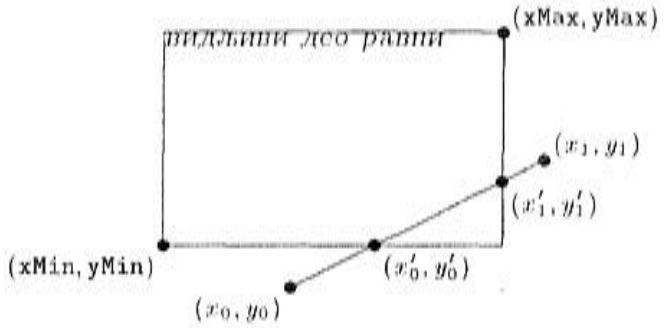
COHEN-SUTHERLAND–ОВ АЛГОРИТАМ ЗА ИСЕЦАЊЕ ВИДЉИВОГ ДЕЛА ДУЖИ

Драган Машуловић, Нови Сад

Приликом цртања дужи, њене крајње тачке не морају припадати видљивом делу равни. Проблем одређивања оног дела дужи који припада том правоугаонику зове се *исецање видљивог дела дужи* (енгл. clipping). Наивно решење овог проблема се своди на то да се приликом цртања дужи за сваку тачку посебно проверава да ли припада видљивом делу равни или не. Cohen-Sutherland–ов алгоритам израчунава крајње тачке видљивог дела дужи, па да се тако добијена (евентуално скраћена) дуж нацрта позивом процедуре `Line` која ништа не проверава.

