Grafika komputerowa
Wykład 9

Algorytmy wyznaczania obiektów zasłonietych

Romuald Kotowski

Instytut Informatyki i Automatyki

Państwowa Wyższa Szkoła Informatyki i Przedsiębiorczości w Łomży

2009

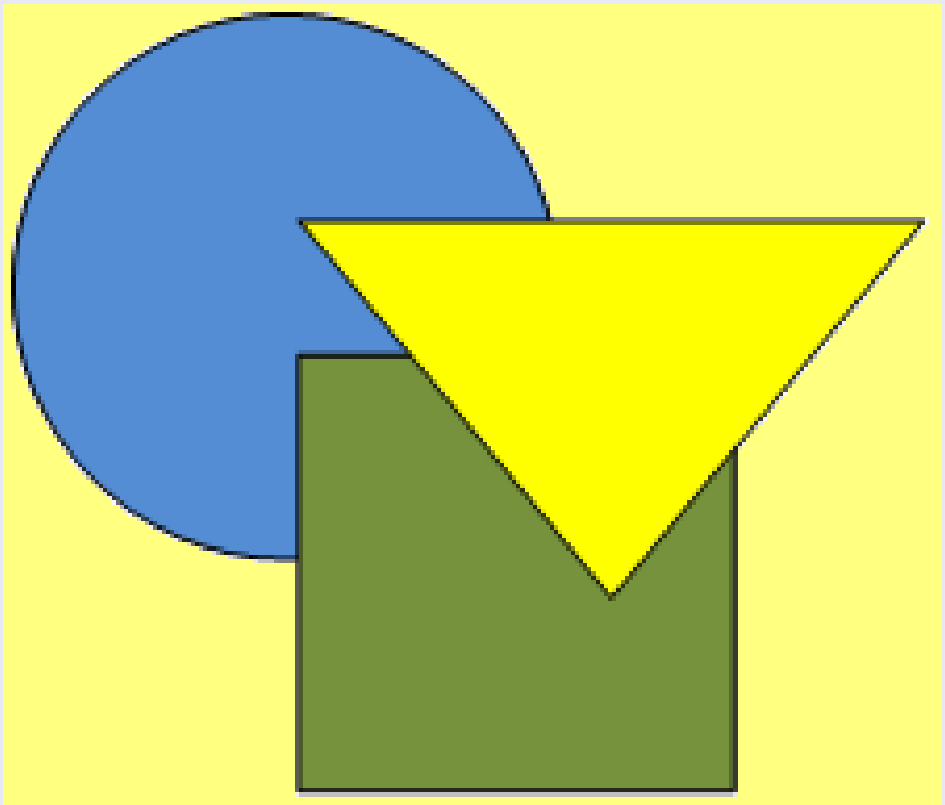
Spis treści

Spis treści

Wstęp

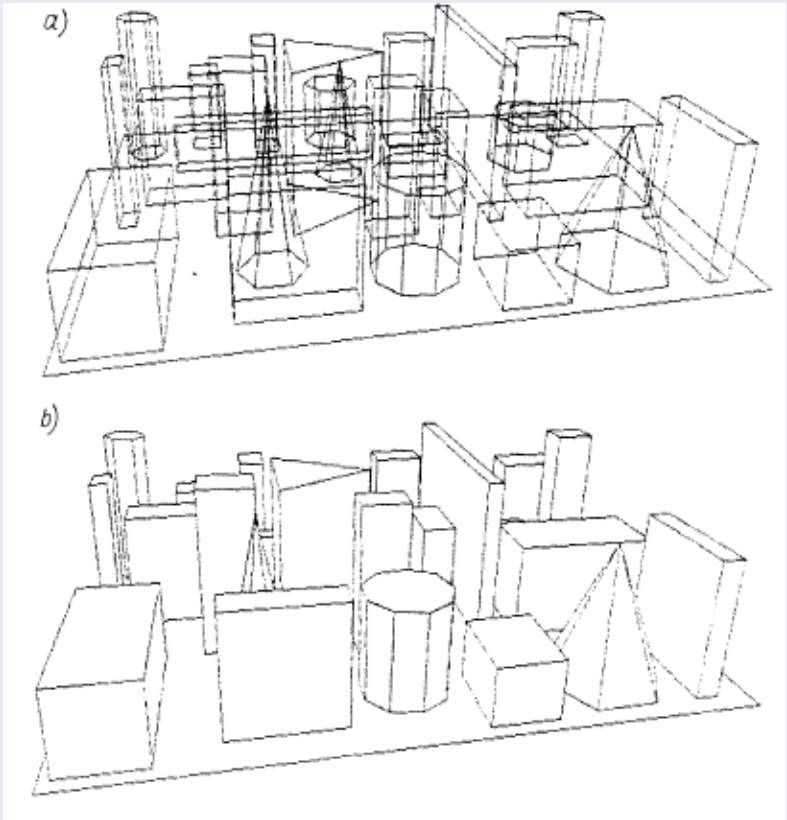
Metody wyznaczania zasłoniętych linii i powierzchni w istotny sposób zależą od urządzeń graficznych, na których tworzymy obraz. W przypadku monitorów ekranowych można malować jedno na drugim, wymazywać wcześniej narysowane elementy itp. Sprowadza się to do zmiany zawartości bufora ekranu. Natomiast większość urządzeń rysujących nie umożliwia stosowania takich technik. Rysunek 1 na monitorze otrzymuje się bardzo łatwo – malujemy w kolejności odwrotnej niż przy tworzeniu obrazów na szkle: najpierw koło, potem częściowo przykrywający je kwadrat i na nich trójkąt. Uzyskanie tego samego rysunku na ploterze jest już bardziej skomplikowane; wymaga bowiem wyznaczenia odpowiednich fragmentów koła i kwadratu.

