Grafika komputerowa

Dr inż. Michał Kruk

Podstawy geometrii analitycznej - wektory

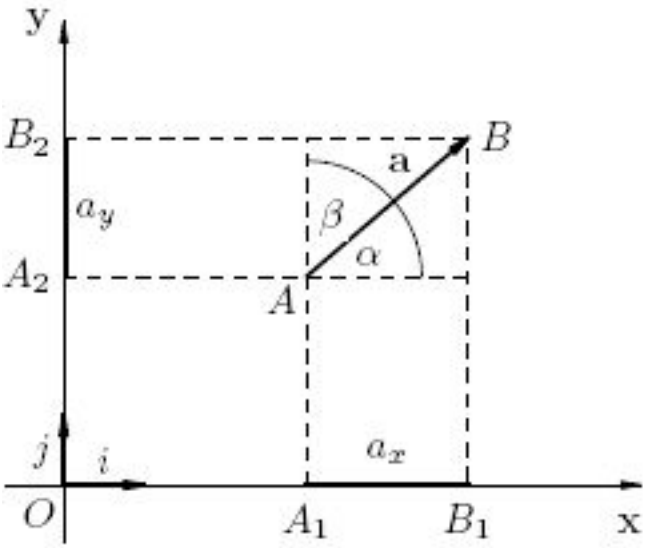
$\overline{A_1B_1} = a_x\mathbf{i}, \qquad \overline{A_2B_2} = a_y\mathbf{j}.$

$\mathbf{a} = a_x\mathbf{i} + a_y\mathbf{j}$

$a_x = x_2 - x_1, \qquad a_y = y_2 - y_1$

$|\mathbf{a}| = \sqrt{a_x^2 + a_y^2}$

$a_x = |\mathbf{a}|\cos\alpha, \qquad a_y = |\mathbf{a}|\cos\beta$

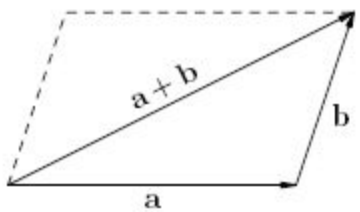


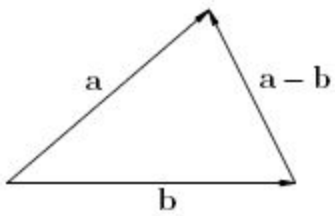
TWIERDZENIE *Jeżeli* $\mathbf{a} = (a_x, a_y)$, $\mathbf{b} = (b_x, b_y)$, $\mathbf{c} = (c_x, c_y)$ *są wektorami w prostokątnym układzie współrzędnych* Oxy *oraz* $\lambda, \mu \in R$, *to:*

1) $\mathbf{a} + \mathbf{b} = (a_x + b_x,\ a_y + b_y)$,
2) $\mathbf{a} - \mathbf{b} = (a_x - b_x,\ a_y - b_y)$,
3) $(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c})$,
4) $\lambda\mathbf{a} = (\lambda a_x, \lambda a_y)$,
5) $\lambda(\mu\mathbf{a}) = (\lambda\mu)\mathbf{a}$,
6) $\lambda(\mathbf{a} + \mathbf{b}) = \lambda\mathbf{a} + \lambda\mathbf{b}$,
7) $(\lambda + \mu)\mathbf{a} = \lambda\mathbf{a} + \mu\mathbf{a}$,
8) $\mathbf{a} = \mathbf{b}$ *wtedy i tylko wtedy, gdy* $a_x = b_x$ *i* $a_y = b_y$.

Podstawy geometrii analitycznej

- <u>Suma i różnica wektorów</u>





Iloczyn skalarny

$$\mathbf{ab} = |\mathbf{a}|\,|\mathbf{b}| \cos \sphericalangle(\mathbf{a}, \mathbf{b}),$$

gdzie:

$|\mathbf{a}|$ — *długość wektora* $\mathbf{a}$,
$|\mathbf{b}|$ — *długość wektora* $\mathbf{b}$,
$\sphericalangle(\mathbf{a}, \mathbf{b})$ — *kąt między wektorami* $\mathbf{a}$ *i* $\mathbf{b}$

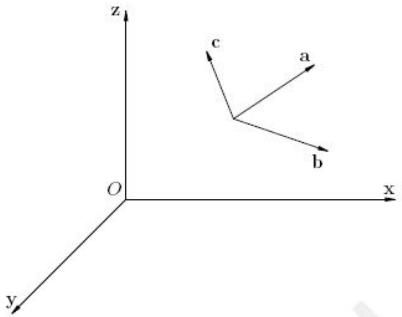
1) *jeżeli* $\mathbf{a} = (a_x, a_y)$, $\mathbf{b} = (b_x, b_y)$, *to* $\mathbf{ab} = a_x b_x + a_y b_y$,
2) *jeżeli* $\mathbf{a} = (a_x, a_y, a_z)$, $\mathbf{b} = (b_x, b_y, b_z)$, *to* $\mathbf{ab} = a_x b_x + a_y b_y + a_z b_z$

- Wektory są do siebie prostopadłe, gdy ab=0

Iloczyn wektorowy

DEFINICJA. *Niech wektory* $\mathbf{a}$, $\mathbf{b}$ *będą niezerowe i nierównoległe. Iloczynem wektorowym wektora* $\mathbf{a}$ *przez wektor* $\mathbf{b}$, *oznaczamy go przez* $\mathbf{a} \times \mathbf{b}$, *nazywamy taki wektor* $\mathbf{c} = \mathbf{a} \times \mathbf{b}$, *że:*

1) $|\mathbf{c}| = |\mathbf{a}|\,|\mathbf{b}| \sin \sphericalangle(\mathbf{a}, \mathbf{b})$,
2) $\mathbf{c} \perp \mathbf{a}$ *i* $\mathbf{c} \perp \mathbf{b}$ (*wektor* $\mathbf{c} = \mathbf{a} \times \mathbf{b}$ *jest prostopadły do wektora* $\mathbf{a}$ *i do wektora* $\mathbf{b}$),
3) *uporządkowana trójka* $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ *ma orientację zgodną z układem* $Oxyz$, *czyli* $W(\mathbf{a}, \mathbf{b}, \mathbf{c}) > 0$.



$$\mathbf{c} = \mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} =$$

$$= (a_y b_z - a_z b_y)\mathbf{i} + (a_z b_x - a_x b_z)\mathbf{j} + (a_x b_y - a_y b_x)\mathbf{k}$$

Równanie płaszczyzny

$$A(x-x_0)+B(y-y_0)+C(z-z_0)=0,$$

(A,B,C) jest wektorem prostopadłym do płaszczyzny, (x_0,y_0,z_0) jest punktem, przez który ta płaszczyzna przechodzi, a (x,y,z) jest dowolnym punktem leżącym na tej płaszczyźnie.

