Podstawy Grafiki Komputerowej

Wykład 5: Widoczność

Andrzej Łukaszewski

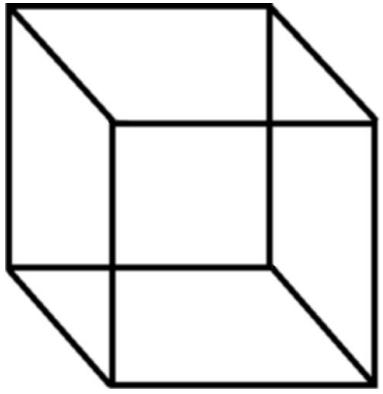
Pracownia Grafiki Komputerowej
(LIGHT — Laboratory of Imaging and GrapHic Techniques)

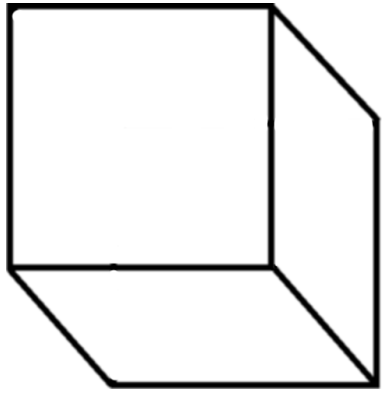
Instytut Informatyki, Uniwersytet Wrocławski

4 listopada 2019

Widoczność: wstęp

- Po właściwej transformacji współrzędnych uwzględniającej rzutowanie do renderowania trzeba uwzględnić widoczność.
- Widoczność to po paralaksie (para oczu) i perspektywie kolejna informacja dla systemu widzenia o przestrzennym położeniu przedmiotów, 3D.
- Niejednoznaczność interpretacji:





(rzut równoległy bez perspektywy, widoczność!)

Widoczność: definicja problemu

Widoczność obiektu/ściany ?

Czy obiekt/ściana jest widoczna dla danego obserwatora? W całości, częściowo?

Uogólnienie: widoczność obiektu z regionu

Dla wszystkich położeń obserwatora w regionie np. prostopadłościennym czy obiekt jest widoczny: z jakiegoś punktu ? dla każdego punktu ?

Definicja: widoczność punktu

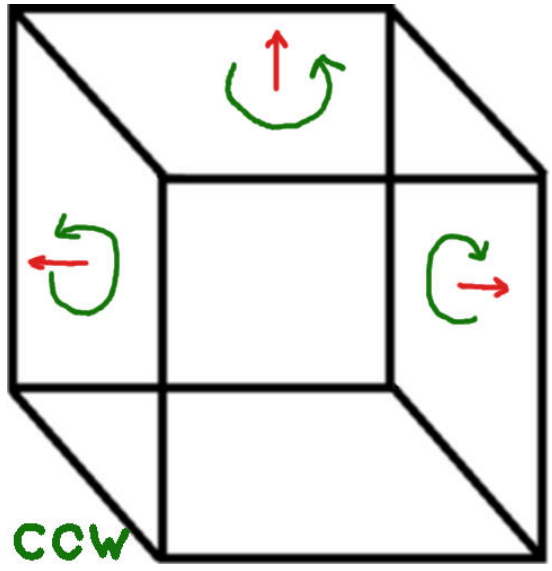
Punkt P jest widoczny dla obserwatora znajdującego się w O jeśli odcinek PO nie przecina innego obiektu (który go zasłania).

Widoczność: przegląd i klasyfikacja

- Jeden z pierwszych klasycznych problemów C.G.:
  Usuwanie niewidocznych linii i powierzchni (hidden surface/line removal/determination)
- Klasyczne rozwiązania z lat ‘60-‘70-tych:
  - ray casting 1968,
  - z—bufor 1974,
  - sortowanie, . . .
- Klasyfikacja: algorytmy działające z precyzją: obiektową lub obrazową (pikselową)
  Albo ustalamy widoczne części ścian albo dla pikseli ustalamy widoczność.

