STRUCTURAL DESCRIPTION OF DIGITIZED STRAIGHT LINE SEGMENTS

Jacek Lebiedź
Technical University of Gdańsk
Faculty of Electronics, Telecommunications and Informatics
e-mail: jacekl@pg.gda.pl

ABSTRACT

This paper presents a structural description of digitized straight line segments and shows a scan converting algorithm based on this description. At the end, the obtained structural algorithm is compared with Bresenham's algorithm. The former seems to be faster than the latter.

Keywords: structural description, discrete line segments, scan conversion, raster graphics

1. INTRODUCTION

We can describe every line on a lattice (raster) as a sequence of adjacent pixels. Therefore it can be drawn using the unit movement pattern like that of a king piece in chess. Every direction of the movement can be encoded as an octal digit. This kind of encoding called Freeman's chain coding [Freem70, Brons85, Pavli82] is shown in Fig. 1.

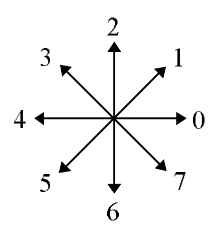


Fig. 1. Freeman's chain coding scheme.

Obviously, every segment is a line too and it can be represented by corresponding with it Freeman's chain. What properties must Freeman's chain possess to describe a straight line segment? Freeman has formulated three properties of Freeman's chain representing a straight line segment [Freem70, Brons85]:

1. At most two types of octal digits can be present, and these can differ only by unity, modulo eight (e.g. these are digits 0 and 1 for an angle of segment slope $\varphi \in (0,\pi/4)$; Fig. 2).
2. One of the two octal digits always occurs singly (e.g. each digit one must be surrounded by zeros for an angle of segment slope $\varphi \in (0, \arctan(1/2))$; Fig. 2).
3. Successive occurrences of the single octal digit are as uniformly spaced as possible among codes of the other value (e.g. 010010010 for a straight line segment connecting (0,0) and (9,3); Fig. 2).

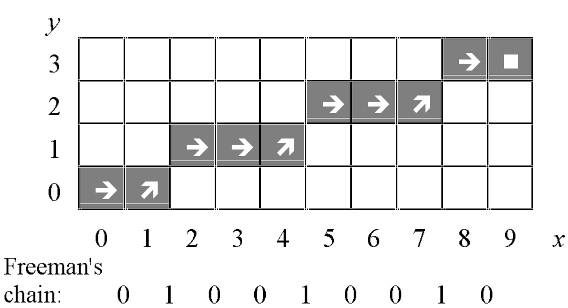


Fig. 2. The digitized straight line segment connecting (0,0) and (9,3), and its Freeman's chain.

On the ground of these properties, Brons has formulated an algorithm for drawing of straight line segment [Brons85, Castl85]. This algorithm, in short, consists in an iterative "blending" of two sets of identical digits.

Time-consuming string manipulation is a basic disadvantage of this algorithm. What is more, lengths of strings may come up to a value equal to a number of segment pixels, in extreme instance. We cannot estimate their maximal lengths by a constant. Therefore we can state that Brons' algorithm cannot be competitive for the standard conditional Bresenham's one [Brese65, Brese87] that calculates one pixel per loop and uses just a few variables. Only an exact structural description enables us to construct an algorithm faster than the Bresenham's.

2. STRUCTURAL DESCRIPTION

Assume for simplification that we deal with a straight line segment which has an angle of slope $\varphi \in (0,\ \arctan(1/2))$. In accordance with the first Freeman's property, Freeman's chain of this segment will consist only of zeros and ones. According to the second Freeman's property, each digit one will be surrounded by zeros (the number of zeros is greater than the number of ones). If we designate coordinates of the segment beginning by (x_s, y_s), and its end by (x_e, y_e), then we can write a number of ones in Freeman's chain as $\beta = y_e - y_s$, and a number of zeros as $\alpha = x_e - x_s - \beta$ (Fig. 2).

Using these designations, we can express the first two Freeman's properties in a formula:

$$\prod_{i=1}^{\beta} 0^{\alpha_{2i-1}}\, 1\, 0^{\alpha_{2i}}, \qquad \sum_{j=1}^{2\beta} \alpha_j = \alpha,$$

where $\prod_{i=1}^{\beta} A_i = A_1 \ldots A_n$, $\quad A^n = \underbrace{AA\ldots A}_{n}$.

