БРЕСЕНХАМОВ АЛГОРИТАМ ЗА ЦРТАЊЕ ДУЖИ

Драган Машуловић, Институт за математику, Нови Сад

Дуж као геометријски објект нема дебљину и састоји се од бесконачно (заправо, континуум) много тачака. С друге стране, растер рачунарског екрана се састоји од коначно много пиксела који су дводимензионални објекти, рецимо кружићи. Алгоритам за цртање дужи одређује колекцију пиксела који леже на или у близини дате (идеалне) дужи.

У овом чланку ћемо представити једно чувено решење проблема растеризације дужи: Бресенхамов инкрементални алгоритам[1]. Реч "инкрементални" значи да се у сваком кораку алгоритма на основу координата пиксела који је управо осветљен одређују координате наредног пиксела кога треба осветлити. Бресенхамов алгоритам је значајан бар из два разлога:

- не користи реалну аритметику чиме се добија на брзини, и
- даје оптимално решење у следећем смислу: не постоји колекција пиксела која има мање средње квадратно одступање од идеалне дужи.

Нека су $A(x_0, y_0)$ и $B(x_1, y_1)$ крајње тачке дужи коју треба нацртати. Претпоставићемо да је $x_0 < x_1$ и да коефицијент правца праве AB припада интервалу $[0, 1]$. Ако уведемо следеће ознаке: $\Delta_y = y_1 - y_0$ и $\Delta_x = x_1 - x_0$, једначина праве AB има облик $y = kx + n$ за $k = \Delta_y/\Delta_x$ и неко n. Множењем са Δ_x и сређивањем добијамо:

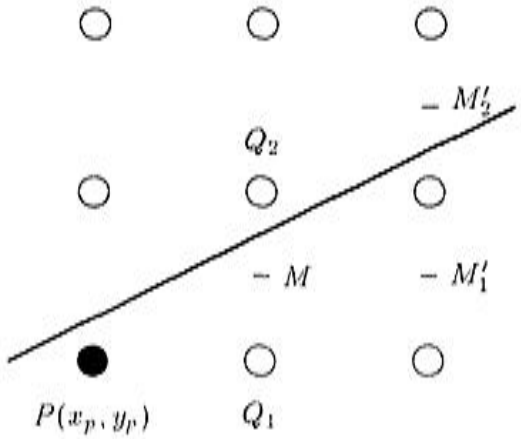
$$\Delta_y x - \Delta_x y + n\Delta_x = 0.$$

Из техничких разлога (једне несташне двојчице која се касније појави), помножићемо ову једначину са 2. Тако, уз ознаку $c = 2n\Delta_x$ добијамо

$$2\Delta_y x - 2\Delta_x y + c = 0.$$

Претпоставимо да смо у претходном кораку осветлили пиксел $P(x_p, y_p)$, Сл. . У следећем кораку ћемо осветлити или пиксел $Q_1(x_p + 1, y_p)$ или пиксел $Q_2(x_p + 1, y_p + 1)$ према следећем критеријуму. Посматрајмо тачку $M(x_p + 1, y_p + \frac{1}{2})$ која је средиште дужи $[Q_1Q_2]$. Ако се она налази испод праве AB, осветлићемо пиксел Q_1; у супротном ћемо осветлити пиксел Q_2. Да ли се тачка M налази испод или изнад праве AB утврђујемо на уобичајени начин. Нека је $\lambda_M = 2\Delta_y(x_p + 1) - 2\Delta_x(y_p + \frac{1}{2}) + c$. Ако је $\lambda_M > 0$, тачка M се налази испод праве AB; у супротном се налази на правој AB или изнад ње. (Ово је позната особина једначине праве:

[1] Bresenham, J. E., Algorithm for Computer Control of a Digital Plotter, IBM Systems Journal, 4(1) 1965, str. 25–30



Сл. 1: Одабир наредног пиксела при Бресенхамовом алгоритму

када неку тачку уврстимо у једначину праве, знак добијеног броја говори о томе са које стране праве се налази тачка!)

Приликом цртања дужи крећемо од тачке са x-координатом x_0 и понављамо поступак док не дођемо до тачке са x-координатом x_1. Основни алгоритам, зато, изгледа овако:

```
x := x0;
y := y0;
Dot(x, y);
WHILE  x < x1  DO BEGIN
    IF  λ_M(x+1,y+1/2) ≤ 0  THEN
        x := x + 1
    ELSE BEGIN
        x := x + 1;
        y := y + 1
    END;
    Dot(x, y);
END
```

(При томе, `Dot` је функција која осветљава пиксел на датим координатама. Није део стандардног Pascala, а њен еквивалент се може наћи у библиотекама које прате имплементацију.)