Weźmy pod uwagę trzy punkty nie leżące na jednej prostej

$$P_0(x_1,y_1,z_1), \qquad P_1(x_2,y_2,z_2), \qquad P_2(x_3,y_3,z_3).$$

Punkty te wyznaczają płaszczyznę. Dowodzi się, że płaszczyzna ta jest opisana równaniem

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0 \tag{7.40}$$

Układ równań

$$\begin{cases} A_1(x-x_1)+B_1(y-y_1)+C_1(z-z_1)=0 \\ A_2(x-x_2)+B_2(y-y_2)+C_2(z-z_2)=0 \end{cases}$$

nazywamy równaniem krawędziowym prostej, która powstaje w wyniku przecięcia się płaszczyzn π_1 i π_2.

Podstawowe algorytmy geometryczne

- Wzajemne położenie trzech punktów
- Niech będą dane trzy punkty P,Q,R
- Zbadanie położenia punktu R względem prostej PQ sprowadza się do policzenia wyznacznika macierzy

$$A = \begin{pmatrix} p_x & p_y & 1 \\ q_x & q_y & 1 \\ r_x & r_y & 1 \end{pmatrix}$$

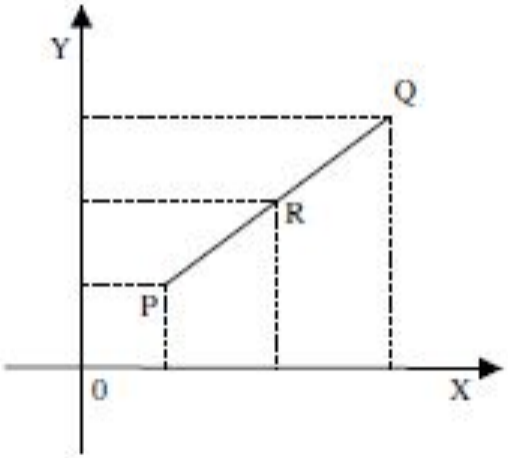
- Znak wyznacznika macierzy A jest równy znakowi sinusa kąta nachylenia wektora PR do wektora PQ.
- Jeżeli wyznacznik ten wynosi 0, to punkty P, Q i R są współliniowe.
- Punkt R leży po lewej stronie wektora PQ, jeżeli wyznacznik jest większy od 0 i odwrotnie punkt R leży po prawej stronie wektora PQ, gdy wyznacznik jest ujemny

Podstawowe algorytmy geometryczne

- Czy dwa punkty leżą po tej samej stronie prostej?
- Niech dana będzie prosta PQ i punkty P1 i P2
- Wystarczy sprawdzić, czy wyznaczniki macierzy **A** posiadają takie same znaki

Podstawowe algorytmy geometryczne

- Przynależność punktu do odcinka
- Spełnione muszą być nierównosci: min(px, qx)<= rx, rx<= max(px, qx), min(py, qy) <=ry i ry<= max(py, qy)
- Punkty P, Q i R muszą b
współliniowe.



Podstawowe algorytmy geometryczne

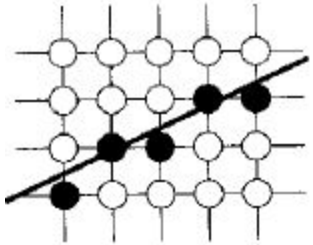
- <u>Przecinanie odcinków</u>
- Odcinki PQ i RS przecinają się wtedy i tylko wtedy, gdy punkty P i Q leżą po przeciwnych stronach prostej RS, a punkty R i S leżą po przeciwnych stronach prostej PQ lub któryś z końców jednego z odcinków należy do drugiego odcinka.

Podstawowe algorytmy rysownia prymitywów

- Prymityw – figura geometryczna, z której buduje się inne – bardziej skomplikowane
- Najczęściej używanymi prymitywami są:
  - Odcinki
  - Trójkąty
  - Krzywe
  - Okręgi, koła, sfery, a najczęściej - łuki
  - Prostokąty, kwadraty

Rysowanie odcinków

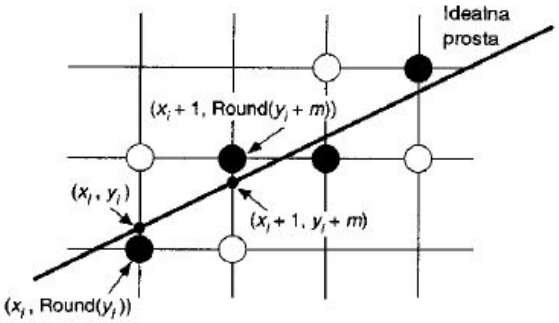
– Przedstawienie problemu:



– Jak najdokładniej przedstawić odcinek?
– Algorytm musi być bardzo wydajny - bardzo często używany

Algorytm przyrostowy

- Równanie:
  - Y = mx+b , dla każdego x
  - Wyświetlanie piksela (x_i,round(y_i))
- Problemy:
  - Mała efektywność
  - Wykonywane zmiennopozycyjne mnożenie, dodawani e



Algorytm przyrostowy c.d.

- Można zauważyć, że:

$$y_{i+1} = mx_{i+1} + B = m(x_i + \Delta x) + B = y_i + m\Delta x$$

i jeżeli $\Delta x = 1$, to $y_{i+1} = y_i + m$.

- Stąd otrzymujemy:

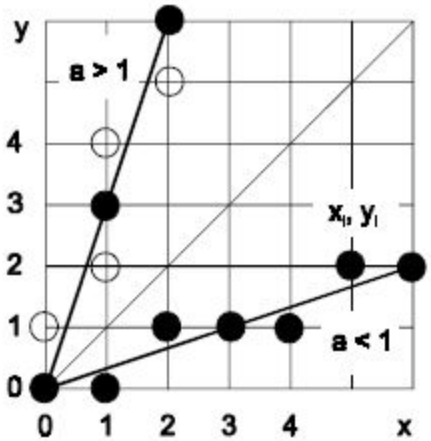
$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i + m;$$

Algorytm przyrostowy

- Nie jest potrzebna wartość B
- Wymagane są punkty początkowe i końcowe
- Dla |m|>1 przyrost y będzie większy niż 1, należy wtedy odwrócić x i y – powiększać y o 1 i wyliczać $\Delta x = \Delta y/m = 1/m$.
- Problem: stałe dodawanie przyrostu i zaokrąglanie powoduje kumulowanie się błędu

Algorytm przyrostowy



Jeżeli $a > 1$ analizujemy równanie

$$y = \frac{1}{a} x$$

i rysujemy punkty (y_i, x_i).