The third Freeman's property tells us that the values α_j should be as near each other as possible. In fact it means that α_j must be equal to the one of two values ([.] - integer part):

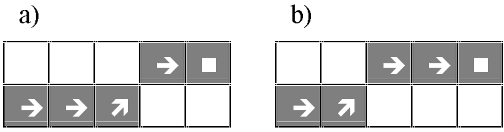
$$\left[\frac{\alpha}{2\beta}\right] = \alpha \ \underline{\text{div}}\ (2\beta) \quad \text{or} \quad \left[\frac{\alpha}{2\beta}\right] + 1 = \alpha \ \underline{\text{div}}\ (2\beta) + 1.$$

Distribution of these values should be as uniform as possible. As it is shown below, the following iterative formula ensures this property:

$$\alpha_j = \text{round}\left(\frac{\alpha}{2\beta} j - \sum_{k=1}^{j-1} \alpha_k\right),$$

where round(.) means the nearest integer i.e. $\text{round}(x) = [x + \frac{1}{2}]$. Another rounding function can be used too, but we will not consider it (Fig. 3).

a) b)



Freeman's chains: 0 0 1 0 0 1 0 0

Fig. 3. An example of digitized straight line segment, for which different definitions of the function round give different Freeman's chains (i.e. different shapes):

$$\text{round}(x) = [x + \tfrac{1}{2}] \quad \text{(a)}$$

$$\text{or} \quad \text{round}(x) = \lim_{\zeta \to 0} [x + \tfrac{1}{2} - \zeta] \quad \text{(b)}.$$

We can express this formula without using of the function round:

$$\alpha_j = \left(\alpha j - 2\beta \sum_{k=1}^{j-1} \alpha_k + \beta\right) \underline{\text{div}}\ (2\beta).$$

Note that:

$$\alpha_{2\beta} = \left(2\beta\alpha - 2\beta \sum_{k=1}^{2\beta-1} \alpha_k + \beta\right) \underline{\text{div}}\ (2\beta) =$$

$$= \left(\alpha - \sum_{k=1}^{2\beta-1} \alpha_k\right),$$

hence $\sum_{k=1}^{2\beta} \alpha_k = \alpha$.

Because the proposed formula needs a division for each α_j, we will look for a more "efficient" expression. To simplify our consideration, we introduce a new designation $\gamma_j = 2\beta\alpha_j$. Obviously, $\gamma_j \ \underline{\text{mod}}\ (2\beta) = (2\beta\alpha_j) \ \underline{\text{mod}}\ (2\beta) = 0$. Using the identity: $\forall x \in \mathbf{N}\ \forall y \in \mathbf{N} \quad y\ (x \ \underline{\text{div}}\ y) = x - x \ \underline{\text{mod}}\ y$, we obtain:

$$\gamma_j = 2\beta\alpha_j = \left(\alpha j - 2\beta \sum_{k=1}^{j-1} \alpha_k + \beta\right) \underline{\text{div}}\ (2\beta) * 2\beta =$$

$$= \left(\alpha j - \sum_{k=1}^{j-1} \gamma_k + \beta\right) \underline{\text{div}}\ (2\beta) * 2\beta =$$

$$= \left(\alpha j - \sum_{k=1}^{j-1} \gamma_k + \beta\right) - \left(\alpha j - \sum_{k=1}^{j-1} \gamma_k + \beta\right) \underline{\text{mod}}\ (2\beta) =$$

$$= (\alpha j + \beta) - (\alpha j + \beta) \ \underline{\text{mod}}\ (2\beta) - \sum_{k=1}^{j-1} \gamma_k ,$$

hence $\sum_{k=1}^{j} \gamma_k = (\alpha j + \beta) - (\alpha j + \beta) \ \underline{\text{mod}}\ (2\beta)$.

Now, we can calculate γ_j as a result of a subtraction:

$$\gamma_j = \sum_{k=1}^{j} \gamma_k - \sum_{k=1}^{j-1} \gamma_k =$$

$$= (\alpha j + \beta) - (\alpha(j-1) + \beta) + - (\alpha j + \beta) \ \underline{\text{mod}}\ (2\beta) + (\alpha(j-1) + \beta) \ \underline{\text{mod}}\ (2\beta) =$$

$$= \alpha - (\alpha j + \beta) \ \underline{\text{mod}}\ (2\beta) + (\alpha(j-1) + \beta) \ \underline{\text{mod}}\ (2\beta) =$$

$$= \begin{cases} \alpha - (\alpha j + \beta - \alpha(j-1) - \beta) \ \underline{\text{mod}}\ (2\beta) \\ \quad \text{for } (\alpha j + \beta) \ \underline{\text{mod}}\ (2\beta) \geq (\alpha(j-1) + \beta) \ \underline{\text{mod}}\ (2\beta), \\ \alpha - (\alpha j + \beta - \alpha(j-1) - \beta) \ \underline{\text{mod}}\ (2\beta) + 2\beta \\ \quad \text{for } (\alpha j + \beta) \ \underline{\text{mod}}\ (2\beta) < (\alpha(j-1) + \beta) \ \underline{\text{mod}}\ (2\beta). \end{cases}$$

