Algoritm

$$
\begin{aligned}
&\text{fulg}(p, d, k, \beta) := \left|
\begin{array}{l}
(xA \quad yA \quad xB \quad yB) \leftarrow p \\
\begin{pmatrix} xC \\ yC \end{pmatrix} \leftarrow \begin{pmatrix} \frac{xA + k \cdot xB}{k+1} \\ \frac{yA + k \cdot yB}{k+1} \end{pmatrix} \\
\text{if } xB - xA \neq 0 \\
\quad \left|
\begin{array}{l}
m \leftarrow \frac{yB - yA}{xB - xA} \\
distx \leftarrow \frac{d}{\sqrt{m^2 + 1}} \\
disty \leftarrow \frac{m \cdot d}{\sqrt{m^2 + 1}}
\end{array}
\right. \\
\text{otherwise} \\
\quad \left|
\begin{array}{l}
distx \leftarrow 0 \\
disty \leftarrow d
\end{array}
\right. \\
\begin{pmatrix} xD \\ yD \end{pmatrix} \leftarrow \begin{pmatrix} -distx + xC \\ -disty + yC \end{pmatrix} \quad \text{if } xB - xA < 0 \\
\begin{pmatrix} xD \\ yD \end{pmatrix} \leftarrow \begin{pmatrix} distx + xC \\ disty + yC \end{pmatrix} \quad \text{if } xB - xA > 0 \\
\begin{pmatrix} xE \\ yE \end{pmatrix} \leftarrow \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \cdot \begin{pmatrix} xD - xC \\ yD - yC \end{pmatrix} + \begin{pmatrix} xC \\ yC \end{pmatrix} \\
\begin{pmatrix} xF \\ yF \end{pmatrix} \leftarrow \begin{pmatrix} \cos(-\beta) & -\sin(-\beta) \\ \sin(-\beta) & \cos(-\beta) \end{pmatrix} \cdot \begin{pmatrix} xD - xC \\ yD - yC \end{pmatrix} + \begin{pmatrix} xC \\ yC \end{pmatrix} \\
\begin{pmatrix} xC & yC \\ xE & yE \\ xF & yF \end{pmatrix}
\end{array}
\right.
\end{aligned}
$$

Note algoritm *Fulg*

- acest algoritm realizează construcția unei unitati de fulg ce prezintă un segment [AB] și un punct C de unde pleacă două ramuri simetrice în punctele E și F
- A și B sunt punctele segmentului de la care se porneste
- C este punctul de pe segmentul [AB] de unde pleacă ramurile
- se calculează pozitia punctului D în funcție de poziția sa în cadran, D fiind un punct intermediar de pe segmentul [AB] pentru a putea calcula punctele E și F
- se calculează poziția punctelor E si F ale ramurilor care se obțin prin rotația punctului D în jurul punctului C, la un unghi ales, și în ambele sensuri, sus și jos.

Note algoritm *Fulg și Iterare*

- fiecare linie are un punct C de unde pleacă alte două linii, punctul C se determină prin calcul geometric. Dispunerea segmentelor prezintă o simetrie față de un alt segment până la formarea unei ramuri. Fiecare două segmente dispuse simetric se calculează funcție de cel precedent prin iterarea funcției *fulg()*. Lungimea segmentelor *d* se calculează pe baza unui coeficient de contracție.
- aceste *if-uri* s-au introdus pentru pentru a evita cazul *1/0*, folosinduse notații pentru a evita infiniții

Algoritm

$$
\begin{aligned}
&\text{fulg}(p, d, k, \beta) := \left|
\begin{array}{l}
(xA \quad yA \quad xB \quad yB) \leftarrow p \\
\begin{pmatrix} xC \\ yC \end{pmatrix} \leftarrow \begin{pmatrix} \frac{xA + k \cdot xB}{k+1} \\ \frac{yA + k \cdot yB}{k+1} \end{pmatrix} \\
\text{if } xB - xA \neq 0 \\
\quad \left|
\begin{array}{l}
m \leftarrow \frac{yB - yA}{xB - xA} \\
distx \leftarrow \frac{d}{\sqrt{m^2 + 1}} \\
disty \leftarrow \frac{m \cdot d}{\sqrt{m^2 + 1}}
\end{array}
\right. \\
\text{otherwise} \\
\quad \left|
\begin{array}{l}
distx \leftarrow 0 \\
disty \leftarrow d
\end{array}
\right. \\
\begin{pmatrix} xD \\ yD \end{pmatrix} \leftarrow \begin{pmatrix} -distx + xC \\ -disty + yC \end{pmatrix} \quad \text{if } xB - xA < 0 \\
\begin{pmatrix} xD \\ yD \end{pmatrix} \leftarrow \begin{pmatrix} distx + xC \\ disty + yC \end{pmatrix} \quad \text{if } xB - xA > 0 \\
\begin{pmatrix} xE \\ yE \end{pmatrix} \leftarrow \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \cdot \begin{pmatrix} xD - xC \\ yD - yC \end{pmatrix} + \begin{pmatrix} xC \\ yC \end{pmatrix} \\
\begin{pmatrix} xF \\ yF \end{pmatrix} \leftarrow \begin{pmatrix} \cos(-\beta) & -\sin(-\beta) \\ \sin(-\beta) & \cos(-\beta) \end{pmatrix} \cdot \begin{pmatrix} xD - xC \\ yD - yC \end{pmatrix} + \begin{pmatrix} xC \\ yC \end{pmatrix} \\
\begin{pmatrix} xC & yC \\ xE & yE \\ xF & yF \end{pmatrix}
\end{array}
\right.
\end{aligned}
$$

