Grafika komputerowa
Wykład 2
Algorytmy rastrowe

Romuald Kotowski

Instytut Informatyki i Automatyki

Państwowa Wyższa Szkoła Informatyki i Przedsiębiorczości w Łomży

2009

Spis treści

Spis treści

Spis treści

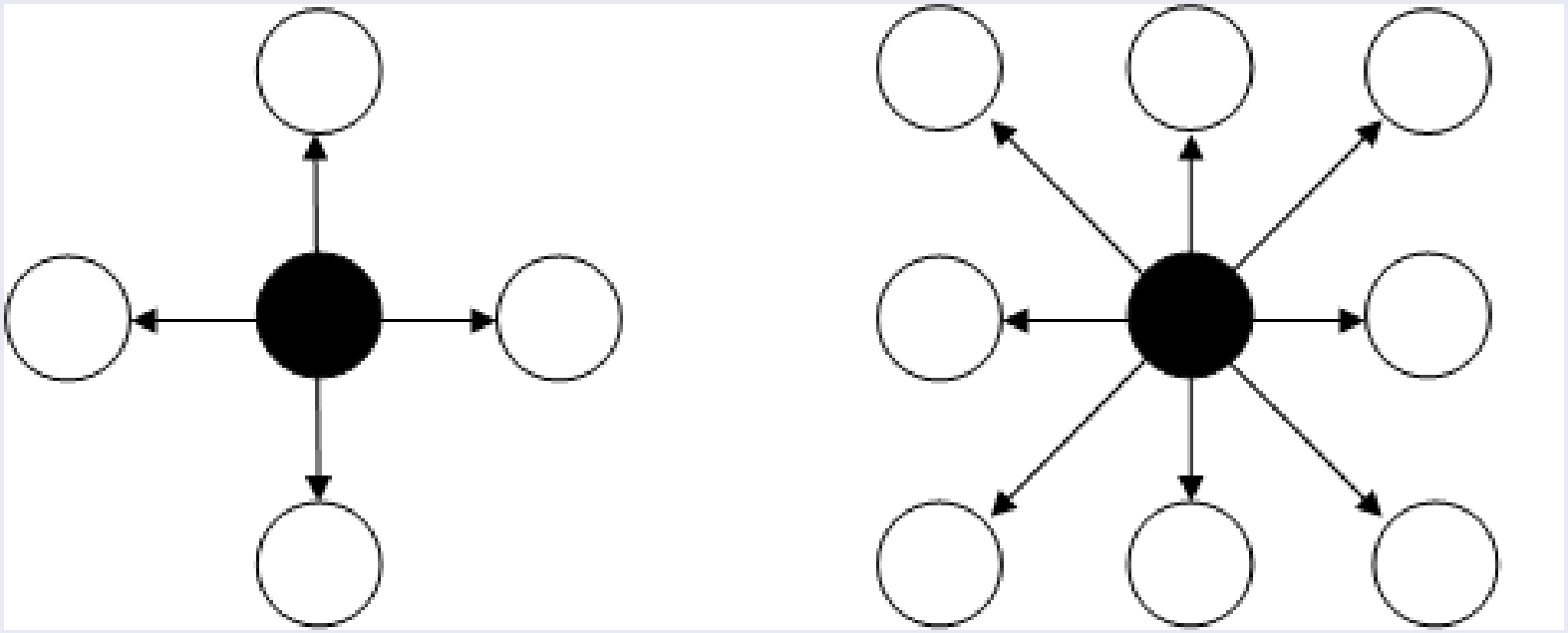
Algorytm Bresenhama

Rysowanie odcinków

Rysowanie odbywa się w **pikselowym układzie współrzędnych**, czyli przyjmujemy, że odległości pomiedzy pikselami wynoszą 1. Odcinek rysujemy pomiędzy współrzędnymi (x_0, y_0) a (x_k, y_k).