Simplifying, we obtain:

$$\gamma_j=\begin{cases}\alpha-\alpha\ \underline{\mathrm{mod}}\ (2\beta) \\ \quad \text{for } (\alpha j+\beta)\ \underline{\mathrm{mod}}\ (2\beta)\geq(\alpha(j-1)+\beta)\ \underline{\mathrm{mod}}\ (2\beta), \\ \alpha-\alpha\ \underline{\mathrm{mod}}\ (2\beta)+2\beta \\ \quad \text{for } (\alpha j+\beta)\ \underline{\mathrm{mod}}\ (2\beta)<(\alpha(j-1)+\beta)\ \underline{\mathrm{mod}}\ (2\beta).\end{cases}$$

Introducing $\lambda_j=(\alpha(j-1)+\beta)\ \underline{\mathrm{mod}}\ (2\beta)$, we have:

$$\lambda_{j+1}=(\alpha j+\beta)\ \underline{\mathrm{mod}}\ (2\beta)=$$

$$=(\alpha+\alpha(j-1)+\beta)\ \underline{\mathrm{mod}}\ (2\beta)=$$

$$=\begin{cases}(\alpha(j-1)+\beta)\ \underline{\mathrm{mod}}\ (2\beta)+\alpha\ \underline{\mathrm{mod}}\ (2\beta) \\ \quad \text{for } \alpha\ \underline{\mathrm{mod}}\ (2\beta)+(\alpha(j-1)+\beta)\ \underline{\mathrm{mod}}\ (2\beta)<2\beta, \\ (\alpha(j-1)+\beta)\ \underline{\mathrm{mod}}\ (2\beta)+\alpha\ \underline{\mathrm{mod}}\ (2\beta)-2\beta \\ \quad \text{for } \alpha\ \underline{\mathrm{mod}}\ (2\beta)+(\alpha(j-1)+\beta)\ \underline{\mathrm{mod}}\ (2\beta)\geq 2\beta.\end{cases}$$

$$=\begin{cases}\lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta) & \text{for } \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)<2\beta, \\ \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)-2\beta & \\ & \text{for } \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)\geq 2\beta.\end{cases}$$

Now, we can express γ_j in the simpler way:

$$\gamma_j=\begin{cases}\alpha-\alpha\ \underline{\mathrm{mod}}\ (2\beta) & \text{for } \lambda_{j+1}\geq\lambda_j, \\ \alpha-\alpha\ \underline{\mathrm{mod}}\ (2\beta)+2\beta & \text{for } \lambda_{j+1}<\lambda_j,\end{cases}$$

but

$$\lambda_{j+1}\geq\lambda_j \Leftrightarrow \lambda_{j+1}=\lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta) \Leftrightarrow \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)<2\beta,$$

$$\lambda_{j+1}<\lambda_j \Leftrightarrow \lambda_{j+1}=\lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)-2\beta \Leftrightarrow \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)\geq 2\beta,$$

thus

$$\gamma_j=\begin{cases}\alpha-\alpha\ \underline{\mathrm{mod}}\ (2\beta) & \text{for } \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)<2\beta, \\ \alpha-\alpha\ \underline{\mathrm{mod}}\ (2\beta)+2\beta & \text{for } \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)\geq 2\beta.\end{cases}$$

Since $\gamma_j=2\beta\alpha_j$, we can pass to α_j and write:

$$\alpha_j=\begin{cases}\alpha\ \underline{\mathrm{div}}\ (2\beta) & \text{for } \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)<2\beta, \\ \alpha\ \underline{\mathrm{div}}\ (2\beta)+1 & \text{for } \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)\geq 2\beta.\end{cases}$$

Uniformity of distribution of both digits is self-evident.

