovaného tělesa. Zobrazíme potom průsečík nejbližšího z nich. Pokud je zdroj světla z místa průsečíku viditelný, určí se barva na základě velikosti odchylky k normálovému vektoru, nebo pokud je zdroj zakrytý, vytvoří se stín. Pokud paprsek žádný objekt neprotne, pak pixel bude mít barvu pozadí. Tento výpočtový čas zabírá až 90% z celkového času zpracování. Vyplatí se tedy vypracovat pomocné metody, kdy zjistíme, které paprsky protínají resp. neprotínají dané plochy - tělesa. Za tím účelem se objekty "obalují" jednoduššími objekty jako jsou koule, krychle, hranoly a pod. U těchto objektů dokážeme snadněji omezit "kužely" paprsků, které má smysl zkoumat.



Obr. 8.13

Mějme tedy objekt - plochu H danou parametrickou rovnicí

$\overline{H}(u, v) = (x(u, v), y(u, v), z(u, v))$ s parametry u a v.

Paprsek, který budeme sledovat bude vyjádřen taktéž parametricky

$\overline{R}(t) = \overline{P} + t \cdot \overline{D}$, kde t je parametr a $\overline{D}$ je vektor paprsku.

Průsečík dostaneme řešením rovnice $\overline{H}(u,v) = \overline{R}(t) = 0$.

Výpočet můžeme provést různými metodami numerických výpočtů.

Shrnutí pojmů

Metody na odstraňování neviditelných hran jsou rozděleny na dvě části.

Obrazová metoda vychází ze zkoumání jednotlivých pixelů obrazu a rozhoduje. který objekt v daném pixelu je zobrazen. Obrazové metody jsou: Z-buffer, malířův algoritmus, Watkinsův řádkový algoritmus, Warnockův algoritmus, plovoucí horizont.

Objektová metoda zkoumá 3D objekty jednak samostatně a dále vzájemné souvislosti. Jestli se překrývají a podobně. Charakteristický algoritmus, který tvoří (v podstatě) základ objektových metod je Robertsův algoritmus, který bude probrán v následující textové části.

Efektivní algoritmy pro řešení viditelnosti jsou však kombinací metod objektových i obrazových.

Otázky 8. Viditelnost

1. Popište způsob určování viditelnosti hran resp. stěn u hranatých těles.
2. Charakterizujte princip objektových a obrazových metod viditelnosti objektů.
3. Popište principy metod: Z-buffru, Malířova algoritmu, Watkinsůva řádkového algoritmu, Warnockova algoritmu.
4. Vysvětlete princip plovoucího horizontu.