Wstęp



Rys. 1: Przykład rysunku, który znacznie trudniej jest wykonać na ploterze niż wyświetlić na ekranie

Wstęp



Rys. 2: Rysunek tego samego obiektu: a) ze wszystkimi liniami; b) z usuniętymi fragmentami zasłoniętymi

Wstęp

Podział metod na dyskretne (pikselowe) i analityczne dobrze ilustrują dwie różne metody wizualizacji powierzchni, jakie za chwilę omówimy. Pierwsza wykorzystuje algorytm Bresenhama rysowania odcinków i wyznacza zbiory pikseli odpowiadające widocznym fragmentom linii, a w drugiej nie zasłonięte obszary są określane analitycznie, co wymaga rozwiązywania układów równań nieliniowych.

Inna klasyfikacja omawianych dziś algorytmów jest związana z przestrzenią, w której działają. Potrzebne obliczenia i porównania między elementami geometrycznymi rysowanej sceny mogą być wykonywane w trójwymiarowej przestrzeni obiektu; takie metody są nazywane *metodami przestrzeni danych*. Odróżnia się je od *algorytmów przestrzeni obrazu*, w których są badane dwuwymiarowe rzuty elementów obiektu na płaszczyznę rysunku i dopiero gdy rzuty maja części wspólne, rozstrzyga się zasłanianie w trójwymiarowej przestrzeni danych.

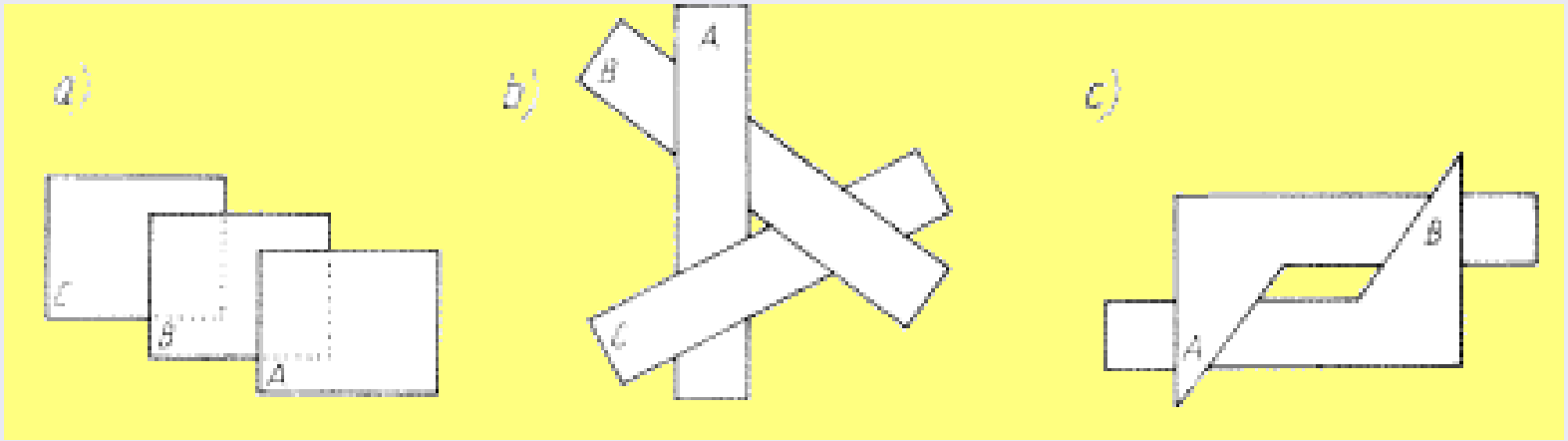
Wstęp

Algorytmy przestrzeni danych najczęściej dają dokładne wyniki analityczne i są na ogół kosztowniejsze. *Metody przestrzeni obrazu* stosuje się głównie na monitorach rastrowych; niektóre z tych metod – na przykład algorytm Watkinsa przeglądania obrazu liniami poziomymi – naśladują pracę takiego urządzenia, a większość uwzględnia jego rozdzielczość. Różne są charakterystyki kosztu obu grup metod. Liczba działań wykonywanych w algorytmach przestrzeni danych zależy głównie od złożoności rysowanej sceny, na przykład od liczby krawędzi obiektów wielościennych, a w metodach przestrzeni obrazu jest raczej funkcją liczby pikseli monitora.

Zadanie wyznaczania widocznych fragmentów obiektu trójwymiarowego można traktować jako *zadanie sortowania*. Elementy leżące dalej od obserwatora (głębiej w kierunku patrzenia) są ewentualnie zasłaniane przez lezące bliżej. Trzeba pamiętać, że

- relacja częściowego zasłaniania nie jest przechodnia: z faktu, że A zasłania fragment B, a B zasłania część C, wcale nie musi wynikać, że A zasłania fragment C (rys. 3a); czasami jest odwrotnie: A zasłania część B, dalej B zasłania fragment C, a C zasłania część A (rys. 3b);
- relacja częściowego zasłaniania nie jest relacją antysymetryczną: A może częściowo zasłaniać B, a jednocześnie B zasłaniać fragment A (rys. 3c).