At last the structural description of a digitized straight line segment with a beginning (x_s,y_s) and an end (x_e,y_e), and with an angle of slope $\varphi\in(0,\ \arctan(1/2))$ (i.e. $(x_e-x_s)>2(y_e-y_s)$) is following:

$$\prod_{i=1}^{\beta} 0^{\alpha_{2i-1}}\ 1\ 0^{\alpha_{2i}},$$

where $\prod_{i=1}^{\beta} A_i=A_1\ldots A_n$, $\quad A^n=\underbrace{AA\ldots A}_{n}$,

$$\alpha_j=\begin{cases}\alpha\ \underline{\mathrm{div}}\ (2\beta) & \text{for } \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)<2\beta, \\ \alpha\ \underline{\mathrm{div}}\ (2\beta)+1 & \text{for } \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)\geq 2\beta,\end{cases}$$

$\lambda_j=(\alpha(j-1)+\beta)\ \underline{\mathrm{mod}}\ (2\beta)$,

i.e. $\lambda_1=\beta$,

$$\lambda_{j+1}=\begin{cases}\lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta) & \text{for } \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)<2\beta, \\ \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)-2\beta & \text{for } \lambda_j+\alpha\ \underline{\mathrm{mod}}\ (2\beta)\geq 2\beta,\end{cases}$$

$\beta=y_e-y_s$ - the number of ones in Freeman's chain of segment,

$\alpha=x_e-x_s-\beta$ - the number of zeros in this chain.

3. STRUCTURAL ALGORITHM

On the ground of this formula, we can formulate an algorithm for drawing of a straight line segment for an angle of segment slope $\varphi\in(0,\ \arctan(1/2))$. Because of duplicated sequence of operations in its main loop, this algorithm is not optimal yet.

Algorithm 1. Structural algorithm for drawing of a straight line segment connecting points (x_s,y_s), (x_e,y_e) for an angle of segment slope $\varphi\in(0,\ \arctan(1/2))$; in this algorithm and in the next one, auxiliary variables, the values of which depend entirely on other variables, have names structured in accordance with the reverse Polish notation, where small letters and digits mean operands (variables and numbers), capital letters mean operators: X - multiplication, M - remainder (mod), D - integer division (div).

```
var x,y,xe,ye,dx,dy,a,b,b2X,ab2XM,ab2XD,l,i,j: integer;
x := xs;  y := ys;  xe := xe;  ye := ye;
dx := xe-x;  dy := ye-y;
a := dx-dy;   b := dy;   l := b;
b2X := b + b;
ab2XM := a mod b2X;   ab2XD := a div b2X;
plot(x,y);
for i := 1 to b do
    begin
        for j := 1 to ab2XD do
            begin  x := x + 1;   plot(x,y)  end;
        l := l + ab2XM;
        if  l ≥ b2X  then
            begin l := l - b2X; x := x + 1; plot(x,y) end;
        x := x + 1;   y := y + 1;   plot(x,y);
        for j := 1 to ab2XD do
            begin  x := x + 1;   plot(x,y)  end;
        l := l + ab2XM;
        if  l ≥ b2X  then
            begin l := l - b2X; x := x + 1; plot(x,y) end
    end.
```

Additionally, we can optimize this algorithm. Note that the group of zeros α_{2i} adjoins the group α_{2i+1} for $i = 1 .. \beta - 1$. To bring together both the groups we should count $\gamma_{2i} + \gamma_{2i+1}$.

$\gamma_{2i} + \gamma_{2i+1}$ =

$$
=\begin{cases}
2\alpha - 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) & \text{for } \lambda_{2i} + \alpha\ \underline{\mathrm{mod}}\ (2\beta) < 2\beta \\
 & \ \ \& \ \lambda_{2i+1} + \alpha\ \underline{\mathrm{mod}}\ (2\beta) < 2\beta, \\
2\alpha - 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) + 2\beta & \text{for } \lambda_{2i} + \alpha\ \underline{\mathrm{mod}}\ (2\beta) < 2\beta \\
 & \ \ \& \ \lambda_{2i+1} + \alpha\ \underline{\mathrm{mod}}\ (2\beta) \geq 2\beta \\
 & \text{or } \lambda_{2i} + \alpha\ \underline{\mathrm{mod}}\ (2\beta) \geq 2\beta \\
 & \ \ \& \ \lambda_{2i+1} + \alpha\ \underline{\mathrm{mod}}\ (2\beta) < 2\beta, \\
2\alpha - 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) + 4\beta & \text{for } \lambda_{2i} + \alpha\ \underline{\mathrm{mod}}\ (2\beta) \geq 2\beta \\
 & \ \ \& \ \lambda_{2i+1} + \alpha\ \underline{\mathrm{mod}}\ (2\beta) \geq 2\beta,
\end{cases}
$$