Algorytm przyrostowy

- Algorytm dla |m|<1
- Pominięto przypadek poziomy i pionowy

```
void   Line(int x0, int y0, int x1, int y1, int value)
{                          /* Zakłada się, że –1 ≤ m ≤ 1, x0 < x1 */
    int    x;              /* x zmienia się od x0 do x1 z przyrostem jednostkowym */
    float  dy, dx, y, m;

    dy = y1 – y0;
    dx = x1 – x0;
    m = dy / dx;
    y = y0;
    for (x = x0; x <= x1; x++) {
        WritePixel(x, (int) floor(y + 0.5), value);     /* Ustawienie wartości piksela */
        y += m;                                         /* Zwiększenie y o nachylenie m */
    }
}
```

Algorytm z punktem środkowym

- Wadą algorytmu przyrostowego jest:
  - operowanie na zmiennopozycyjnym m
  - operacja zaokrąglania
- Zalety algorytmu z punktem środkowym
  - Operuje na liczbach całkowitych
  - Nie używa operacji zaokrąglania
- Algorytm został opracowany przez Bresenhama

Ułamki w innych systemach

- mnożymy liczbę przez podstawę systemu
- jako nową liczbę pod spodem zapisujemy część ułamkową otrzymanego iloczynu (0.cośtam), natomiast część całkowitą (to, co w wyniku otrzymanym po pomnożeniu stało przed przecinkiem) zapisujemy po prawej stronie.

Przykład

- Liczba 0,625 w systemie binarnym
- 0,625* 2=1,25 | 1
- 0,25*2=0,5 | 0
- 0,5*2=1 | 1
- Obliczenia kończą się w przypadku otrzymania liczby całkowitej
- Podczas kodowania ułamków otrzymane cyfry spisujemy, odwrotnie niż w przypadku liczb całkowitych, **od góry do dołu!**
- Często należy brać wynik w przybliżeniu

Liczby rzeczywiste - przykład

1984.0415 = 11111000000.00001010101

Normalizacja:

1.1111000000000001010101

Mantysa (przyjmujemy określoną długość):

m = 1111000000

Cecha (liczba miejsc o które przesunęliśmy przecinek w kodzie uzupełnień do dwóch)

c = (01010)U2

Liczby rzeczywiste - przykład

Liczba po przeliczeniu

1984.0415 = (0 01010 1111000000)FP2

0 – znak

01010 – cecha

1111000000 - mantysa

Dekodowanie

s = 0, a więc mamy: $(-1)^0$

m = 1111000000, a więc mamy: $2^{-1} + 2^{-2} + 2^{-3} + 2^{-4}$

c = $(01010)_{U2}$ = 10, a więc mamy 2^{10}

- <u>Składając wszystko razem:</u>

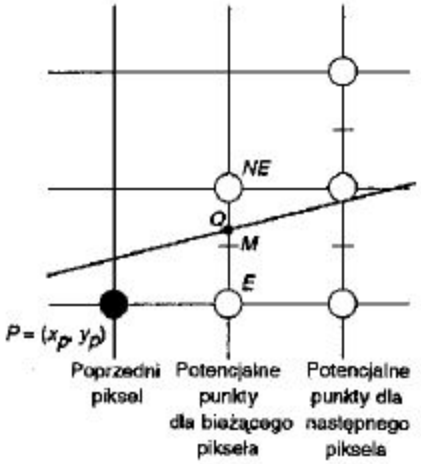
$$(-1)^0 * (\mathbf{2^0} + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4}) * 2^{10} =$$

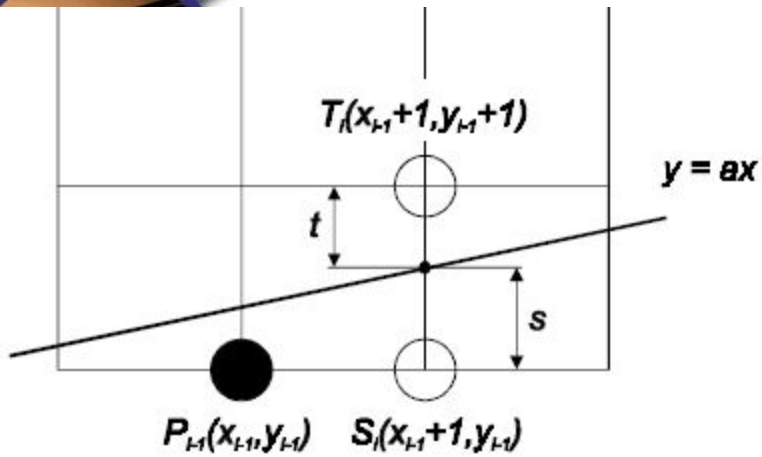
$$2^{10} + 2^9 + 2^8 + 2^7 + 2^6 = 1024 + 512 + 256 + 128 + 64 = 1984$$

- <u>Zakodowanie 10 bitami spowodowało „obcięcie”</u>

Algorytm z punktem środkowym

- Niech P będzie punktem początkowym
- W następnym kroku do wyboru są dwa punkty: NE i E
- Punkt Q leży na przecięciu prostej $x_{i+1}=x+1$
- Obliczana jest różnica odległości między E i Q i NE i Q
- Nowym punktem będzie punkt o mniejszej odległości





Z rysunku

$$s = \frac{dy}{dx}(x_{i-1}+1) - y_{i-1}$$

$$t = y_{i-1} + 1 - \frac{dy}{dx}(x_{i-1}+1)$$

a więc,

$$s - t = 2\frac{dy}{dx}(x_{i-1}+1) - 2y_{i-1} - 1$$

Inaczej kryterium wyboru punktu wyraża się jako

$$dx(s-t) = 2(x_{i-1}dy - y_{i-1}dx) + 2dy - dx.$$

Po podstawieniu $d_i = dx(s-t)$ uzyskuje się

$$d_i = 2x_{i-1}dy - 2y_{i-1}dx + 2dy - dx. \qquad (1)$$

Zwiększając w formalny sposób indeksy $(i+1 \to i)$ otrzymuje się

$$d_{i+1} = 2x_i dy - 2y_i dx + 2dy - dx \qquad (2)$$

Odejmując stronami od równania (2) równanie (1) uzyskuje się związek między wartościami kryterium wyboru dla i - tego i i+1- go kroku