Algorytm Bresenhama

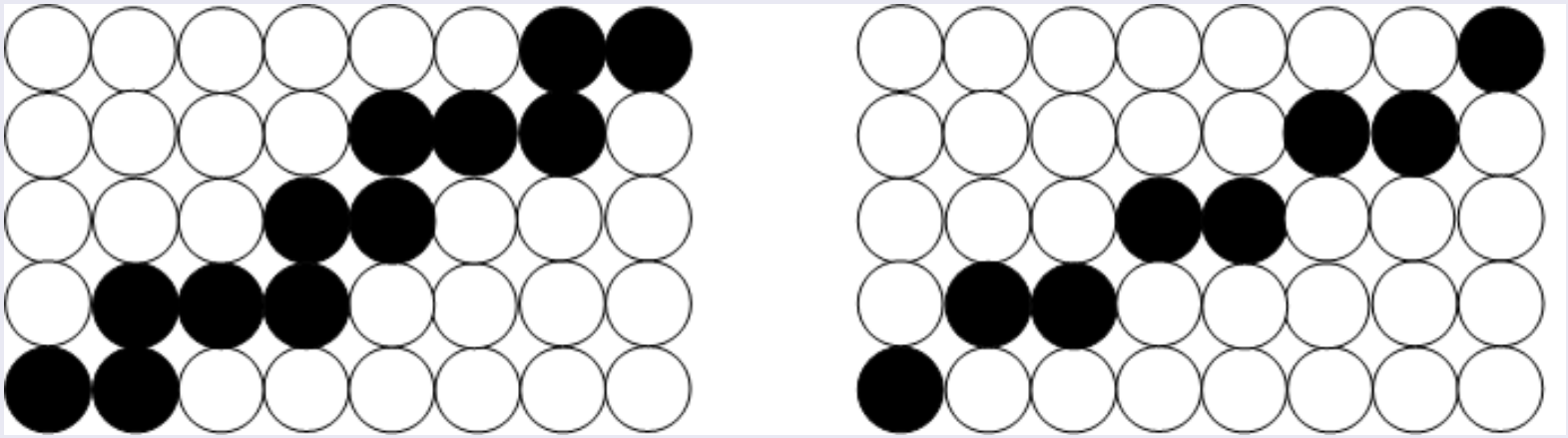
Rysowanie odcinków



Rys. 1: Cztero- i ośmiokierunkowy wybór pikseli

Algorytm Bresenhama

Rysowanie odcinków



Rys. 2: Wybory pikseli przybliżających odcinek z wyborem cztero- i ośmiokierunkowym

„Schodkowość" (kanciastość) możemy osłabić zmieniając jasność wybranych pikseli.
Antyaliasing – likwidowanie (wyrównywanie) kanciastości.