Widoczność: back face culling

Trywialne odrzucanie ścian tylnych



- Działa dla prostych przypadków: ..... jakich?
- Warunki: ściany są jednostronne i wektory normalne wskazują poprawnie czyli wierzchołki podane w ustalonej kolejności: *CW*/*CCW*, koszt: 1 iloczyn skalarny
- Metoda preprocessingu dla innych metod
- OpenGL: back CW, front CCW, czyli tylko FRONT:

  `glEnable(GL_CULL_FACE); glCullFace(GL_BACK);`

Widoczność: bufor głębokości

- 1974: E. Catmull/USA i równolegle W.Strasser/Niemcy
- z—bufor: dla każdego piksela głębokość
- algorytm w przestrzeni obrazu
- stosowany w sprzęcie (karty, OpenGL), a także programowo (np. tak powstało wiele filmów: renderer firmy Pixar)

Algorytm: inicjalizacje bufora koloru i z-bufora ($\pm inf$ lub far cl.)

```
1  dla każdego obiektu
2     dla każdego piksela (x,y) zrasteryzowanego obiektu
3        pz = oblicz odległość od obserwatora
4        jeśli pz<GetZBuffer(x,y)
5        wtedy
6           PutZBuffer(x,y,pz)
7           PutPixel(x,y, odpowiedni kolor)
```

Widoczność: bufor głębokości

Uwagi:

1. Działa ogólnie dla obiektów dla których potrafimy podać sposób rasteryzacji rzutu i metodę obliczania głębokości.
2. rasteryzacja/rysowanie wielu wielokątów niepotrzebnie
3. koszt zależy od kolejności (kolejność od tyłu do przodu zła)
4. koszt rośnie wraz z ilością obiektów ale bardziej wraz z ilością rasteryzowanych pikseli wiec podział nie powoduje katastrofy
5. dokładność bufora 16,24,32 bity między płaszczyznami obcinającymi $\rightarrow$ artefakty

Obliczanie odległości dla wielokątów mamy równanie płaszczyzny, tak więc w odpowiednich współrzędnych:

$$z = \frac{-D - Ax - By}{C}$$

metoda różnicowa: $\Delta z = -A/C \cdot \Delta x$

Widoczność: OpenGL

```
1  // Trywialne eliminowanie ścian tylnych
2  //     -  uaktywniamy test:
3   glEnable(GL_CULL_FACE)
4  //     - określamy orientacje dla FRONT:
5   glFrontFace(GL_CCW);  // domyślnie CCW
6  //     - określamy które odrzucać: BACK/FRONT
7   glCullFace(GL_BACK);  // domyślne
8
9  // Bufor głębokości  czyli  z-bufor:
0  //     -  uaktywniamy test:
1   glEnable(GL_DEPTH_TEST);
2  //     -  określamy funkcje LESS/LEQUAL/GEQUAL/..
3   glDepthFunc(GL_LESS);
4  //     -  czyścimy bufor przed renderowaniem
5   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Widoczność: sortowanie

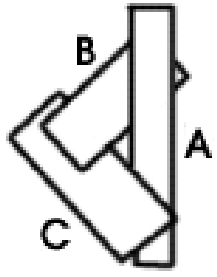
Co nam da posortowanie względem głębokości,
czyli jak pracują malarze?

Algorytm malarza

Proste malowanie: renderujemy obiekty od tyłu do przodu tak, że zamalowujemy zasłonięte obiekty.

Jak ustalić porządek ?

- posortowanie po z nie wystarcza
- zasłanianie cyklicznie 3 trójkątów.



(rys.: wikipedia)

- wzajemne zasłanianie 2 wielokątów niewypukłych . . .

Widoczność: Newell-Newell-Sancha

Ogólny schemat algorytmu:

1. posortować
2. poprawić
3. wyświetlić algorytmem malarza

Poprawianie: dla każdej pary ścian sprawdzamy czy dalsza B nie zasłania wcześniejszej A, do pierwszego testu który przejdzie:

1. zakresy x lub y rzutów A i B są rozłączne
2. A w całości po tej samej stronie płaszczyzny B co obs.
3. B w całości po przeciwnej stronie płaszczyzny A co obs.
4. rzuty A i B rozłączne

Jeśli nie spełnione to zamieniamy i wykrywamy problemy . . . np. cykle.

Widoczność: drzewa BSP

Fuchs, Kaden, Naylor 1980

Intensywne przetwarzanie wstępne, linowe wyświetlanie.

Klasyczne zastosowania: Doom, symulatory lotnicze.

Drzewo BSP określa nam kolejność zasłaniania więc do renderowania wystarczy algorytm malarza.