$$
=\begin{cases}
2\alpha - 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) & \text{for } \lambda_{2i} + \alpha\ \underline{\mathrm{mod}}\ (2\beta) < 2\beta \\
 & \ \ \& \ \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) < 2\beta, \\
2\alpha - 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) + 2\beta & \text{for } \lambda_{2i} + \alpha\ \underline{\mathrm{mod}}\ (2\beta) < 2\beta \\
 & \ \ \& \ \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) \geq 2\beta \\
 & \text{or } \lambda_{2i} + \alpha\ \underline{\mathrm{mod}}\ (2\beta) \geq 2\beta \\
 & \ \ \& \ \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) < 4\beta, \\
2\alpha - 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) + 4\beta & \text{for } \lambda_{2i} + \alpha\ \underline{\mathrm{mod}}\ (2\beta) \geq 2\beta \\
 & \ \ \& \ \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) \geq 4\beta,
\end{cases}
$$

$$
=\begin{cases}
2\alpha - 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) < 2\beta, \\
2\alpha - 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) + 2\beta & \text{for } 2\beta \leq \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) < 4\beta, \\
2\alpha - 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) + 4\beta & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) \geq 4\beta,
\end{cases}
$$

hence $\alpha_{2i} + \alpha_{2i+1}$ =

$$
=\begin{cases}
2(\alpha\ \underline{\mathrm{div}}\ (2\beta)) & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) < 2\beta, \\
2(\alpha\ \underline{\mathrm{div}}\ (2\beta)) + 1 & \text{for } 2\beta \leq \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) < 4\beta, \\
2(\alpha\ \underline{\mathrm{div}}\ (2\beta)) + 2 & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) \geq 4\beta.
\end{cases}
$$

Note that

$$\alpha\ \underline{\mathrm{mod}}\ (2\beta) \geq \beta \quad \Leftrightarrow \quad \alpha\ \underline{\mathrm{div}}\ \beta = 2(\alpha\ \underline{\mathrm{div}}\ (2\beta)) + 1,$$
$$\alpha\ \underline{\mathrm{mod}}\ (2\beta) < \beta \quad \Leftrightarrow \quad \alpha\ \underline{\mathrm{div}}\ \beta = 2(\alpha\ \underline{\mathrm{div}}\ (2\beta)).$$

Thus $\alpha_{2i} + \alpha_{2i+1}$ =

$$
=\begin{cases}
\left.\begin{array}{ll}
\alpha\ \underline{\mathrm{div}}\ \beta & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) < 2\beta \\
\alpha\ \underline{\mathrm{div}}\ \beta + 1 & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) \geq 2\beta
\end{array}\right\} & \text{if } \alpha\ \underline{\mathrm{mod}}\ (2\beta) < \beta \\
\left.\begin{array}{ll}
\alpha\ \underline{\mathrm{div}}\ \beta & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) < 4\beta \\
\alpha\ \underline{\mathrm{div}}\ \beta + 1 & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) \geq 4\beta
\end{array}\right\} & \text{if } \alpha\ \underline{\mathrm{mod}}\ (2\beta) \geq \beta
\end{cases}
$$

$$
=\begin{cases}
\alpha\ \underline{\mathrm{div}}\ \beta & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ \beta) < 2\beta, \\
\alpha\ \underline{\mathrm{div}}\ \beta + 1 & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ \beta) \geq 2\beta,
\end{cases}
$$

because

$$
\alpha\ \underline{\mathrm{mod}}\ \beta = \begin{cases}
\alpha\ \underline{\mathrm{mod}}\ (2\beta) & \text{for } \alpha\ \underline{\mathrm{mod}}\ (2\beta) < \beta, \\
\alpha\ \underline{\mathrm{mod}}\ (2\beta) - \beta & \text{for } \alpha\ \underline{\mathrm{mod}}\ (2\beta) \geq \beta,
\end{cases}
$$

where λ_{2i+2} =

$$
=\begin{cases}
\lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) < 2\beta, \\
\lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) - 2\beta & \text{for } 2\beta \leq \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) < 4\beta, \\
\lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) - 4\beta & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) \geq 4\beta,
\end{cases}
$$

$$
=\begin{cases}
\left.\begin{array}{ll}
\lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) < 2\beta \\
\lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) - 2\beta & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) \geq 2\beta
\end{array}\right\} & \text{if } \alpha\ \underline{\mathrm{mod}}\ (2\beta) < \beta \\
\left.\begin{array}{ll}
\lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) - 2\beta & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) < 4\beta \\
\lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) - 4\beta & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ (2\beta)) \geq 4\beta
\end{array}\right\} & \text{if } \alpha\ \underline{\mathrm{mod}}\ (2\beta) \geq \beta
\end{cases}
$$

$$
=\begin{cases}
\lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ \beta) & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ \beta) < 2\beta, \\
\lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ \beta) - 2\beta & \text{for } \lambda_{2i} + 2(\alpha\ \underline{\mathrm{mod}}\ \beta) \geq 2\beta.
\end{cases}
$$