Algorytm Bresenhama

Rysowanie odcinków

Założenia: $x_0 < x_k$, współczynnik kierunkowy $0 < dy/dx \leq 1$, gdzie

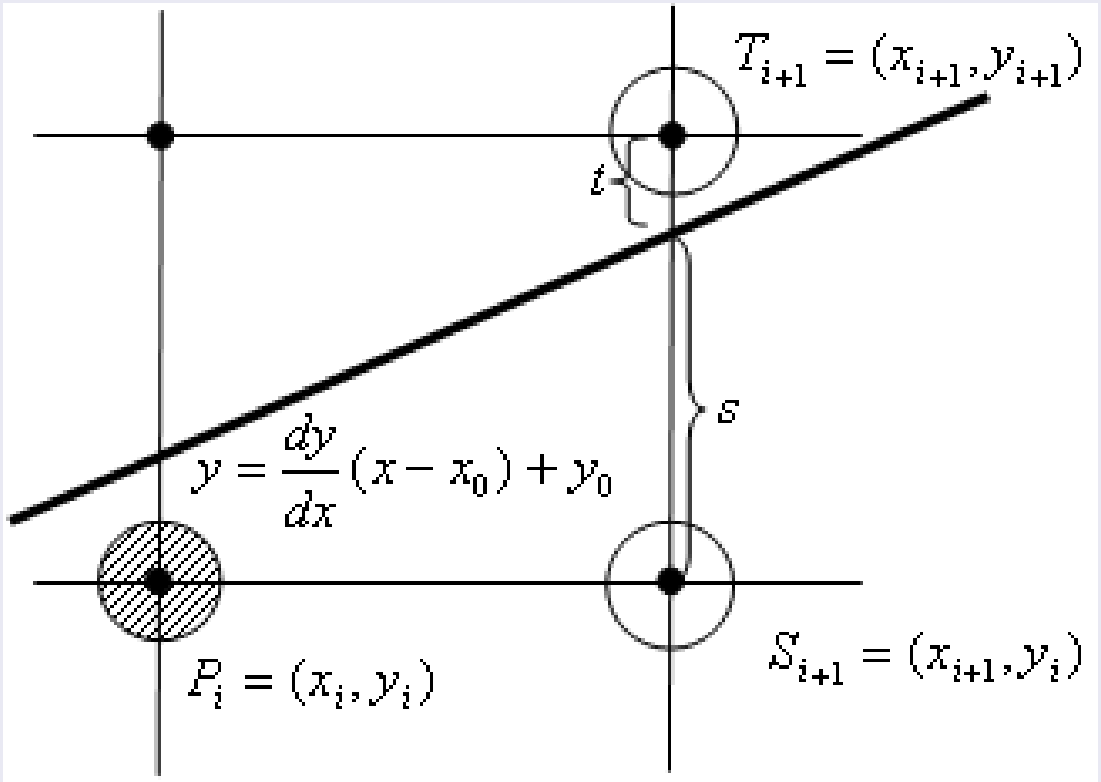
$$dy = y_k - y_0$$

$$dx = x_k - x_0$$

Zaczynamy rysować od piksela $P_0 = (x_0, y_0)$.

Algorytm Bresenhama

Rysowanie odcinków



Rys. 3: Wybór pomiędzy pikselami S_{i+1} a T_{i+1}

Algorytm Bresenhama

Rysowanie odcinków

Kąt nachylenia odcinka: $\alpha \in (0°, 45°)$, więc musimy wybrać pomiędzy pikselem poziomym S_{i+1} a na skos T_{i+1}.
Odległości od nowych pikseli:

$$s = \frac{dy}{dx}(x_i + 1 - x_0) - (y_i - y_0)$$

$$t = (y_i + 1 - y_0) - \frac{dy}{dx}(x_{i+1} - x_0)$$

Sprawdzamy, co jest większe: s czy t? Określa to znak różnicy $(s - t)$. Odejmujemy więc stronami i mnożymy przez dx

$$d_i = dx(s - t) = 2dy(x_i - x_0) - 2dx(y_i - y_0) + 2dy - dx$$

Algorytm Bresenhama

Algorytm Bresenhama rysowania odcinków

Dla $i+1$

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

czyli

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

więc

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

$(x_{i+1} - x_i = 1)$

Algorytm Bresenhama

Algorytm Bresenhama rysowania odcinków

Jeśli $d_i \geq 0 \hookrightarrow P_{i+1} = T_{i+1}, \quad y_{i+1} = y_i + 1$

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

Jeśli $d_i < 0 \hookrightarrow P_{i+1} = S_{i+1}, \quad y_{i+1} = y_i$

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

skąd dla $i = 0$ mamy $d_0 = 2dy - dx \leftarrow$ startowa zmienna decyzyjna

Algorytm Bresenhama

Algorytm Bresenhama rysowania odcinków

Start: $P_0 = (x_0, y_0)$; $dx = x_k - x_0$; $dy = y_k - y_0$;
$dp = 2 * dy$; $dd = 2 * (dy - dx)$;
$d_0 = 2 * dy - dx$;

dla $i = 0, 1, \ldots, x_k - x_0 - 1$

$x_{i+1} = x_1 + 1$;
jeśli $d_i \geq 0$, to $\{d_{i+1} = d_i + dd$; $y_{i+1} = y_i + 1\}$
w przeciwnym razie $\{d_{i+1} = d_i + dp$;
$y_{i+1} = y_i\}$
$P_{i+1} = (x_{i+1}, y_{i+1})$