Wstęp



Rys. 3: a) Częściowe zasłanianie nie jest relacją przechodnią; b) zakleszczenie wzajemnych zasłonięć; c) wzajemne zasłanianie się dwóch ścian

Wyznaczanie zasłoniętych fragmentów linii obiektów wielościennych

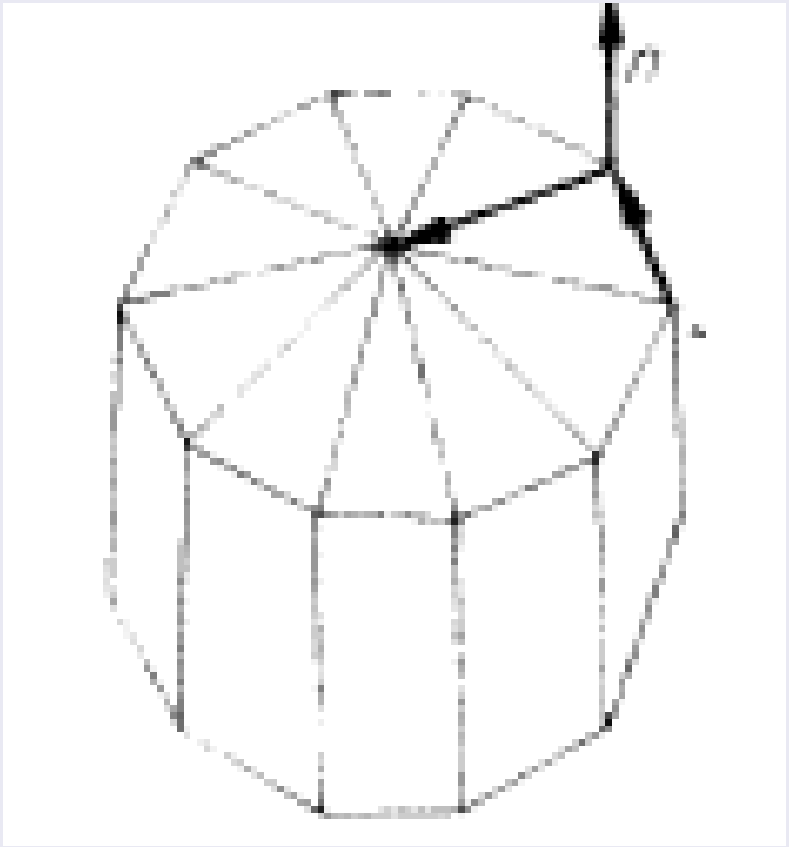
Obiekty wielościenne

Przypomnijmy, że obiekty wielościenne opisuje się na ogół listą wierzchołków (tablicą zawierającą ich współrzędne), listą par numerów wierzchołków określających krawędzie i wreszcie listą numerów krawędzi definiujących ściany.

Rozważmy najpierw przypadek rysowania sceny trójwymiarowej, składającej się z dowolnego, ale tylko jednego wielościanu wypukłego. Możemy pominąć ściany równoległe do kierunku patrzenia; ich rzuty na płaszczyznę rysunku redukują się do odcinków. Każda z pozostałych ścian jest całkowicie widoczna albo cała zasłonięta.

Możemy zakładać, że krawędzie ściany są uporządkowane odwrotnie do kierunku ruchu wskazówek zegara. Wtedy iloczyn wektorowy dowolnych dwóch kolejnych krawędzi wyznacza wektor normalny **n**, prostopadły do ściany i skierowany przeciwnie do kierunku patrzenia, czyli na zewnątrz wielościanu (rys. 4).

Obiekty wielościenne



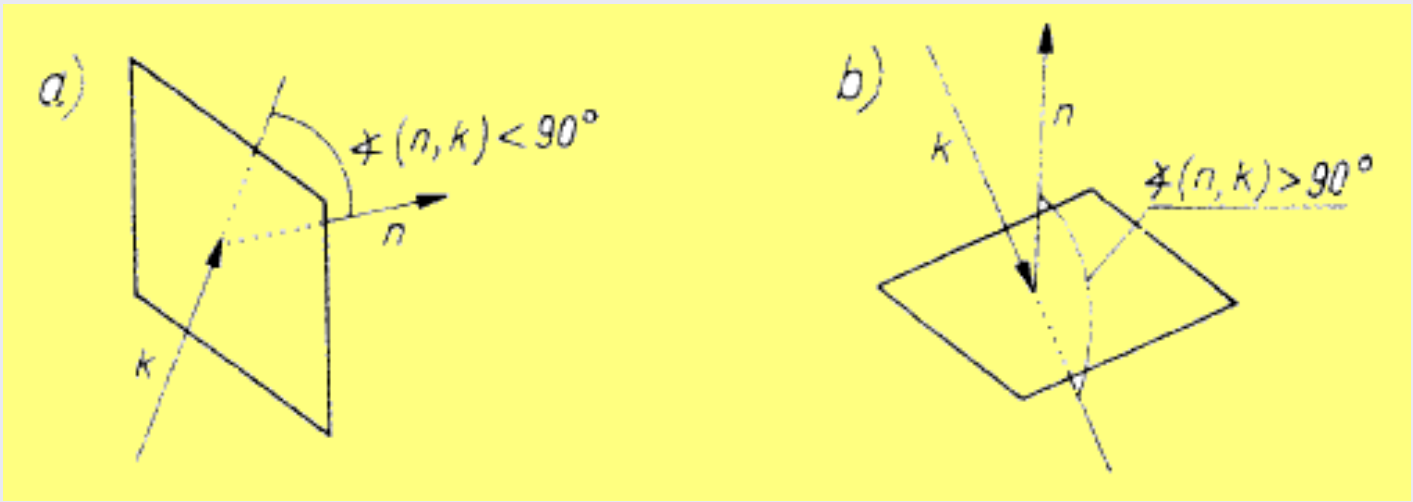
Rys. 4: Określanie widoczności ściany bryły wypukłej

$$n \cdot k = |\mathbf{n}|\,|\mathbf{k}|\,\cos\angle(\mathbf{n},\mathbf{k})$$

k – wektor określający kierunek patrzenia: przy rzucie równoległym **k** jest stały; przy rzucie środkowym za **k** można przyjąć wektor o początku w punkcie położenia obserwatora (czyli w środku rzutowania) i końcu w dowolnym punkcie danej ściany.