```
1  draw(bsptree T, point e)   // rekurencyjnie
2     if (T.empty) return
3     if ( f(T.root)(e) < 0 ) // obs. po ujemnej stronie
4        draw(T.plus, e)      // f - równanie płaszczyzny
5        rasterize(T.triangle)
6        draw(T.minus,e)
7     else
8        draw(T.minus,e)      // tu ujemna dalej
9        rasterize(T.triangle)
0        draw(T.plus, e)
```

Widoczność: drzewa BSP

PROBLEM: Jak wygenerować optymalne drzewo BSP np. dla wielokątów

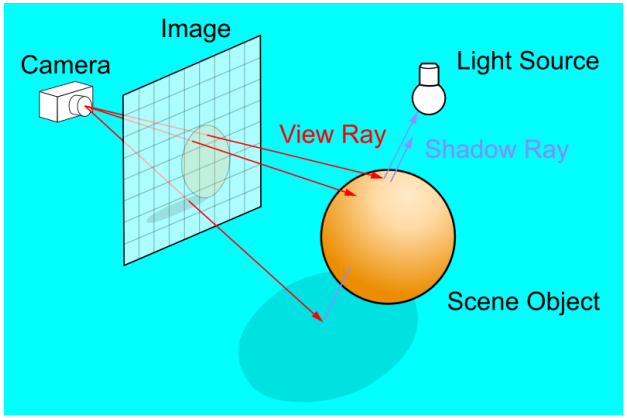
KRYTERIUM: przy wyborze płaszczyzn jak najmniej trójkątów przecinających bo powodują zwiększenie ilości obiektów

Heurystyki np. spośród zbioru wielokątów bierzemy 5 kandydatów i sprawdzamy ile ich podziały powodują rozcięć wielokątów i wybieramy najlepszy i rekurencyjnie tak tworzymy drzewo.

Zastosowanie: renderowanie obiektów przeźroczystych (kolejność!)

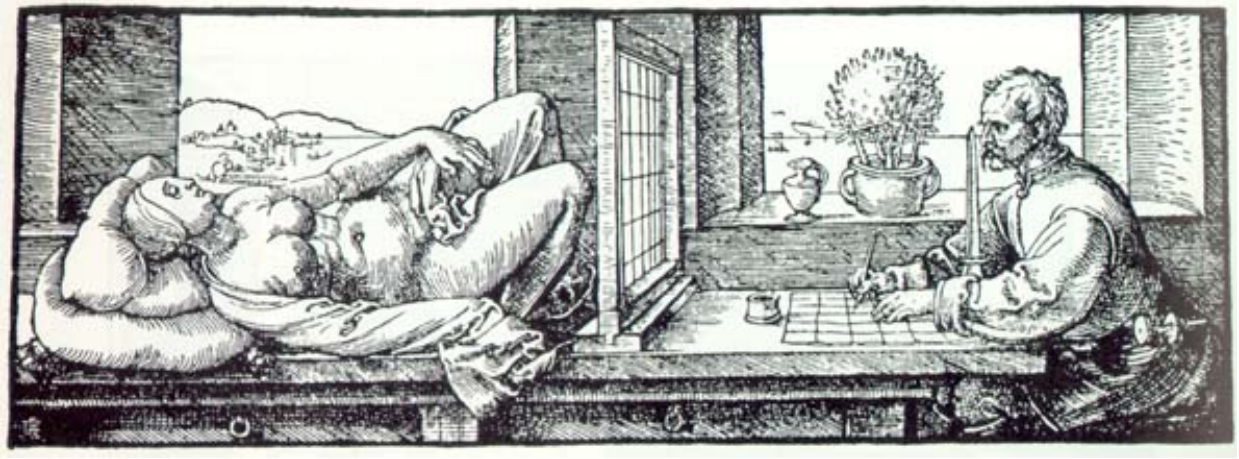
Widoczność: ray casting

Nierekurencyjna metoda śledzenia promieni Appel, 1968



Rys.:By Henrik - Own work, GFDL, https://commons.wikimedia.org/w/index.php?curid=3869326

Studia rozchodzenia światła: Leonardo da Vinci, A. Dürer



Albrecht Duerer, Underweysung der Messung mit dem Zirkel und Richtscheyt (Nurenberg, 1525), Book 3, figure 67.

Rekurencyjny ray–tracing Whitted, 1980

Widoczność: ray casting