$$d_{i+1} - d_i = 2dy(x_i - x_{i-1}) - 2dx(y_i - y_{i-1})$$

Wiadomo, że $x_i - x_{i-1} = 1$, więc

$$d_{i+1} = d_i + 2dy - 2dx(y_i - y_{i-1}) \qquad (3)$$

1. Jeżeli $d_i \geq 0$ $(s \geq t)$ to wybiera się T_i, wtedy $y_i = y_{i-1} + 1$ oraz

$$\boxed{d_{i+1} = d_i + 2(dy - dx)}$$

2. Jeżeli $d_i < 0$ $(s < t)$ to wybiera się S_i, wtedy $y_i = y_{i-1}$ oraz

$$\boxed{d_{i+1} = d_i + 2dy}$$

3. Podstawiając do (1) $(x_0, y_0) = (0,0)$, uzyskuje się kryterium wyboru dla kroku 1

$$\boxed{d_1 = 2dy - dx}$$

Algorytm z punktem środkowym

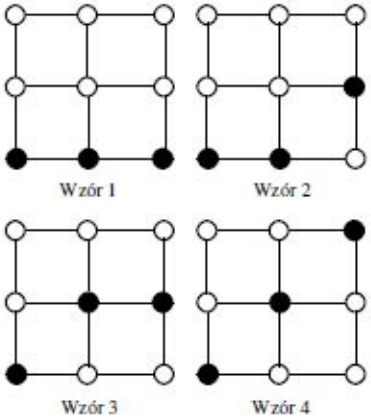
```
void  MidpointLine(int x0, int y0, int x1, int y1, int value)
{
   int   dx, dy, incrE, incrNE, d, x, y;

   dx = x1 – x0;
   dy = y1 – y0;
   d = dy * 2 – dx;              /* Początkowa wartość d */
   incrE = dy * 2;               /* Przyrost używany dla przejścia do E */
   incrNE = (dy – dx) * 2;           /* Przyrost używany dla przejścia do NE */
   x = x0;
   y = y0;
   WritePixel(x, y, value);          /* Piksel początkowy */
   while (x < x1) {
      If (d <= 0) {                  /* Wybór E */
         d += incrE;
         x++;
      } else {                       /* Wybór NE */
         d += incrNE;
         x++;
         y++;
      }
      WritePixel(x, y, value);       /* Wybrany najbliższy piksel */
   }
}
```

Algorytm Wu-Rokne'a

- Algorytm podwójnego kroku
- W każdym kroku wybór nie jednego, a dwóch pikseli
- Współczynnik kierunk $0 < \frac{dy}{dx} = \frac{y_k - y_0}{x_k - x_0} \leqslant 1$
- W celu ustalenia, do ktorego przedziału wielkosci nalezy współczynnik kierunkowy prostej wystarczy obliczyc wartosc wyrazenia: 4dy – dx.
- Wartosc ujemna oznacza, ze współczynnik jest mniejszy niz 1/2 (wzory 1, 2 i 3), w przeciwnym wypadku stosowane beda wzory 2, 3 i 4
- W przypadku, gdy współczynnik jest mniejszy od 1/2 , wa
- początkowa zmiennej decyzyjnej wynosi d0 = 4dy – dx.

$$\begin{cases} wzór\ 1: & d_i < 0 & d_{i+i} = d_i + 4dy \\ wzór\ 2: & 0 \leqslant d_i < 2dy & d_{i+1} = d_i + 4dy - 2dx \\ wzór\ 3: & 2dy \leqslant d_i & d_{i+1} = d_i + 4dy - 2dx \end{cases}$$



- Jeżeli współczynnik kierunkowy prostej jest większy lub r
- 2 , wartość początkowa zmiennej decyzyjnej wynosi: d0 =

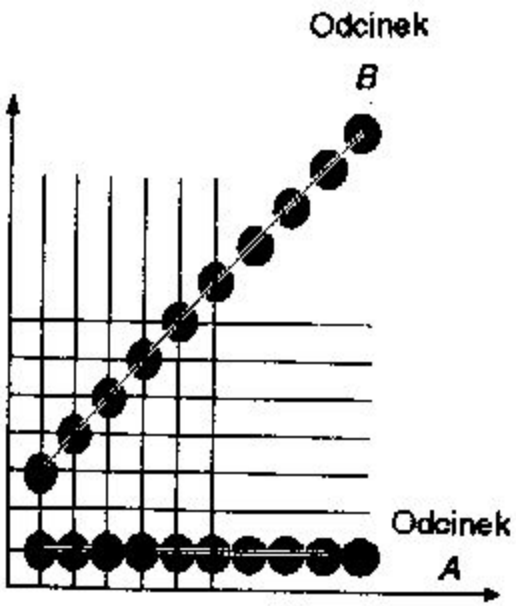
$$\begin{cases} wzór\ 4: & d_i < 0 & d_{i+i} = d_i + 4(dy - dx) \\ wzór\ 2: & 0 \leqslant d_i < 2(dy - dx) & d_{i+1} = d_i + 4dy - 2dx \\ wzór\ 3: & 2(dy - dx) \leqslant d_i & d_{i+1} = d_i + 4dy - 2dx \end{cases}$$

Algorytm EFLA

- (ang. *Extremely Fast Line Algorithm*)
- 1. $v = 32.768 + 65.536y0$,
- 2. $i = 65.536dy/dx$
- 3. $x = x0$,
- 4. piksel(x, $v/65.536$)
- 5. $v = v + i$
- 6. $x = x + 1$
- 7. powtarzaj 4 - 6 dopóki x ~= xk.

Problemy związane z rysowaniem odcinków

- Problem z identycznością odcinków z podanymi w odwrotnej kolejności punktami końcowymi
- Zmiana jasności odcinka w funkcji nachylenia
  - Jeżeli jasność piksela jest I na jednostkę długości wyno $I/\sqrt{2}$ a dla odcinka B tylko



Rysowanie łuków i okręgów

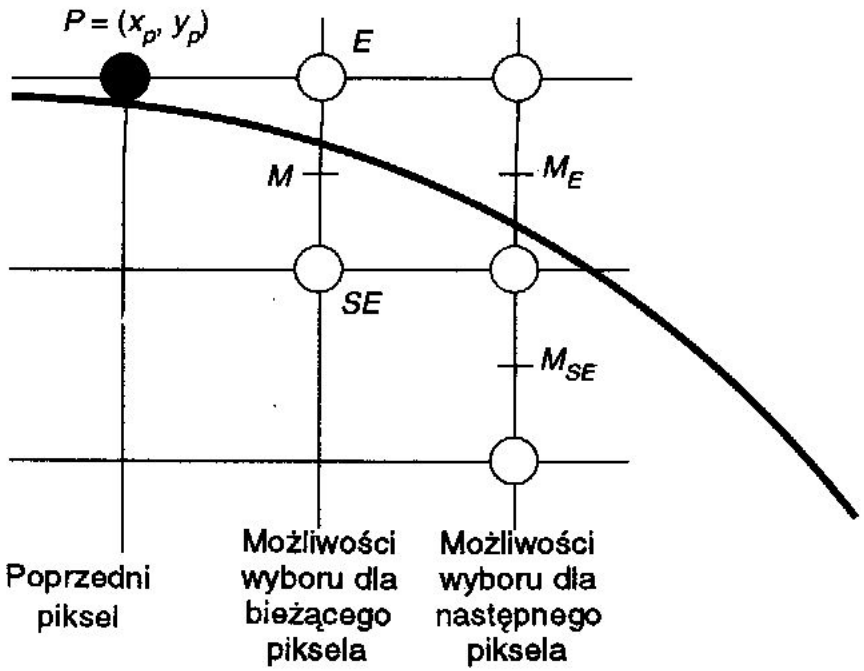
- Rówanie okręgu $x^2 + y^2 = R^2$.

$$y = \pm\sqrt{R^2 - x^2}$$

- W celu narysowania ćwiartki okręgu, zwiększamy x od 0 do R. Inne ćwiartki rysujemy na zasadzie symetrii.
- Metoda nieefektywna – mnożenie, pierwiastkowanie, zaokrąglanie