- jeśli $n \cdot k > 0$, to kąt między **n** i **k** jest mniejszy niż 90°, co oznacza, ze cała ściana jest zasłonięta, tj. skierowana 'tyłem' do kierunku patrzenia (rys. 5a)
- jeśli $n \cdot k < 0$, to cała ściana jest widoczna (rys. 5b).

Obiekty wielościenne



Rys. 5: Skierowanie ścian: a) tyłem; b) frontem do kierunku patrzenia

Spis treści

Algorytm Ricciego

Założenia

Dowolny trójwymiarowy obiekt (scena) zależnie od sposobu rzutowania mieści się w odpowiednim prostopadłościanie lub ostrosłupie widzenia;

- scena jest zbudowana z płaskich wielokątów wypukłych ograniczonych krawędziami i ewentualnie z dodatkowych odcinków nie będących krawędziami ścian;
- żadna z krawędzi ani żaden odcinek nie przenikają żadnej ściany.

Algorytm Ricciego sprowadza się do porównywania rzutu każdej krawędzi i każdego odcinka z rzutami wszystkich ścian i wyznaczania widocznych fragmentów badane1 krawędzi (odcinka). Założenie, że ściany są wielokątami wypukłymi, pozwala na istotne zmniejszenie kosztu obliczenia części wspólnej rzutu odcinka i ściany.

Algorytm Ricciego

Oznaczenia

$P_i(x_i, y_i, z_i)$, $i = 1, 2, \ldots, lp$ – wierzchołki ścian albo końce odcinków tworzących rysowaną scenę

(xp_i, yp_i) – współrzędne rzutów P' punktów P na płaszczyznę rysunku

k_j, $j = 1, 2, \ldots, lk$ – krawędzie

o_l, $l = 1, 2, \ldots, lo$ – odcinki

TO – tablica par krańcowych punktów krawędzi i odcinków; jeśli jakaś krawędź jest wspólna dla kilku ścian, to w TO występuje tylko raz.

LKS – tablica liczby krawędzi ścian; $LKS(m)$ – liczba krawędzi ściany m

$APK(m)$ – adres pierwszej krawędzi ściany m

TK – tablica krawędzi

Algorytm Ricciego

Oznaczenia

Np., m-ta ściana jest ograniczona krawędziami o numerach

$$TK(APK(m)), \ldots, TK(APK(m) + LKS(m) - 1)$$

Zakładamy, że krawędzie ściany są uporządkowane odwrotnie do kierunku ruchu wskazówek zegara. Jeśli pewne ściany tworzą brzeg zamkniętej bryły, a więc mogą być widoczne tylko z jednej strony, to definiujemy je tylko raz, od tej zewnętrznej strony. Pozostałe ściany muszą być określone dwukrotnie; trzeba przy tym pamiętać o odwrotnym uporządkowaniu krawędzi dla różnych stron tej samej ściany.

Algorytm Ricciego wyznaczania widocznych fragmentów linii obiektów wielościennych

I

dla każdej ściany ($m = 1, 2, \ldots, ls$) sprawdzamy, czy nie jest odwrócona 'tyłem'; jeśli tak, to zmieniamy liczbę krawędzi tej ściany na ujemną, $LKS(m) = -LKS(m)$;

dla wszystkich krawędzi k_j ($j = 1, 2, \ldots, lk$) i wszystkich odcinków o_l ($l = 1, 2, \ldots, lo$)

jeśli badany odcinek leży na ścianie odwróconej 'tyłem', (czyli gdy $LKS(NS(l)) < 0$), to go pomijamy;

w tablicy TO znajdujemy numery np – początkowego i nk – końcowego punktu krawędzi (odcinka);

inicjujemy listę widocznych fragmentów rzutu, $r = P'_{np}\,P'_{nk}$, przyjmując $lwf = 1$ oraz $tp(1) = 0$ i $tk(1) = 1$ zakładamy, że odcinek $P'_{np} + t(P'_{nk} - P'_{np})$, ($t \in [0, 1]$) jest cały widoczny;

Algorytm Ricciego . . .

II

dla wszystkich ścian S_m, $m = 1, 2, \ldots, ls$
jeśli $LKS(m) < 0$ lub k_j jest krawędzią S_m lub o_l leży na ścianie S_m **to** daną ścianę pomijamy

w przeciwnym razie wyznaczamy fragment r leżący wewnątrz rzutu ściany S_m;

jeśli ten fragment jest pusty, to przechodzimy do badania zasłaniania danej krawędzi (odcinka) następną ścianą

w przeciwnym razie niech $P'_{np} + t_{min}(P'_{nk} - P'_{np})$ i $P'_{np} + t_{max}(P'_{nk} - P'_{np})$, gdzie $0 \leq t_{min} \leq t_{max} \leq 1$ są końcami fragmentu r leżącego wewnątrz rzutu S_m; w celu rozstrzygnięcia, czy ten fragment jest widoczny, czy zasłonięty ścianą S_m, znajdujemy jego środek, gdzie

$$P'_{sr} = (x_{sr}, y_{sr}) = P'_{np} + t_{sr}(P'_{nk} - P'_{np})$$

i $t_{sr} = (t_{min} + t_{max})/2$, a następnie punkty w przestrzeni $3D$: K – leżący na badanej krawędzi (odcinku) i S – leżący na ścianie S_m, takie, że P'_{sr} jest ich wspólnym rzutem na płaszczyznę rysunku;

Algorytm Ricciego . . .