These formulas enable us to write another structural algorithm for drawing of a straight line segment with an angle of slope $\varphi \in (0, \arctan(1/2))$. It can be labeled the run length slice algorithm [Brese85].

The shown structural algorithms produce correct results for $\varphi \in (\arctan(1/2), \pi/4)$ too, but algorithms based on the formula $\prod 1^* 01^*$ are more efficient for this range of φ.

Algorithm 2. Run length slice algorithm for drawing of a straight line segment connecting points (x_s,y_s), (x_e,y_e) for an angle of segment slope $\varphi\in(0, \arctan(1/2))$.

```
var x,y,xe,ye,dx,dy,a,b,b2X,abD,abM2X,ab2XM,
   ab2XD,l,i,j: integer;
x := xs; y := ys; xe := xe; ye := ye;
dx := xe-x; dy := ye-y;
a := dx-dy;  b := dy;  l := b;
b2X := b + b;
ab2XM := a mod b2X;  ab2XD := a div b2X;
abD :=ab2XD +ab2XD; abM2X :=ab2XM +ab2XM;
if ab2XM ≥ b then
   begin abD :=abD +1; abM2X :=abM2X -b2X end;
plot(x,y);
for j := 1 to ab2XD do
   begin x := x + 1;  plot(x,y) end;
l := l + ab2XM;
if l ≥ b2X then
   begin l := l - b2X;  x := x + 1;  plot(x,y) end;
x := x + 1;  y := y + 1;  plot(x,y);
for i := 2 to b do
   begin
      for j := 1 to abD do
         begin x := x + 1;  plot(x,y) end;
      l := l + abM2X;
      if l ≥ b2X then
         begin l := l - b2X; x := x +1; plot(x,y) end;
      x := x + 1;  y := y + 1;  plot(x,y);
   end;
for j := 1 to ab2XD do
   begin x := x + 1;  plot(x,y) end;
if l + ab2XM ≥ b2X then
   begin x := x + 1;  plot(x,y) end.
```

It is proper to add that the derived algorithms both plot the pixels nearest to the "real" line. It means that they draw the same pixels as the conditional Bresenham's algorithm [Brese65, Brese87].

4. COMPARISON WITH OTHER ALGORITHMS

Note that both of the structural algorithms use limited number of memory cells. They execute only integer arithmetic operations. Thus they are more efficient than Brons' algorithms mentioned at the beginning of the paper [Brons85, Castl85, Pitte85]. What is more, the speed of drawing for both of the above-derived algorithms is comparable with the speed for the conditional Bresenham's algorithm [Brese65, Brese87]. If one of digits is clear dominating in Freeman's chain, the structural algorithms will be faster than Bresenham's. For considered by us angles of segment slope $\varphi\in(0, \arctan(1/2))$, it takes place when the number of zeros is much greater than the numbers of ones ($\alpha \gg \beta$). This situation occurs for small (near zero) angles of segment slope ($\varphi\to 0$).

Fig. 4 shows speed comparison of the structural algorithms (Algorithm 1 and Algorithm 2) with the conditional Bresenham's algorithm (Algorithm 3 in Appendix A). The latter structural algorithm (Algorithm 2) is faster than the former (Algorithm 1) for $\beta\geq 3$, and than the Bresenham's (Algorithm 3) for $\alpha\geq 8$ ($\varphi\in(0, \arctan(1/2))$). We can recognize the run length slice algorithm (Algorithm 2) to be the fastest algorithm we have consider. The calculated execution times of considered algorithms are placed in Table 1. Presented results are based on the assumption that the execution times of basic operations are defined by Table 2.