Израз λ_M такође може да се рачуна инкрементално, чиме се алгоритам значајно убрзава. Нека је $\lambda_M = 2\Delta_y(x_p+1) - 2\Delta_x(y_p+\frac{1}{2}) + c$. Ако смо одабрали пиксел Q_1, следећа тачка на основу чијег положаја доносимо одлуку је $M_1'(x_p+2, y_p+\frac{1}{2})$, и требаће нам вредност израза $\lambda_{M_1'} = 2\Delta_y(x_p+2) - 2\Delta_x(y_p+\frac{1}{2}) + c$. Приметимо да је $\lambda_{M_1'} = \lambda_M + 2\Delta_y$.

Ако смо одабрали пиксел Q_2, следећа тачка на основу чијег положаја доносимо одлуку је $M_2'(x_p+2, y_p+\frac{3}{2})$, и требаће нам вредност израза $\lambda_{M_2'} = 2\Delta_y(x_p+2) - 2\Delta_x(y_p+\frac{3}{2}) + c$. Приметимо да је $\lambda_{M_2'} = \lambda_M + 2(\Delta_y - \Delta_x)$.

Дакле, нема потребе да се изрази $\lambda_{M_1'}$ и $\lambda_{M_2'}$ рачунају од почетка: у првом случају је довољно на λ_M додати $2\Delta_y$, а у другом случају $2(\Delta_y - \Delta_x)$.

Потражимо сада почетну вредност за λ_M која се добија за тачку $M_0(x_0 + 1, y_0 + \frac{1}{2})$:

$$\begin{aligned}\lambda_{M_0} &= 2\Delta_y(x_0+1) - 2\Delta_x(y_0+\tfrac{1}{2}) + c \\ &= (2\Delta_y x_0 - 2\Delta_x y_0 + c) + 2\Delta_y - \Delta_x \\ &= \lambda_A + 2\Delta_y - \Delta_x.\end{aligned}$$

Зато што тачка A припада правој AB имамо $\lambda_A = 0$, и тиме добијамо почетну вредност $\lambda_{M_0} = 2\Delta_y - \Delta_x$. Коначна верзија алгоритма изгледа овако:

```
PROCEDURE Line0(x0, y0, x1, y1 : INTEGER);
VAR
    lambda, x, y, dx, dy, incr1, incr2 : INTEGER;
BEGIN
    dx := x1 - x0;
    dy := y1 - y0;
    lambda := 2 * dy - dx;
    incr1 := 2 * dy;
    incr2 := 2 * (dy - dx);

    x := x0;
    y := y0;
    Dot(x, y);
    WHILE x < x1 DO BEGIN
        IF lambda <= 0 THEN
            BEGIN
                x := x + 1;
                lambda := lambda + incr1
            END
        ELSE
            BEGIN
                x := x + 1;
                y := y + 1;
                lambda := lambda + incr2
            END;
        Dot(x, y)
    END
END {Line0};
```

Напомињемо још једном да процедура `Line0` ради коректно само за $x_0 < x_1$ и само за оне дужи чији коефицијент правца припада интервалу $[0, 1]$.

Задаци

1. Написати процедуру `Line` која црта дуж без икаквих ограничења.

2. Написати процедуру

```
ThickLine(x0, y0, x1, y1, w : INTEGER)
```

која црта дуж са крајњим тачкама `(x0, y0)` и `(x1, y1)`, али тако да дебљина дужи буде `w` пиксела.

Ако је нагиб дужи између -1 и 1, задебљање цртати као вертикални низ од `w` пиксела, од којих је половина изнад, а половина испод основне линије. Ако је нагиб дужи већи од 1 или мањи од -1, задебљање цртати као хоризонтални низ од `w` пиксела, од којих је половина лево, а половина десно од основне линије.

3. Написати процедуру

```
StyledLine(x0, y0, x1, y1 : INTEGER; sty : Style)
```

која црта дуж са крајњим тачкама `(x0, y0)` и `(x1, y1)`, при чему је стил (црта-црта, црта-тачка-црта и сл.) описан променљивом `sty`, а тип `Style` је дефинисан са

```
TYPE Style = SET OF [0 .. 15];
```

Уведите помоћну променљиву `k` која броји нацртане пикселе (односно, пикселе које би процедура `Line` нацртала). Процедуру `Dot(x, y)` позвати само ако је `k MOD 16 IN sty`. На пример, за `sty` = {0, 1, 2, 3, 4, 5, 6, 7} осам пиксела ће бити нацртано, осам прескочено, и тако редом. Добићемо црта-црта линију, где је дужина појединачних цртица 8 пиксела. За `sty` = {0, 1, 2, 3, 4, 5, 6, 7, 11, 12} добијамо црта-тачка-црта линију.

4. Написати процедуру

```
ThickStyledLine(x0, y0, x1, y1, w : INTEGER; sty : Style)
```

која црта подебљану дуж са назначеним стилом.

Статијата прв пат е објавена во списанието Тангента на ДМ на Србија во 2000/01 година