III

jeśli punkt K leży głębiej w kierunku patrzenia niż S, lub jeśli K leży po przeciwnej stronie ściany niż obserwator (środek rzutowania przy rzucie perspektywicznym),
to wyznaczony fragment r jest zasłonięty ścianą S_m i wtedy od sumy przedziałów

$$[tp_1, tk_1] \cup [tp_2, tk_2] \cup \ldots \cup [tp_{lwf}, tk_{lwf}]$$

odejmujemy przedział $[t_{min}, t_{max}]$;

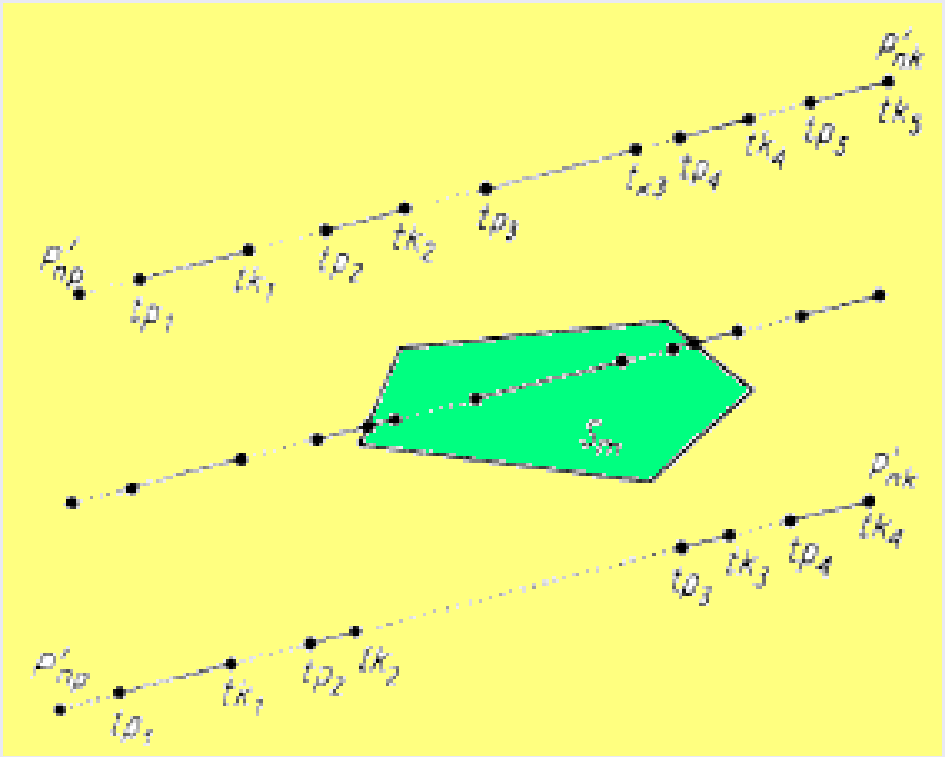
jeśli wynik jest zbiorem pustym, to kładziemy $lwf = 0$ i przerywamy badanie danej krawędzi (odcinka), gdyż cała krawędź jest zasłonięta;

w przeciwnym razie odpowiednio modyfikujemy wartość lwf, a przedziały otrzymane w wyniku odejmowania numerujemy tak (patrz rys. 6), by

$$tp_1 < tk_1 < tp_2 < tk_2 < \cdots < tp_{lwf} < tk_{lwf}$$

wynik: widoczne jest lwf fragmentów krawędzi (odcinka) $P'_{np} + t(P'_{nk} - P'_{np})$ dla $t \in [tp_i, tk_i]$, $i = 1, 2, \ldots, lwf$

Algorytm Ricciego . . .



Rys. 6: Numeracja fragmentów krawędzi w algorytmie Ricciego

Spis treści

Algorytm Appela

Appel w swym algorytmie wykorzystał fakt, że liczba ścian zasłaniających punkty badanego odcinka zmienia się tylko w punktach przecięcia z rzutami szczególnych krawędzi. Algorytm ten doczekał się wielu modyfikacji i ulepszeń, ale ze względu na czytelność idei warto przedstawić pierwowzór. W stosunku do założeń z poprzedniego punktu wprowadzamy dwie zmiany:

- ściany mogą być dowolnymi wielokątami, niekoniecznie wypukłymi
- nie dopuszczamy odcinków leżących na ścianach i nie będących ich krawędziami

Oznaczenie: $lz(P)$ – liczba ścian (pomijając odwrócone tyłem), które zasłaniają punkt P.

Algorytm Appela

I

znajdujemy ściany odwrócone tyłem, pozostałe traktujemy jako potencjalnie widoczne;
pomijamy krawędzie ograniczające tylko ściany odwrócone tyłem, resztę nazywamy istotnymi i wyróżniamy wśród nich *krawędzie konturowe* – ograniczające jednocześnie ściany potencjalnie widoczne i odwrócone tyłem;
dla rzutów $P'_iP'_j$ wszystkich istotnych krawędzi
obliczamy początkową wartość $lz = lz(P'_i)$; wyznaczamy punkty $Q'_k = Q'_k(t_k)$ przecięcia odcinka $P'_i + t(P'_j - P'_i)$, $t \in (0, 1)$ ze wszystkimi krawędziami konturowymi i porządkujemy je tak, by