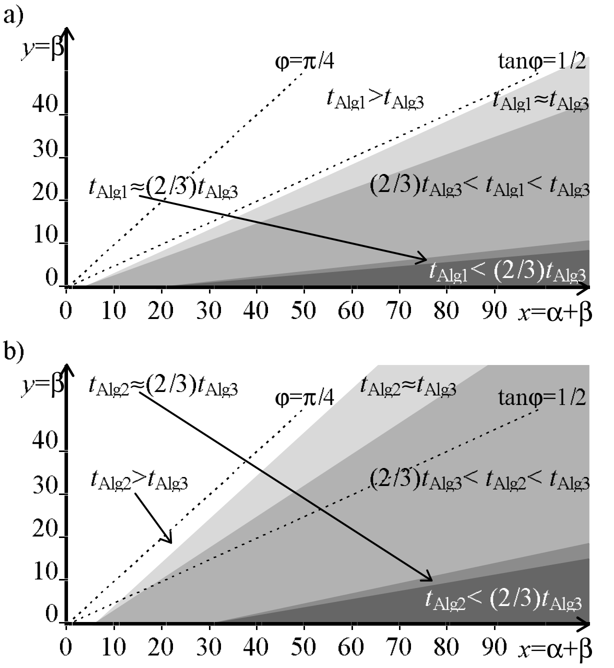


Fig. 4. Speed comparison of the structural algorithms (a - Algorithm 1, b - Algorithm 2) with the Bresenham's conditional algorithm (Algorithm 3 in Appendix A); although the compared structural algorithms were derived for $\varphi\in(0, \arctan(1/2))$, they produce correct results for a wider range $\varphi\in(0, \pi/4)$, but structural algorithms based on the inverse formula $\prod 1^*01^*$ will be faster for $\varphi\in(\arctan(1/2), \pi/4)$ and they should be used for this range of φ;
α - number of zeros in Freeman's chain,
β - number of ones in Freeman's chain.

algorithm	execution time $t_{algorithm}$
Algorithm 1 (structural)	$29+4\alpha+16\beta+2\beta[\alpha/(2\beta)]$
Algorithm 2 (structural)	$40+4\alpha+11\beta+\beta[\alpha/\beta]$
Algorithm 3 (conditional)	$15+9\alpha+10\beta$

Tab. 1. Calculated execution times of considered algorithms ([.] - integer part);
α - number of zeros in Freeman's chain,
β - number of ones in Freeman's chain.

operation	execution time $t_{operation}$
:=	1
+, −	1
$\cdot 2^n$ (shift)	1
in-, de-crementation	1
=, ≠, >, ≥, <, ≤	1
div mod	12
plot	3
if *cond* then *op*	t_{cond} + *cond* ? t_{op} : 1
if *cond* then *op1* else *op2*	t_{cond} + 1 + *cond* ? t_{op1} : t_{op2}
while *cond* do *op*	$(t_{cond} + 1) \cdot (\#+1) + \# \cdot t_{op}$
for *var*:=1 to # do *op*	$(\#+2) + \# \cdot t_{op}$

Tab. 2. Assumed execution times of basic operations (# - the number of loop runs, *cond* ? *u* : *v* ≡ if *cond* then *u* else *v*).

We omit the speed comparison of the structural algorithms with multiple-step or bisection algorithms [Angel91, Bao89, Gill94, Graha94, Graha95, Tokne90] because these improvements can be applied to the structural algorithms too.

5. CONCLUSION

The paper presents a precise structural description of digitized straight line segments with integer ends coordinates. This description may be very useful. For instance, it may serve as a definition of discrete straight line segment. The structural description may be used for deriving and proofing of properties of discrete straight line segments. The presented structural algorithm is just an example of usefulness of structural description.

REFERENCES

[Angel91] E. Angel, D. Morrison: Short Note: Speeding Up Bresenham's Algorithm, *IEEE CG&A*, Vol. 11, No. 6, November 1991, pp. 16÷17.

[Bao89] G. Bao, J. G. Rokne: Quadruple-Step Line Generation, *Computers & Graphics*, Vol.13, No. 4, 1989, pp. 461÷469.

[Brese65] J. E. Bresenham: Algorithm for Computer Control of a Digital Plotter, *IBM Systems J.*, Jan. 1965, pp. 25÷30.

[Brese85] J. E. Bresenham: Run Length Slice Algorithm for Incremental Lines, in *Fundamental Algorithms for Computer Graphics*, R. A. Earnshaw, Ed., Springer, Berlin 1985, pp. 59÷104.

[Brese87] J. E. Bresenham: Ambiguities in Incremental Line Rastering, *IEEE CG&A*, Vol. 7, No. 5, May 1987, pp. 31÷43.

[Brese96] J. E. Bresenham: Pixel-Processing Fundamentals, *IEEE CG&A*, Vol. 16, No. 1, January 1996, pp. 74÷82.