Note algoritm *Fulg*

- acest algoritm realizează construcția unei unitati de fulg ce prezintă un segment [AB] și un punct C de unde pleacă două ramuri simetrice în punctele E și F
- A și B sunt punctele segmentului de la care se porneste
- C este punctul de pe segmentul [AB] de unde pleacă ramurile
- se calculează pozitia punctului D în funcție de poziția sa în cadran, D fiind un punct intermediar de pe segmentul [AB] pentru a putea calcula punctele E și F
- se calculează poziția punctelor E si F ale ramurilor care se obțin prin rotația punctului D în jurul punctului C, la un unghi ales, și în ambele sensuri, sus și jos.

Note algoritm *Fulg și Iterare*

- fiecare linie are un punct C de unde pleacă alte două linii, punctul C se determină prin calcul geometric. Dispunerea segmentelor prezintă o simetrie față de un alt segment până la formarea unei ramuri. Fiecare două segmente dispuse simetric se calculează funcție de cel precedent prin iterarea funcției *fulg()*. Lungimea segmentelor *d* se calculează pe baza unui coeficient de contracție.
- aceste *if-uri* s-au introdus pentru pentru a evita cazul *1/0*, folosinduse notații pentru a evita infiniții

Algoritm Simetrie

$$
\begin{aligned}
&\text{simetrie}(S, N, c, k, \beta) := \\
&\quad \text{for } s \in 0 .. S-1 \\
&\qquad \alpha \leftarrow s \cdot 2 \cdot \pi \cdot S^{-1} \\
&\qquad \text{if } s = 0 \\
&\qquad\quad XC \leftarrow \left[\left(\text{iter}(\alpha, N, c, k, \beta)^{T}\right)^{\langle 0 \rangle}\right]_{0} \\
&\qquad\quad YC \leftarrow \left[\left(\text{iter}(\alpha, N, c, k, \beta)^{T}\right)^{\langle 0 \rangle}\right]_{1} \\
&\qquad\quad XE \leftarrow \left[\left(\text{iter}(\alpha, N, c, k, \beta)^{T}\right)^{\langle 1 \rangle}\right]_{0} \\
&\qquad\quad YE \leftarrow \left[\left(\text{iter}(\alpha, N, c, k, \beta)^{T}\right)^{\langle 1 \rangle}\right]_{1} \\
&\qquad\quad XF \leftarrow \left[\left(\text{iter}(\alpha, N, c, k, \beta)^{T}\right)^{\langle 2 \rangle}\right]_{0} \\
&\qquad\quad YF \leftarrow \left[\left(\text{iter}(\alpha, N, c, k, \beta)^{T}\right)^{\langle 2 \rangle}\right]_{1} \\
&\qquad \text{if } s > 0 \\
&\qquad\quad XC \leftarrow \text{stack}\left[XC, \left[\left(\text{iter}(\alpha, N, c, k, \beta)^{T}\right)^{\langle 0 \rangle}\right]_{0}\right] \\
&\qquad\quad YC \leftarrow \text{stack}\left[YC, \left[\left(\text{iter}(\alpha, N, c, k, \beta)^{T}\right)^{\langle 0 \rangle}\right]_{1}\right] \\
&\qquad\quad XE \leftarrow \text{stack}\left[XE, \left[\left(\text{iter}(\alpha, N, c, k, \beta)^{T}\right)^{\langle 1 \rangle}\right]_{0}\right] \\
&\qquad\quad YE \leftarrow \text{stack}\left[YE, \left[\left(\text{iter}(\alpha, N, c, k, \beta)^{T}\right)^{\langle 1 \rangle}\right]_{1}\right] \\
&\qquad\quad XF \leftarrow \text{stack}\left[XF, \left[\left(\text{iter}(\alpha, N, c, k, \beta)^{T}\right)^{\langle 2 \rangle}\right]_{0}\right] \\
&\qquad\quad YF \leftarrow \text{stack}\left[YF, \left[\left(\text{iter}(\alpha, N, c, k, \beta)^{T}\right)^{\langle 2 \rangle}\right]_{1}\right] \\
&\qquad \begin{pmatrix} \text{stack}\left(XC^{T}, XE^{T}\right) & \text{stack}\left(YC^{T}, YE^{T}\right) \\ \text{stack}\left(XC^{T}, XF^{T}\right) & \text{stack}\left(YC^{T}, YF^{T}\right) \end{pmatrix}
\end{aligned}
$$

Note algoritm simetrie

- algoritmul de mai jos realizează simetria radială a fulgului pentru S direcții date
- α este un multiplu de unghiuri egale calculat funcție de cele S directii date, *s* este indice de multiplicare corespunzător unghiului α
- punctele C, E, F sunt iterate pentru fiecare valoare a lui α, apoi pentru s>=2 are loc lipirea matricilor coloana cu funcția *stack* pentru a da posibilitatea afisarii linilor separat
- nu se lipesc vectorii deoarece există doar un singur unghi (o singură coloană pe matrice) pentru care se calculează punctele
- pentru condiția *s=0* nu există nici ramură, pentru conditia *s=1* există o ramură, iar pentru *s>=2* există ramuri la unghiuri egale între ele, pentru s=2 sunt ramuri la 180 grade, pentru s=3 sunt ramuri la 120 grade