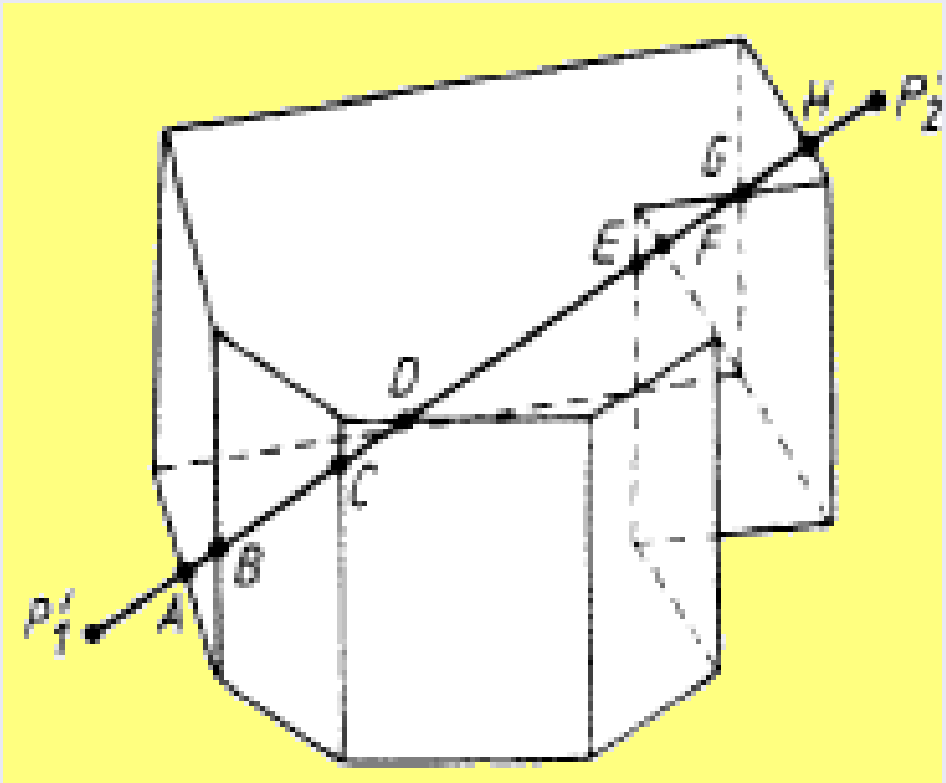
$$t_1 < t_2 < \cdots < t_l$$

Algorytm Appela

II

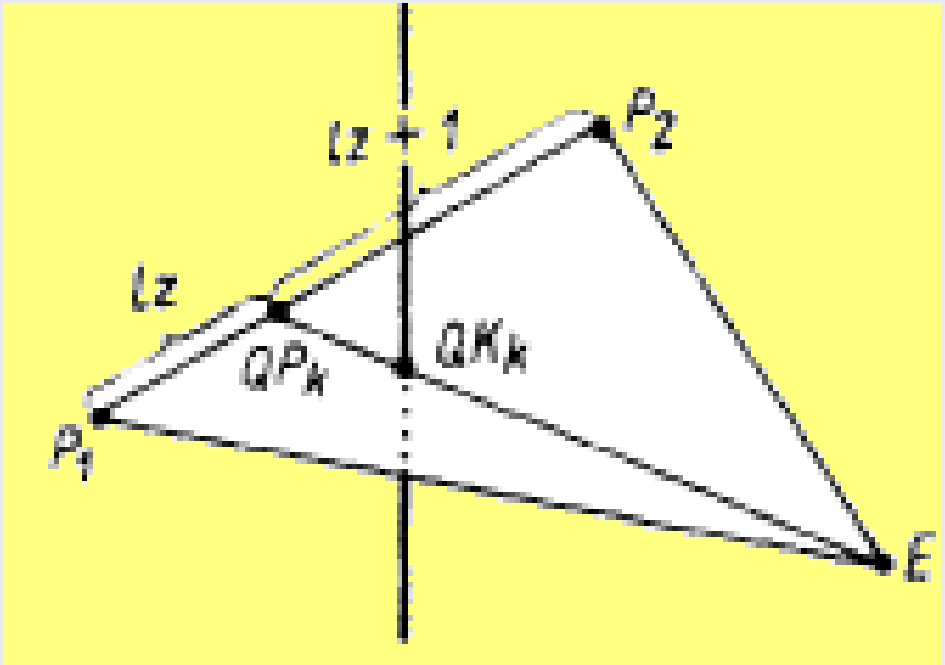
dla $k = 1, 2, \ldots l$
niech QK_k i QP_k będą punktami odpowiednio krawędzi konturowej i odcinka P_iP_j, których wspólnym rzutem jest Q'_k;
wartość lz ulega zmianie w punkcie Q'_k, jeśli QK_p leży bliżej obserwatora (bliżej w kierunku patrzenia);
dla rzutu środkowego taką sytuację ilustruje rys. 8; dokładniej, dla P' leżących za Q'_k jest $lz = lz + 1$, jeśli krawędź P_iP_j 'wchodzi za ścianę', a w przeciwnym przypadku, tzn. kiedy P_iP_j 'wychodzi zza ściany', jest $lz = lz - 1$.