[Brons85] R. Brons: Theoretical and linguistic methods for describing straight lines, in *Fundamental Algorithms for Computer Graphics*, R. A. Earnshaw, Ed., Springer, Berlin 1985, pp. 19÷58.

[Castl85] C. M. A. Castle, M. L. V. Pitteway: An Application of Euclid's Algorithm to Drawing Straight Lines, in *Fundamental Algorithms for Computer Graphics*, R. A. Earnshaw, Ed., Springer, Berlin 1985, pp. 135÷140.

[Dorst84a] L. Dorst, R. P. W. Duin: A Framework for Calculations on Digitized Straight Lines, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 5, September 1984, pp. 632÷639.

[Dorst84b] L. Dorst, A. W. M. Smeulders: Discrete Representation of Straight Lines, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 4, July 1984, pp. 450÷463.

[Freem70] H. Freeman: Boundary encoding and processing, in *Picture processing and psychopictorics*, B. S. Lipkin and A. Rosenfeld, Eds., Academic, New York 1970, pp. 241÷266.

[Gill94] G. W. Gill: N-Step Incremental Straight-Line Algorithms, *IEEE CG&A*, Vol. 14, No. 5, May 1994, pp. 66÷72.

[Graha94] P. Graham, S. S. Iyengar: Double- and Triple-Step Incremental Linear Interpolation, *IEEE CG&A*, Vol. 14, No. 5, May 1994, pp. 49÷53.

[Graha95] P. Graham, S. S. Iyengar, S.-Q. Zheng: Improved Recursive Bisection Line Drawing Algorithms, *Comput. & Graphics*, Vol. 19, No. 6, May 1994, pp. 847÷860.

[Gross96] A. Gross, L. J. Latecki: Modeling Digital Straight Lines, *Proceedings of the 13-th International Conference on Pattern Recognition (Vienna 1996)*, Vol. II, Track B, IEEE Computer Society Press, Los Alamitos 1996, pp. 156÷160.

[Hégro85] G. Hégron: *Synthese d'image: algorithmes élémentaires*, Dunod informatique, Paris 1985.

[Hrúz95] T. Hrúz, I. Považan: A Note to the Structure of Bresenham Slice Algorithm, *Proceedings of WSCG'95*, Vol. II, 1995.

[Janko90] M. Jankowski: *Elementy grafiki komputerowej*, WNT, Warszawa 1990.

[Lebie96] J. Lebiedź: Structural description of digitized circles, *Machine Graphics & Vision*, Vol. 5, Nos. 1/2, 1996, pp. 423÷430.

[Pang90] A. T. Pang: Line-Drawing Algorithms for Parallel Machines, *IEEE CG&A*, Vol. 10, No. 5, September 1990, pp. 54÷59.

[Pavli82] T. Pavlidis: *Algorithms for graphics and image processing*, Berlin, Springer 1982.

[Pitte85] M. L. V. Pitteway: The Relationship between Euclid's Algorithm and Run-Length Encoding, in *Fundamental Algorithms for Computer Graphics*, R. A. Earnshaw, Ed., Springer, Berlin 1985, pp. 105÷112.

[Rokne90] J. G. Rokne, B. Wyvill, X. Wu: Fast Line Scan-Conversion, *ACM Trans. Graphics*, Vol. 9, No. 4, Oct. 1990, pp. 376÷388.

[Taubi94] G. Taubin: Rasterizing algebraic curves and surfaces, *IEEE CG&A*, Vol. 14, No. 3, March 1994, pp. 14÷23.

[Wrigh90] W. E. Wright: Parallelization of Bresenham's Line and Circle Algorithms, *IEEE CG&A*, Vol. 10, No. 5, September 1990, pp. 60÷67.

APPENDIX A: BRESENHAM'S ALGORITHM

Algorithm 3. Conditional Bresenham's algorithm for drawing of a straight line segment connecting points (x_s,y_s), (x_e,y_e) for an angle of segment slope $\varphi\in(0, \pi/4)$ [Brese65, Brese87].

```
var x,y,xe,ye,dx,dy,p,pinc,pdec: integer;
x := xs;  y := ys;  xe := xe;  ye := ye;
dx := xe-x;  dy := ye-y;
pinc := dy+dy;   p := pinc-dx;   pdec := p-dx;
while x ≠ xe do
    begin
        plot(x,y);
        x := x+1;
        if  p < 0 then p := p+pinc
                   else  begin y := y+1; p := p+pdec end
    end;
plot(xe,ye).
```