Rysowanie łuków i okręgów

- Algorytm z punktem środkowym

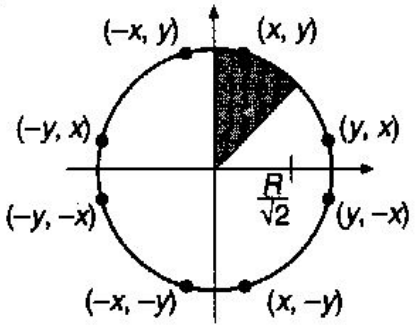


Algorytm z punktem środkowym

```
void   MidpointCircle(int radius, int value)
{
    int    x, y, d;

    x = 0;                          /* Inicjalizacja */
    y = radius;
    d = 1 – radius;
    CirclePoints(x, y, value);
    while (y > x) {
        If (d < 0) {                /* Wybrać E */
            d += x * 2 + 3;
            x++;
        } else {                    /* Wybrać SE */
            d += (x – y) * 2 + 5;
            x++;
            y—;
        }
        CirclePoints(x, y, value);
    }
}
```

```
void CirclePoints (float x, float y, int value);
{
    WritePixel (x, y, value);
    WritePixel (y, x, value);
    WritePixel (y, –x, value);
    WritePixel (x, –y, value);
    WritePixel (–x, –y, value);
    WritePixel (–y, –x, value);
    WritePixel (–y, x, value);
    WritePixel (–x, y, value);
}
```



Wypełnianie obszarów

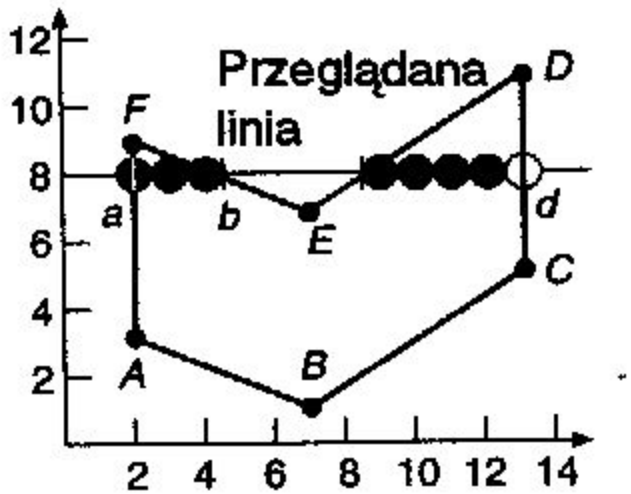
- Wypełnianie obszaru jest drugim po rysowaniu odcinka lub łuku, najczęściej występującym problemem związanym z prymitywami
- Wype

```
for (y od ymin do ymaks prostokąta){      /* Dla linii */
    for (x od xmin do xmaks){      /* Dla piksela wewnętrznego */
        WritePixel(x, y, value);
    }
}
```

zadan czy podać x_{min}, x_{max} 1 y_{min} 1 y_{max}

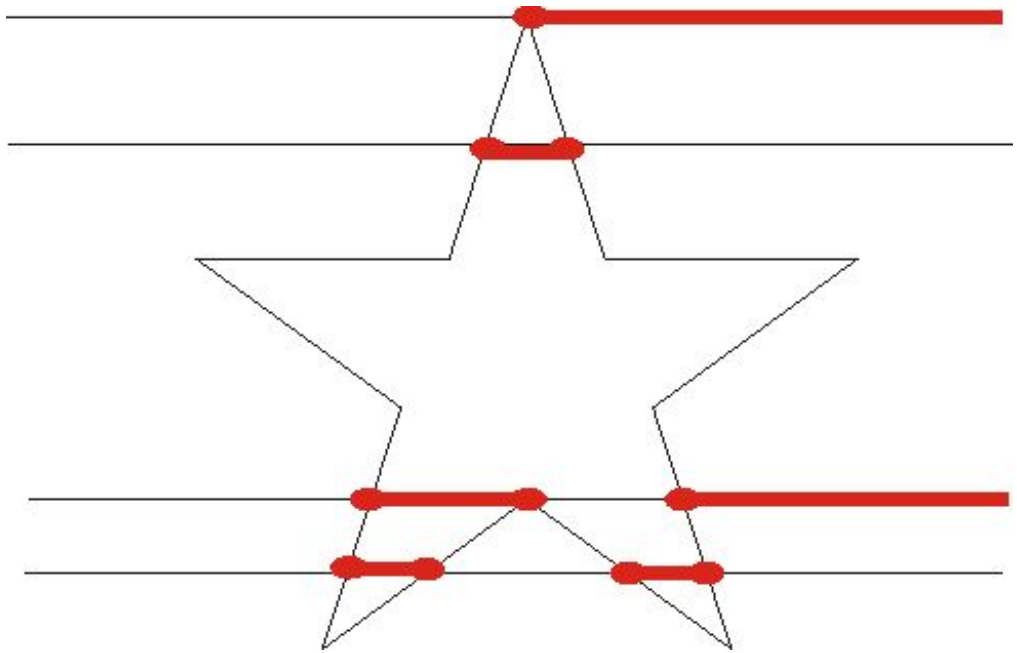
Wypełnianie wielokątów

- Algorytm wypełnia obszar między lewym a prawym końcem odcinka

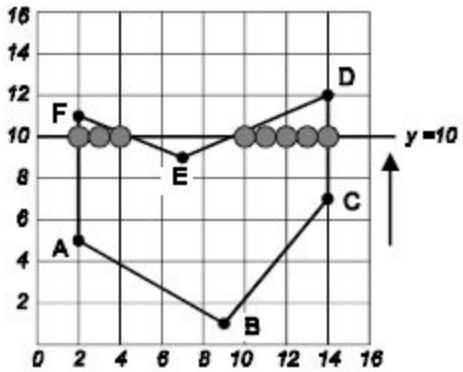


Wypełnianie przez kontrolę parzystości

- Problem z ekstremami



Algorytm skanowania linii



Opis wieloboku:

Krawędź	y_{min}	y_{max}	x_{min}	x_{max}
AB	1	5	9	2
BC	1	7	9	14
CD	7	12	14	14
DE	9	12	7	14
EF	9	11	7	2
FA	5	11	2	2

Algorytm skanowania linii

Krok 0

Utworzyć globalną tablicę krawędzi (ET).

Krok 1

Ustawić y na najmniejszej wartości współrzędnej y z globalnej tablicy krawędzi (ET), czyli y dla pierwszej niepustej grupy krawędzi.

Krok 2

Wyzerować aktywną tablicę krawędzi (AT).