Algorytm Appela



Rys. 7: Zasłanianie fragmentów krawędzi w algorytmie Appela

Algorytm Appela



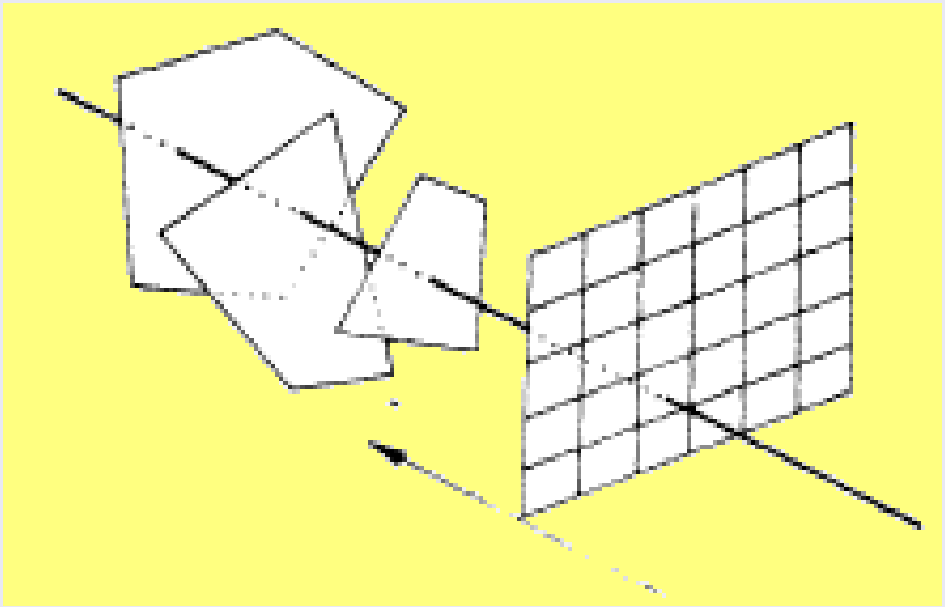
Rys. 8: Zmiana liczby ścian zasłaniających

Spis treści

Algorytm z buforem głębokości

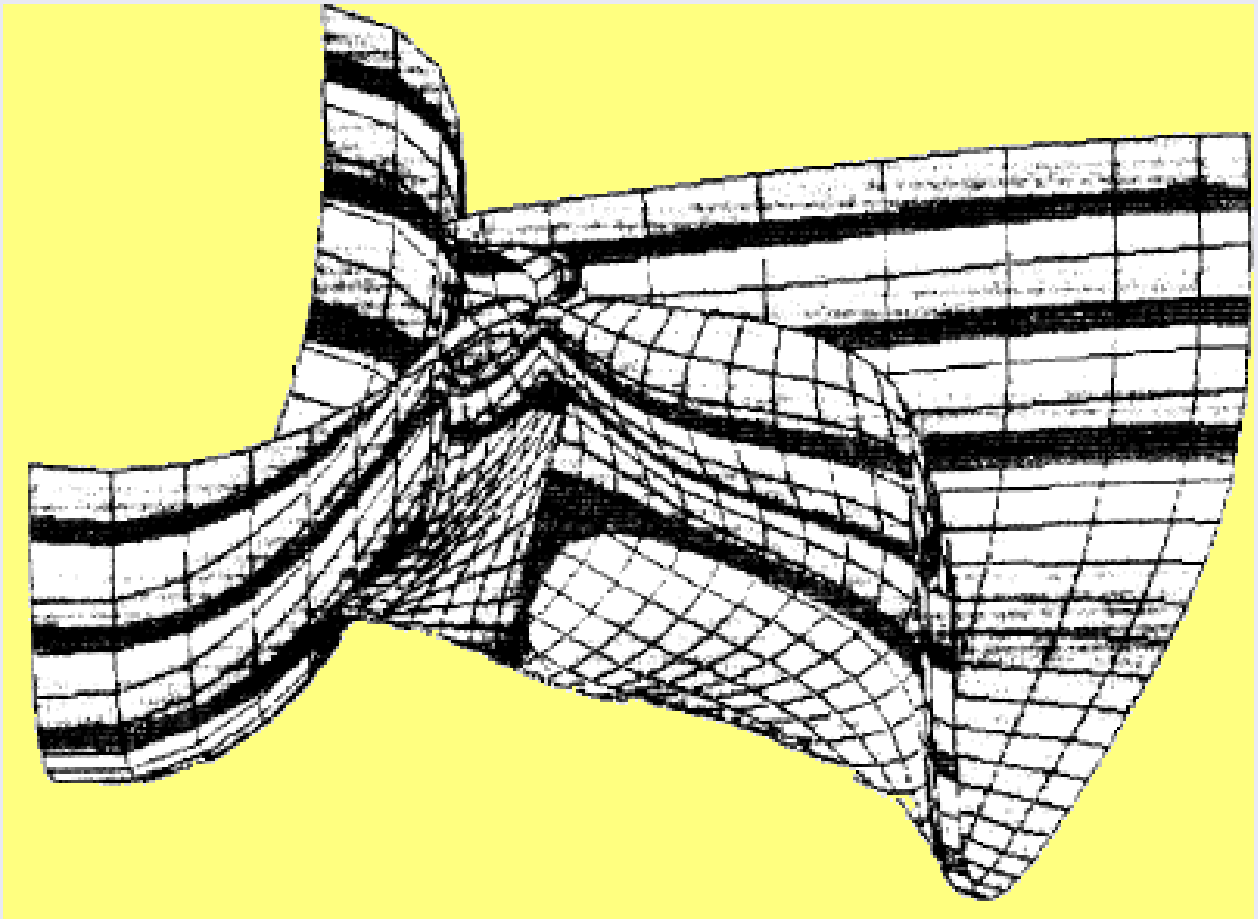
Algorytm z buforem głębokości jest jednym z najłatwiejszych do implementacji, a przy tym można go stosować do szerokiej klasy obiektów. Służy do wyznaczania widocznych fragmentów ścian i ich wizualizacji na urządzeniu rastrowym: kolorowa drukarka, obraz na ekranie,... Będziemy wypełniali bufor, np. ekranu, znajdując dla każdego piksela ścianę leżącą najbliżej przed nim w kierunku patrzenia i przypisując pikselowi kolor/jasność odpowiedniego fragmentu tej ściany (rys. 9).
Ściany rysowanego obiektu mogą być dowolnymi (niekoniecznie wypukłymi) wielokątami; mogą też mieć dziury i wzajemnie się przenikać. Zakładamy, że dana jest funkcja $kolor(S, x, y, z)$ przypisująca każdemu punktowi o współrzędnych (x, y, z) leżącemu na ścianie S kolor/jasność. W przykładowym rys. 10 różne kolory odpowiadają kolejnym poziomym warstwom w przestrzeni $\mathbb{R}^3$.