Algorytm Bresenhama

Rysowanie odcinków

Oznaczenia:

- S – ruch poziomy
- T – ruch po przekątnej

Przykład: narysuj odcinek $(0,0) \rightarrow (131,16)$
Rozwiązanie: $S^4T(S^7T)^4S^8(S^7T)^5S^8T(S^7T)^4S^4$
Ta notacja ułatwia kompresję, skalowanie rysunków, obcinanie do zadanych kształtów, rysowanie wykresów z usuwaniem zasłoniętych linii.

Rysowanie okręgu

Rysowanie okręgu

Przy rysowaniu okręgu musimy uwzględnić aspekt urządzenia graficznego (równy stosunkowi odległości środków sąsiednich pikseli w poziomie do odległości środków sąsiednich pikseli w pionie). Zakładamy, że $a = p/q$ jest liczbą wymierną, p, q – liczby naturalne.

Wprowadzamy dwa układy współrzędnych

- $0xy$ – układ współrzędnych pikselowych
- $0XY$ – układ współrzędnych rzeczywistych

Rysowanie okręgu

Rysowanie okręgu – złożenia

Założenie: R – promień okręgu jest liczba naturalną i leży w środku układu współrzędnych.

Zadanie: Wybrać piksele przybliżające krzywą

$$X^2 + Y^2 - R^2 = 0$$

(zapis w układzie współrzędnych rzeczywistych) lub

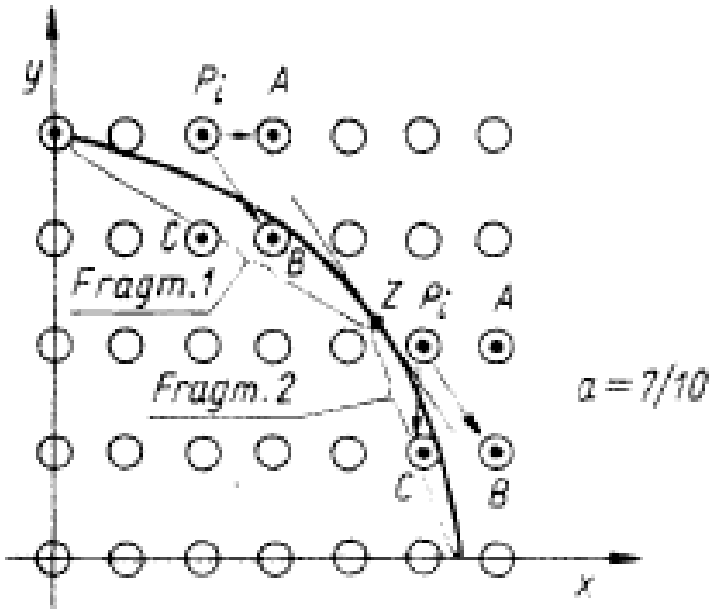
$$(ax)^2 + y^2 - R^2 = 0$$

(zapis w układzie współrzędnych pikselowych), czyli

$$f(x, y) = p^2x^2 + q^2y^2 - q^2R^2 = 0$$

Rysowanie okręgu

Symetria – ograniczamy się do jednej ćwiartki. Rozpoczynamy od $P_0 = (0, R)$, w kierunku zgodnym z ruchem wskazówek zegara.