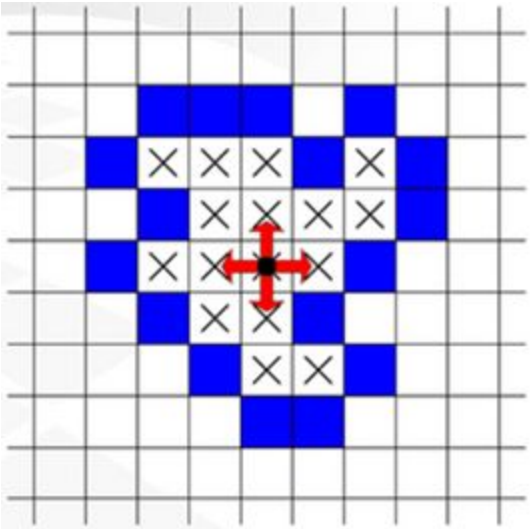
Krok 3

Powtarzać tak długo, dopóki tablica globalna (ET) i tablica aktywna (AT) nie będą puste.

- Przenieść z grupy y tablicy globalnej (ET) do tablicy aktywnej (AT) te krawędzie, dla których $y_{min} = y$ i posortować je ze względu na x.
- Wypełnić piksele w linii y, wykorzystując pary x z tablicy aktywnej (AT).
- Usunąć z tablicy aktywnej (AT) te krawędzie, dla których $y = y_{max}$.
- Zwiększyć y o 1 (następna linia).
- Dla każdej pary krawędzi, która nie jest pionowa wyliczyć i wstawić do tablicy aktywnej (AT) nowe wartości x.

Wypełnianie przez spójność

- Należy zdefiniować siatkę – 4 czy 8 spójną
- Należy zdefiniować punkt startowy – ziarno, leżący wewnątrz wielokąta



Rekurencyjny algorytm powodziowy

- zakłada sprawdzanie koloru każdego z czterech sąsiadów piksela startowego
- dalej postępujemy tak samo badając kolor pikseli sąsiadujących z sąsiadami piksela startowego itd.
- rozrzutność algorytmu objawiająca się wielokrotnym badaniem koloru tego samego piksela

Wypełnianie przez spójność

Przyjęto: c_b – barwa brzegu, c_f – barwa wypełnienia

```
procedure wypełnij1(x,y)
    begin
      set_pixel(x,y,c_f);
      if (barwa(x-1,y) inna niż c_ b i inna niż c_f) wypełnij1(x-1,y);
      if (barwa(x+1,y) inna niż c_ b i inna niż c_f) wypełnij1(x+1,y);
      if (barwa(x,y-1) inna niż c_ b i inna niż c_f) wypełnij1(x,y-1);
      if (barwa(x,y+1) inna niż c_ b i inna niż c_f) wypełnij1(x,y+1);
    end
```

Algorytm Smitha

- W algorytmie Smitha obszar wypełniany jest liniami poziomymi w nastepujacy sposób:

— zrzuć współrzędne piksela startowego (x, y) na stos,

— dopóki stos nie jest pusty powtarzaj:

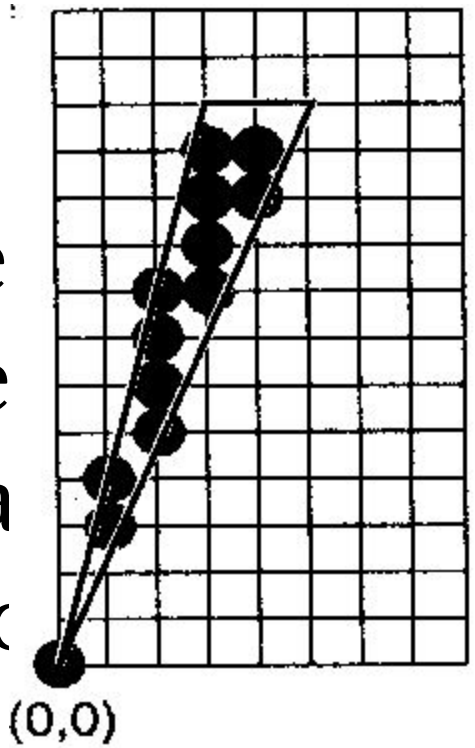
— pobierz współrzędne punktu ze stosu,

— zrzuć na stos współrzędne punktów leżących nad i pod punktem bieżącym, jeżeli ich kolor jest różny od koloru brzegu i koloru wypełnienia; sprowadza się to do obserwacji sytuacji pod i nad rysowaną linią - punkt powinien być zrzucany tylko jeden raz przy zmianie koloru nad (pod) linią, np. podczas „wyjścia” spod pikseli brzegowych,

— wypełnij obszar w lewo (prawo), aż do napotkania piksela brzegowego (czyli o kolorze brzegu lub wypełnienia)

Drzazgi

- Wielokąty o krawędziach leżących bardzo blisko siebie
- Należy spróbkować i wypełnić drzazgę z wię rozdzielczością, a nastę uśrednić barwę/lumina wracając do rozdzielczo rastra



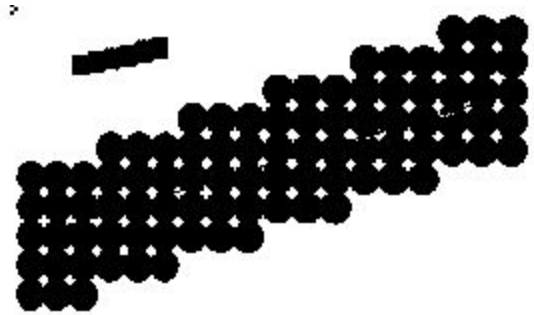
Pogrubianie

- Najprostsze rozwiązanie:
- Umieszczamy środek pędzla w każdym pikselu konturu i malujemy otoczenie
- Wiele problemów:
  - Jaki kształt ma pędzel?
  - Jaka jest orientacja pędzla nieokrągłego?
  - Jak malować pędzlem prostokątnym (jaka orientacja) ?
  - Co się dzieje na wierzchołkach wielokąta?

Pogrubianie

- Metoda powielania kolumn
- Dla pochyleń z zakresu

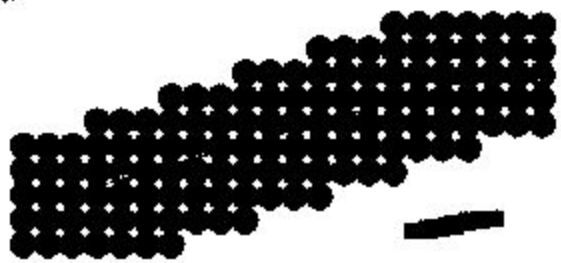
-1 do 1 powielane są kolumn



- Dla pozostałych wiersze
- Niestety, zawsze końce odcinków będą pionowe lub poziome
- Dla odcinków poziomych i pionowych grubość t będzie inna dla odcinków nachylonych np. pod kątem 45 stopni – t / pier(2)