Algorytm z buforem głębokości



Rys. 9: Idea algorytmu z buforem głębokości

Algorytm z buforem głębokości



Rys. 10: Nie zasłonięte fragmenty powierzchni B-sklejanej wyznaczone algorytmem z buforem głębokości

Algorytm z buforem głębokości

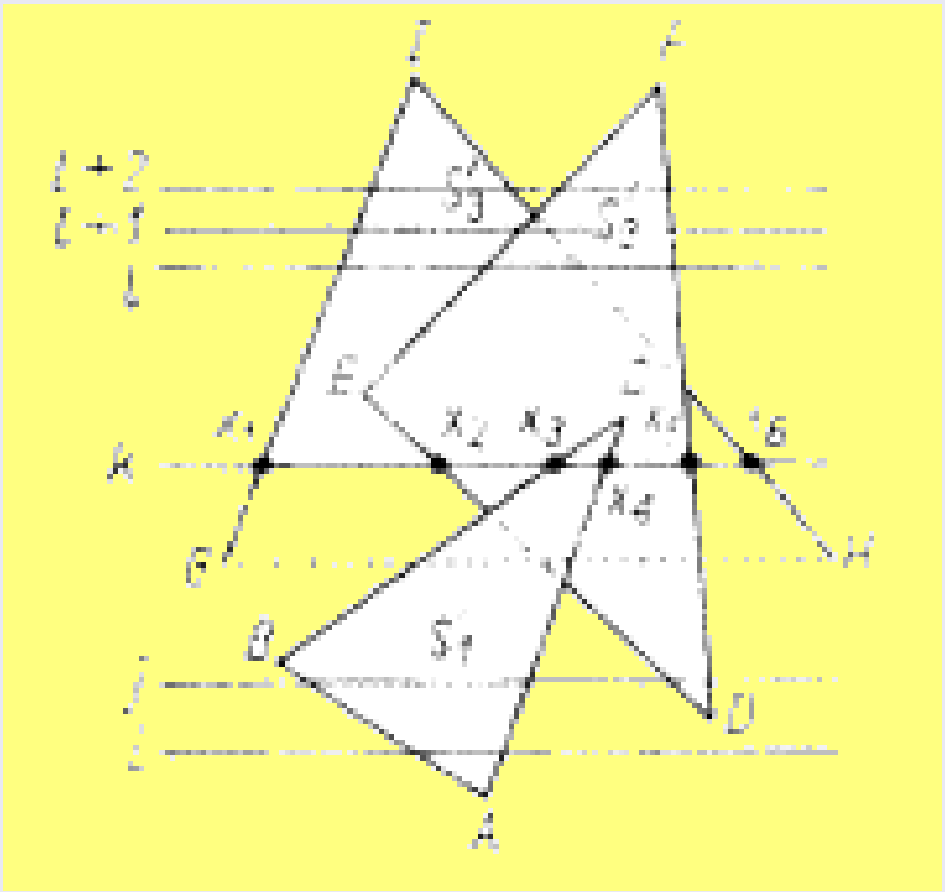
Inicjujemy bufor *buf* ekranu i tablicę *tab* głębokości, podstawiając
dla wszystkich pikseli (i, j) ekranu
$but(i, j) = tlo$;
$tab(i, j) = rmax$;
wartość *tlo* określa kolor tła obrazu, a *rmax* jest największą liczbą rzeczywistą w stosowanej arytmetyce;
dla wszystkich ścian S_m, rysowanej sceny
dla wszystkich pikseli (i, j) zawartych w rzucie S_m obliczamy współrzędne rzeczywiste (x, y) odpowiadające pikselowi (i, j);
obliczamy współrzędną z przeciwobrazu (x, y) leżącego na ścianie S_m;
jeśli $z < tab(i, j)$, **to**
$\{tab(i, j) = z;\ buf(i, j) = kolor(S_m, x, y, z)\}$

Spis treści

Algorytm przeglądania liniami poziomymi Watkinsa

Zakładamy, ze ściany nie przenikają się i każda jest jednokolorowa (to dla uproszczenia opisu). Podkreślmy natomiast, że wielokąty mogą być dowolne – nie ograniczamy się do wypukłych. Przez krawędzie rozumiemy dalej rzuty krawędzi ścian na płaszczyznę rysunku.
Początkowo bufor ekranu wypełniamy kolorem tła. Rzut rysowanej sceny przeglądamy liniami poziomymi od dołu do góry l z lewa na prawo. Tak samo jak przy wypełnianiu wielokąta, tworzymy tablicę krawędzi aktywnych *TKA*, zawierającą informacje tylko o tych krawędziach, które przecina dana linia pozioma y.

Algorytm z buforem głębokości



Rys. 11: Przeglądanie rzutów ścian liniami poziomymi

Literatura

[1] M. Jankowski, Elementy grafiki komputerowej, WNT, 2006

Koniec? :-(

Koniec wykładu 9