Rys. 4: Podział ćwiartki okręgu na fragmenty w algorytmie Bresenhama

Rysowanie okręgu

W punkcie Z (dzieli ćwiartkę na dwie części) współczynnik kierunkowy

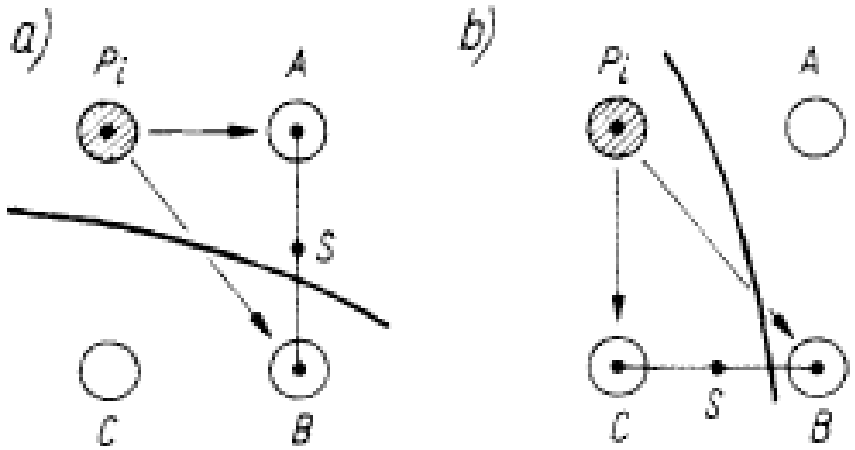
$$\frac{dy}{dx} = -\frac{f_x}{f_y} = -\frac{2p^2x}{2q^2y} = -1$$

W górnej ósemce ($p^2x < q^2y$) zwiększając x o 1 wybieramy P_{i+1} spośród A i B

W dolnej ósemce ($p^2x < q^2y$) zmniejszamy y o 1 wybieramy P_{i+1} spośród B i C

Jak wybierać?

Rysowanie okręgu

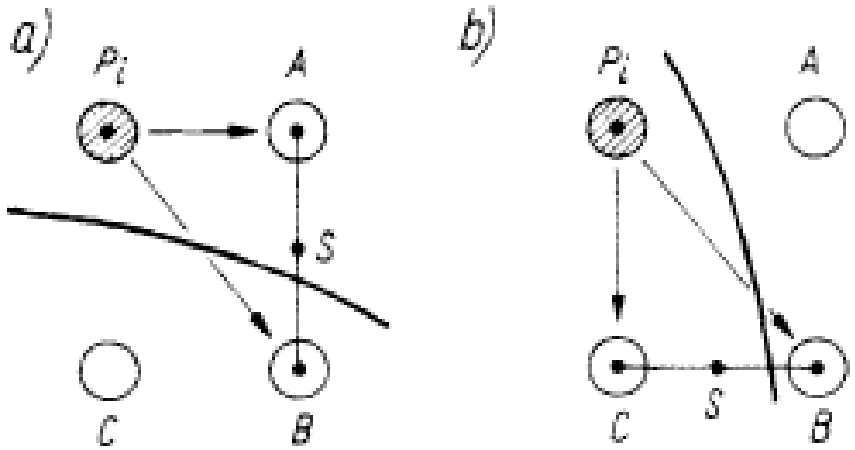


Rys. 5: Wybór pikseli w algorytmie Bresenhama rysowania okręgu: a) we fragmencie 1; b) we fragmencie 2

O wyborze będzie decydować wartość f w punkcie środkowym S między alternatywnymi pikselami.
Pierwszy fragment:

- jeśli $f_{S_i} = f(x_i + 1, y_i - 1/2) > 0$, to S zewnątrz okręgu i wybieramy $P_{i+1} = B$
- jeśli $f_{S_i} = f(x_i + 1, y_i - 1/2) < 0$, to S wewnątrz okręgu i wybieramy $P_{i+1} = A$

Rysowanie okręgu



Rys. 6: Wybór pikseli w algorytmie Bresenhama rysowania okręgu: a) we fragmencie 1; b) we fragmencie 2

O wyborze będzie decydować wartość f w punkcie środkowym S między alternatywnymi pikselami.
Drugi fragment:

- jeśli $f_{S_i} = f(x_i + 1/2, y_i - 1) > 0$, to S zewnątrz okręgu i wybieramy $P_{i+1} = C$
- jeśli $f_{S_i} = f(x_i + 1/2, y_i - 1) < 0$, to S wewnątrz okręgu i wybieramy $P_{i+1} = B$