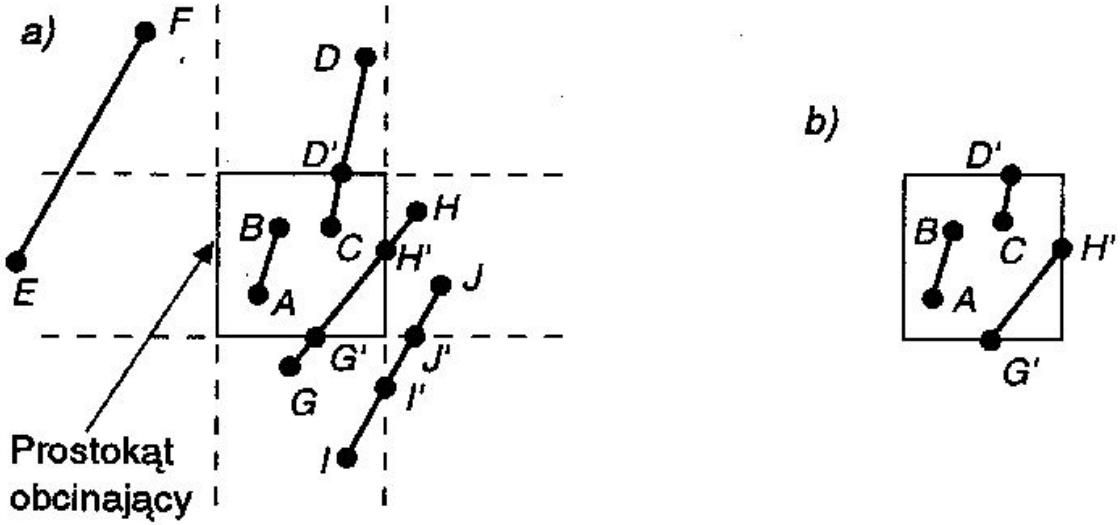
Pogrubianie - metoda ruchomego pióra

- Metoda ruchomego pióra
- Prostokątne pióro porusza się wzdłuż jednopikselowego

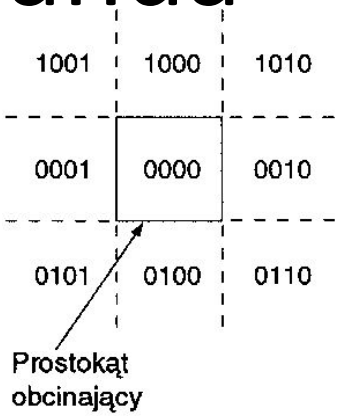


Obcinanie

- Przedstawienie rysunku na ekranie wymaga określenia fragmentu, który będzie obrazowany

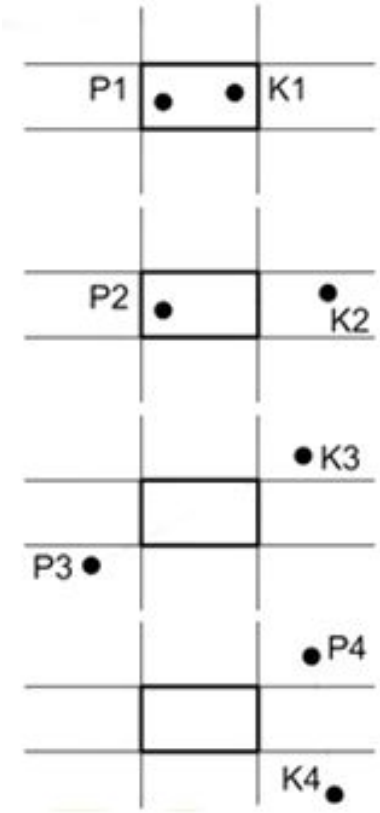


Algorytm Cohena-Sutherlanda



- Służy do obcinania odcinków do prostokątnego okna
- Działa na podstawie analizy punktów końcowych
- Dzieli płaszczyznę na 9 obszarów
- Krawędzie okna wyznaczają cztery proste: prawą, lewą, górna i dolną
- Kolejne bity kodu określają poziome i pionowe pasy
- Operacja AND przeprowadzona na kodach końców odcinka pozwala odrzucić te odcinki, które na pewno są poza oknem. Spośród pozostałych odcinków należy wybrać te, które rzeczywiście mają wspólne punkty z oknem oraz przyciąć do jego rozmiaru.
- Jeśli wynik operacji AND jest różny od zera – należy odrzucić odcinek jako niemający na pewno punktów wspólnych z oknem.
- Jeśli wynik operacji AND jest zerowy – odcinek może przecinać okno. W takiej sytuacji należy rozważyć przypadek szczególny gdy kody obu końców są zerowe (punkty P1 i K1 na rysunku), wtedy cały rysunek leży wewnątrz okna.
- Natomiast jeśli kody końców są niezerowe to określają one którymi prostymi należy przyciąć odcinek.

Algorytm Cohena-Sutherlanda



P1 = 0000
K1 = 0000
P1 AND K1 = 0000

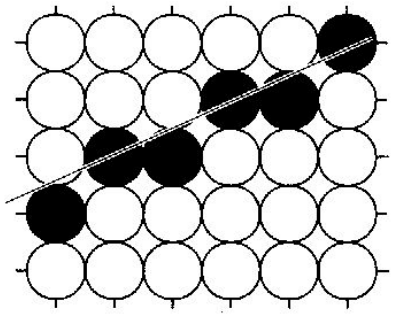
P2 = 0000
K2 = 0010
P2 AND K2 = 0000

P3 = 0101
K3 = 1010
P3 AND K3 = 0000

P4 = 1010
K4 = 0110
P4 AND K4 = 0010

Usuwanie zakłóceń

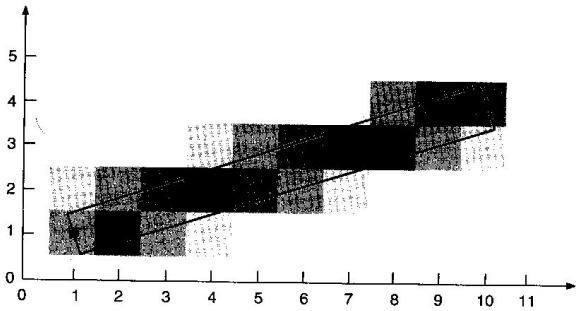
- Przy rysowaniu metodą zapal piksel lub nie łatwo dostrzec „zębate” kształty prymitywów
- Dla odcinków poziomych i