```
1  dla każdego piksela
2     oblicz promień od obserwatora przez ekran
3     znajdź najbliższy obiekt przecięty przez promień
4     ustaw kolor piksela dla obiektu (oświetlenie?)
```

Trywialnie: trzeba znaleźć przecięcia z wszystkimi obiektami aby znaleźć najbliższe.

Bardziej zaawansowane algorytmy czynią ten algorytm praktycznym. Po dodaniu struktur przyspieszających redukujemy koszt ilości przecięć względem ilości obiektów N z $O(N)$ do $O(\log N)$.

Co jest efektywniejsze: z—bufor czy ray casting ?

Przykłady: Embree, OptiX, Imagination Tech. (Caustics)

Widoczność: ray casting

Generowanie promienia pierwotnego od obserwatora na podstawie lokalnego układu wektorów

1. W od obserwatora do wirtualnego ekranu.
2. U, V rozpinają płaszczyznę ekranu i określają kąty widzenia (np. jak w gluFrustum)

Promień jest określony przez punkt początkowy P i kierunek D.

$$D(x,y) = (W - U - V) + \frac{2x}{XRES}U + \frac{2y}{YRES}V$$

Widoczność: ray casting

Znajdowanie przecięcia ze sferą:

Promień:

$$R(t) = P + tD$$

Równanie sfery (lub innej powierzchni uwikłanej):

$$F([x, y, z]) = 0$$

Po podstawieniu mamy równanie do rozwiązania:

$$F(R(t)) = 0$$

Dla sfery otrzymujemy wiec jedno równanie kwadratowe zmiennej t. Po obliczeniu t łatwo otrzymać punkt $R(t)$ a także obliczyć wektor normalny do sfery który może być potrzebny do obliczeń oświetlenia.

Widoczność: culling c.d.

Obiekty mogą być niewidoczne z kilku powodów:

- leżą poza obszarem widzenia
- są skierowane tyłem
- są zasłonięte przez inne

Stąd mamy metody eliminujące część obiektów przed np. wysłaniem na GPU:

- view frustum culling
- back face culling
- occlusion culling, occluder fusion

Ostatnia klasa metod eliminacji obiektów zasłoniętych to szereg nowszych nietrywialnych algorytmów.

Widoczność: ..

Portal Rendering: portale i sektory

(dla wnętrz)

PVS: Potentially Visible Set (z punktu, z regionu)

- dokładne
- konserwatywne
- agresywne
- aproksymacyjne

(sceny otwarte, miasta)

P.Wonka et al. “Guided Visibility Sampling” — SIGGRAPH 2006