Algorytm rysowania okręgu

Start: $x = 0;\ \ y = R;$

$pp = p * p;\ \ pp4 = 4 * pp;\ \ pp8 = 8 * pp;$

$qq = q * q;\ \ qq4 = 4 * qq;\ \ qq8 = 8 * qq;$

$fx = 0;\ \ fy = qq8 * R;\ \ fs = pp4 - qq4 * R + qq;$

dopóki $fx < fy$

$P = (x, y);\ x = x + 1;\ fx = fx + pp8;$

jeśli $fs \leq 0$, to $fs = fs + fx + pp4;$

w przeciwnym razie $\{y = y - 1;\ fy = fy - qq8;$

$fs = fs + fx + pp4 - fy\ \}$

$fs = fs - (fx + fy)/2 + 3 * (pp - qq)$

dopóki $y \geq 0$

$P = (x, y);\ y = y - 1;\ fy = fy - qq8;$

jeśli $fs \leq 0$, to $\{\ x = x + 1;$

$fx = fx + pp8;\ fs = fs + fx - fy + qq4\ \}$

w przeciwnym razie $fs = fs - fy + gg4$

Wypełnianie obszarów

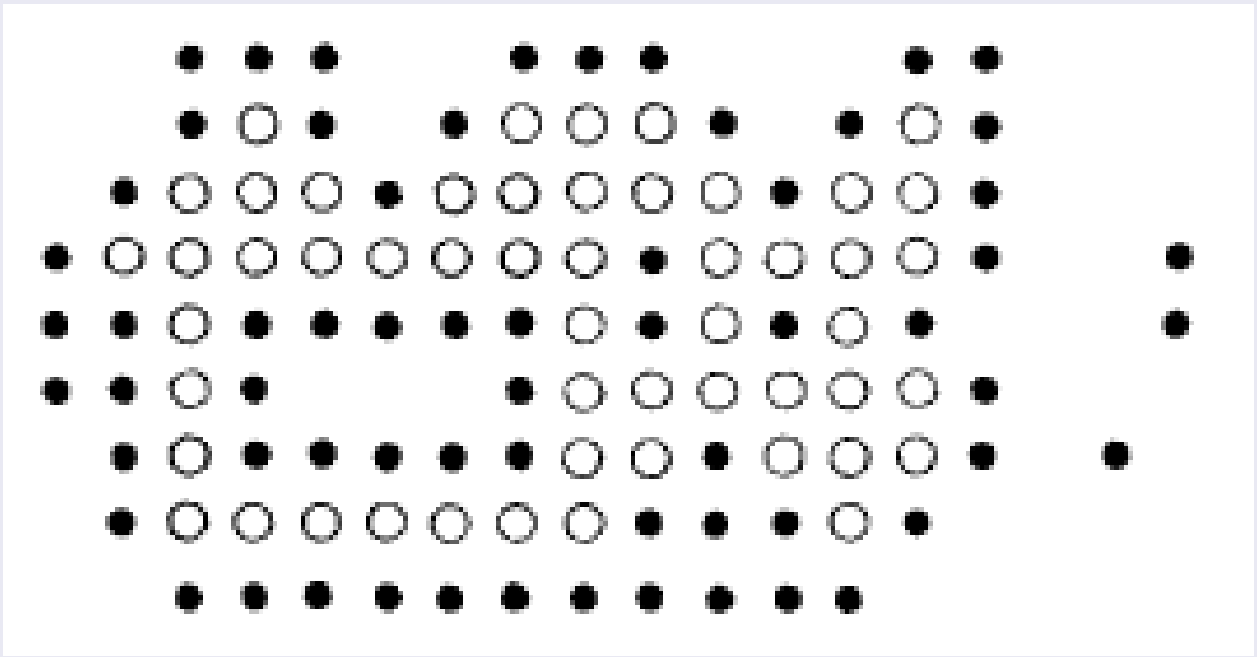
Spójność

Zbiór pikseli jest spójny, jeśli do dowolnego piksela z tego zbioru można przejść do każdego innego przez piksele sąsiednie.