Pionowych można stosować jed

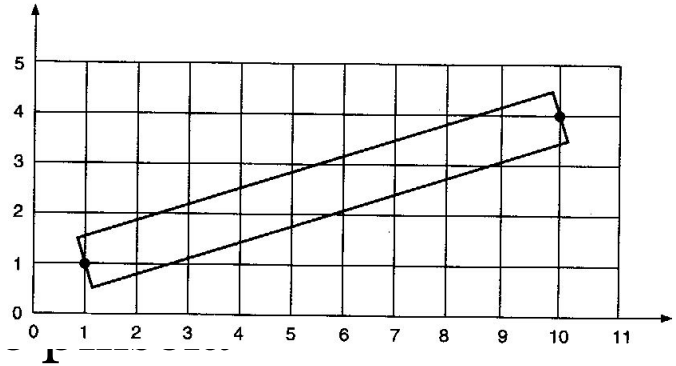
piksel

- Dla pozostałych ej dwa o różnych jasnoś



Usuwanie zakłóceń

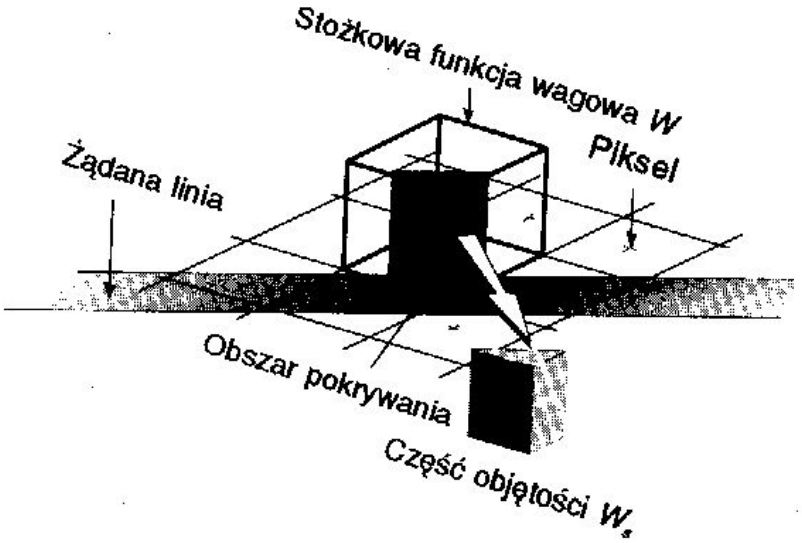
- Metoda dodawania pikseli o różnych jasnościach proporcjonalnych do zajmowanej powierzchni nosi nazwę <u>bezwagowego próbkowania powie</u>
- Jasność piksela przeciętego przez odcinka zmniejsza się w funkcji od od środka piksela – im prymityw tym ma mniejszy wpływ na jasnoś
- Prymityw nie może wpływać na jasność piksela, jeżeli nie przecina on kwadratu reprezentującego piksel
- Równe pola wnoszą równe jasności, niezależnie od odległości

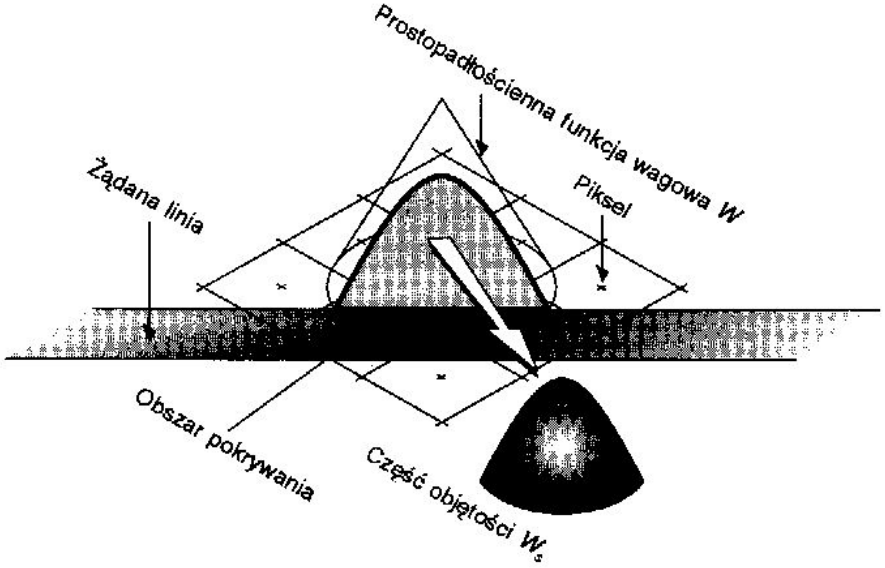


Usuwanie zakłóceń

- Wagowe próbkowanie powierzchni
- Jasność piksela przeciętego przez krawędź odcinka zmniejsza się w funkcji odległości od środka piksela – im prymityw jest dalej tym ma mniejszy wpływ na jasność piksela
- Prymityw nie może wpływać na jasność piksela, jeżeli nie przecina on kwadratu reprezentującego piksel
- Udział takich samych powierzchni nie jest taki sam – mała powierzchnia blisko środka ma większy wpływ niż powierzchnia znajdująca się w większej odległości

Porówanie metod



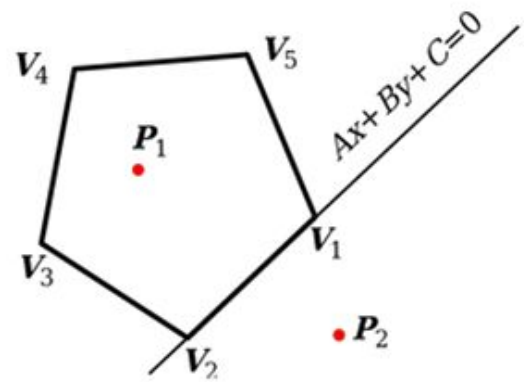


Przynależność punktu do wielokąta

- Prosta wyznaczona przez dwa kolejne wierzchołki wielokąta dzieli płaszczyznę na dwie półpłaszczyzny, z których tylko jedna zawiera wielokąt.
- Podstawiając współrzędne badanego punktu do równania prostej można na podstawie znaku równania stwierdzić po której stronie prostej punkt się znajduje. Punkt P1 będzie po tej samej stronie prostej wyznaczonej przez V1 V2 co np. wierzchołek V4. Natomiast punkty P2 i V4 będą po przeciwnych stronach tej prostej.
- Algorytm sprawdzania przynależności punktu do wnętrza wielokąta wypukłego:

1. Obejść wielokąt zgodnie z porządkiem kolejnych jego wierzchołków.
2. W każdym kroku wyznaczyć :prostą przechodzącą przez bieżący i następny wierzchołek.
3. Sprawdzić czy badany punkt jest po tej samej stronie prostej co jeden z wierzchołków wielokąta.
4. Jeśli w którymkolwiek kroku badany punkt nie spełnia tego warunku, przerwać sprawdzanie – punkt jest na zewnątrz.
5. Jeśli po obejściu całego wielokąta okaże się, że punkt był zawsze (!) po tej samej stronie co reszta wielokąta to punkt jest wewnątrz.

Dla wielokąta wypukłego



Przynależność punktu do wielokąta

- Jeśli liczba przecięć z bokami wielokąta jest nieparzysta to punkt leży wewnątrz wielokąta (punkt P1 na rysunku).
- Jeśli liczba przecięć jest parzysta, to punkt jest na zewnątrz.
- Algorytm jest bardzo prosty, jednak pamiętając o problemach ze szczególnymi przypadkami w algorytmie wypełniania przez kontrolę parzystości, należy przypuszczać, że podobne sytuacje wystąpią i tutaj.
- Do algorytmu należy dodać dwa warunki:
- Jeśli punkt przecięcia jest ekstremum lokalnym, to liczymy go podwójnie.
- Jeśli półprosta zawiera cały bok wielokąta, to traktujemy to jako jeden pseudo-wierzchołek