- gdy sąsiadów 4: zbiór czterospójny
- gdy sąsiadów 8: zbiór ośmiospójny

Wypełnianie obszarów

Spójność



Rys. 7: Obszar o brzegu ośmiospójnym i dziurach (piksele czarne) wypełniony zbiorem czterospójnym (piksele białe)

Wypełnianie obszarów

Spójność

Najczęściej rozważany przypadek: wnętrze obszaru jest zbiorem czterospójnym, a ograniczający brzeg – ośmiospójnym.
Niech brzeg został narysowany pewnym kolorem *cb*. Dopuśćmy istnienie dziur (wysp) wewnątrz obszaru – będą nimi podobszary ograniczone ośmiospójnymi brzegami pikseli w kolorze *cb*. W szczególnym przypadku (np. *cb* – czarny) dziurami mogą być pojedyncze piksele w tym kolorze. *Podkreślmy, że kolorem cb mogą też być wyświetlone piksele leżące na zewnątrz zadanego obszaru.*
Znając ziarno (ang. seed), czyli piksel leżący wewnątrz obszaru, możemy wypełnić go nowym kolorem i próbować siać ten kolor (albo, inaczej, propagować przez spójność) w czterech kierunkach, tzn. sprawdzać, czy 4 sąsiednie piksele należą do wnętrza obszaru i czy nie zostały jeszcze wypełnione nowym kolorem. Dalej postępujemy tak samo, badając piksele sąsiadujące z sąsiadami ziarna itd. Ten rekurencyjny algorytm możemy przedstawić w formie podanej niżej procedury.

Wypełnianie obszarów

Wypełnienie obszaru przez sianie

procedure fill4(x, y, cb, cn) {(x, y) - współrzędne piksela}
begin {cb - kolor brzegu, cn - kolor wypełnienia}
if color$(x, y) \neq cb$ **and** color$(xy) \neq cn$ **then**
{kolor piksela (x, y) jest różny od cb i cn}
setcolor(x, y, cb, cn); {wypełnienie piksela (x, y) kolorem cb}
fill4$(x, y - 1, cb, cn)$;
fill4$(x, y + 1, cb, cn)$; {propagacja w czterech kierunkach}
fill4$(x - 1, y, cb, cn)$;
fill4$(x + 1, y, cb, cn)$;
end
end

Wypełnianie obszarów

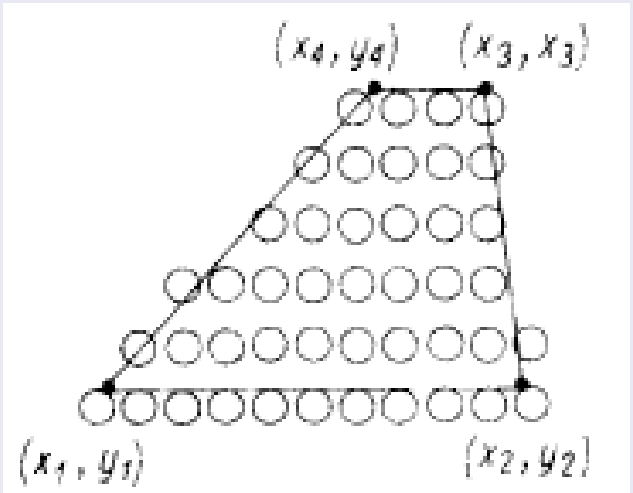
Wypełnienie obszaru przez sianie

Zapis algorytmu jest bardzo prosty i krótki. Jego praktyczna realizacja może jednak sprawiać trudności i być kosztowna. Przy dużym obszarze głębokość rekurencji jest na ogół znaczna, a wtedy nierzadko następuje przepełnienie budowanego stosu. Algorytm jest rozrzutny – kolor tego samego piksela bada się kilka razy, nawet pięciokrotnie. Opracowano wiele bardziej efektywnych metod. Dotyczą one najczęściej nie działania na poszczególnych pikselach, ale na ich grupach. Mogą to być na przykład poziome segmenty definiowane jako ciągi leżących wewnątrz obszaru pikseli sąsiadujących ze sobą w poziomie. Przyjmuje się, że taki ciąg ma maksymalną możliwą długość, a więc segment jest z lewej i z prawej strony ograniczony pikselami brzegowymi. Ponadto zakłada się, że nie zawiera on pikseli w nowym kolorze *cn*. Segment określają jednoznacznie współrzędne piksela leżącego najbardziej na lewo i długość (liczba tworzących go pikseli).

Wypełnianie obszarów

Wypełnienie trapezu

Inny przypadek danych: obszar jest określony analitycznie jako wielokąt.
Niech płaszczyzna rysunku będzie opisana współrzędnymi pikselowymi. Jako przykład rozpatrzymy wypełnianie trapezu o podstawach równoległych do osi $0x$. Na ogół wierzchołki trapezu mają współrzędne rzeczywiste, czyli nie pokrywające się ze środkami pikseli, tak jak na rys. 8



Rys. 8: Wypełnianie trapezu

Wypełnianie obszarów

Wypełnienie trapezu

Niech $y_{min} = \text{round}(y_1)$, a $y_{max} = \text{round}(y_3)$, gdzie $\text{round}(r)$ jest zaokrągleniem liczby rzeczywistej r do najbliższej liczby całkowitej. Z założenia:

$$y_1 = y_2; \qquad y_3 = y_4$$

Niech

$$c_l = \frac{x_4 - x_1}{y_4 - y_1}; \qquad c_p = \frac{x_3 - x_2}{y_3 - y_2}$$

Przy tych oznaczeniach algorytm wypełniania trapezu może przebiegać następująco:

Wypełnianie obszarów

Wypełnienie trapezu

dla $y = y_{min}, y_{min} + 1, y_{min} + 2, \ldots, y_{max}$,

- wyznacz przecięcia x_l i x_p linii poziomej y z prostymi

$$x = x_1 + (y - y_1)c_l; \qquad x = x_2 + (y - y_2)c_p$$

- wypełnij nowym kolorem (wzorcem) piksele leżące na tej linii od round(x_l) do round(x_p)

Wypełnianie obszarów

Wypełnienie trapezu

Zauważmy, że dla
$y = y_1 = y_2$ jest $x_l = x_1$, $x_p = x_2$
oraz że między przecięciami x_l i x_p obu prostych z linią poziomą y a przecięciami x'_l i x'_p z kolejną linią poziomą $y + 1$ zachodzą zależności

$$x'_l = x_l + c_l; \qquad x'_p = x_p + c_p$$

Jeśli wykorzystamy te zależności, to algorytm uprości się do postaci

Wypełnianie obszarów

Wypełnienie trapezu

podstaw $x_l = x_1;\ x_p = x_2$

dla $y = y_{min}, y_{min} + 1, y_{min} + 2, \ldots, y_{max}$,

- wypełnij nowym kolorem (wzorcem) piksele leżące na tej linii od round(x_l) do round(x_p)
- zmień $x_l = x_l + c_l;\ x_p = x_p + c_p$

Zadanie wypełniania dowolnego wielokąta można sprowadzić do omówionego wyżej przypadku szczególnego. Każdy wielokąt można rozłożyć na sumę trapezów. Taki rozkład może być przydatny nie tylko do wypełniania obszaru danym kolorem (wzorcem), lecz także przy działaniach na wielokątach i przy rysowaniu obiektów trójwymiarowych z uwzględnieniem oświetlenia. Pozwala także zaoszczędzić obliczeń, gdy wielokrotnie wypełniamy obszary tego samego kształtu, ale różnej wielkości, na przykład przy drukowaniu mniejszych i większych czcionek ustalonego kroju.

Literatura

[1] J. E. Bresenham, *Algorithm for computer control of a digital plotter*, IBM Systems Journal, vol. 4, no. 1, 1965

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

Koniec? :-(

Koniec wykładu 2