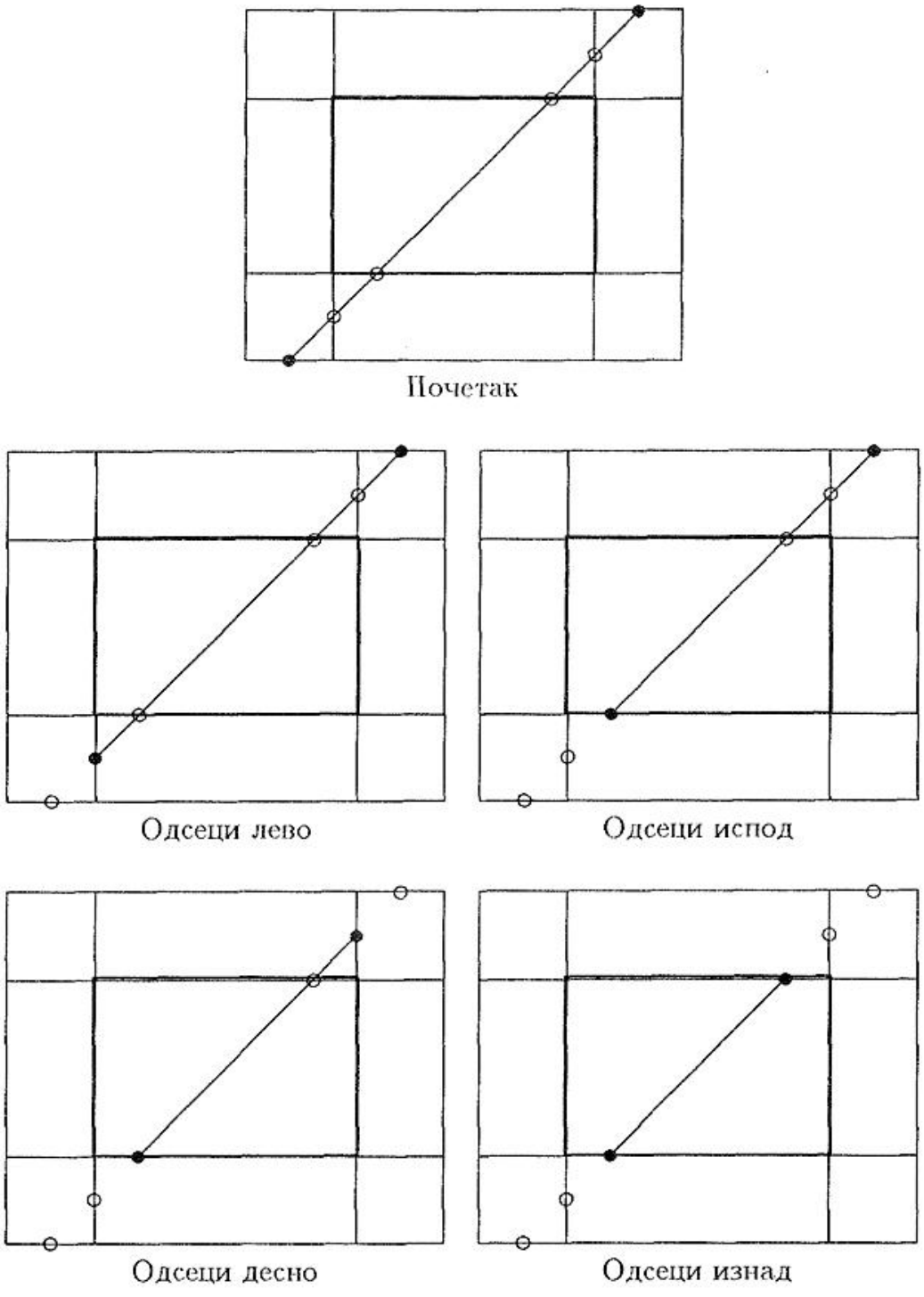


Идеја Cohen-Sutherland–овог алгоритма је да се полазна дуж "окреше" тако што се одсече део који је сувише лево, потом део који је сувише испод, онда део који је сувише десно и на крају део који је сувише изнад, Сл. . Оно што остане, ако уопште нешто остане, је видљиви део дужи. Напоменимо да редослед одсецања делова који штрче записи од положаја дужи према видљивом делу равни и не мора се поклапати са наведеним редоследом.

Реализација Cohen-Sutherland–овог алгоритма је нешто суптилнија. Претпоставимо да треба одредити видљиви део дужи $[AB]$. Свакој од крајњих тачака дужи се додели подскуп скупа {`LEFT`, `RIGHT`, `BOTTOM`, `TOP`} који описује положај те тачке према правим $x =$ `xMin`, $x =$ `xMax`, $y =$ `yMin`, $y =$ `yMax`, редом, како је то показано на слици испод. При томе, ако се тачка налази у видљивом делу равни, додељен јој је празан скуп.

`LEFT` `TOP`	`TOP`	`RIGHT` `TOP`
`LEFT`		`RIGHT`
`LEFT` `BOTTOM`	`BOTTOM`	`RIGHT` `BOTTOM`

Нека је тачки A додељен скуп $\mathcal{O}_A$, а тачки B скуп $\mathcal{O}_B$. Рад алгоритма се заснива на следећим чињеницама:



Сл. 1. Фазе Cohen-Sutherland-овог алгоритма

- Ако је $\mathcal{O}_A = \mathcal{O}_B = \emptyset$, обе тачке се налазе у видљивом делу равни. Зато је цела дуж видљива, и процес се завршава прихватањем дужи.
- Ако је $\mathcal{O}_A \cap \mathcal{O}_B \neq \emptyset$, тада се обе тачке налазе у истој полуравни која је садржана у невидљивом делу равни. Зато је цела дуж невидљива, и процес се завршава одбацивањем дужи.
- Ако није детектован ниједан од претходна два случаја, онда је бар један од скупова $\mathcal{O}_A$, $\mathcal{O}_B$ непразан. Рецимо, $\mathcal{O}_A \neq \emptyset$. Уочимо произвољан елемент скупа $\mathcal{O}_A$ и одсечемо одговарајући део дужи. Поново одредимо скуп $\mathcal{O}_A$ и вратимо се на почетак.

Следи процедура која имплементира овај алгоритам.

```
VAR
    (* koordinate donjeg levog i gornjeg desnog temena vidljivog *)
    (* dela ravni, ali kao realni brojevi                         *)

    rxMin, ryMin, rxMax, ryMax : REAL;


(* glavna procedura                                                  *)
(* x0, y0, x1, y1  su krajnje tacke duzi; na kraju ove promenljive *)
(*                 sadrze koordinate skracene duzi                   *)
(* visible = FALSE ako je cela duz nevidljiva; tada su vrednosti   *)
(*                 promenljivih x0, y0, x1, y1 nedefinisane          *)
(* visible = TRUE  ako je neki deo polazne duzi vidljiv              *)

PROCEDURE ClipLine(VAR x0, y0, x1, y1 : INTEGER; VAR visible : BOOLEAN);
TYPE
    Pos = (LEFT, RIGHT, BOTTOM, TOP);
    Positions = SET OF Pos;

VAR
    x : ARRAY [0 .. 1] OF REAL;
    y : ARRAY [0 .. 1] OF REAL;
    outCode : ARRAY [0 .. 1] OF Positions;
    i : INTEGER;

    (* procedura odredjuje skup atributa date tacke *)

    PROCEDURE ComputeOutCode(x, y : REAL; VAR c : Positions);
    BEGIN
        c := [];
        IF      y > ryMax THEN c := [TOP]
        ELSE IF y < ryMin THEN c := [BOTTOM];

        IF      x > rxMax THEN c := c + [RIGHT]
        ELSE IF x < rxMin THEN c := c + [LEFT]
    END;

BEGIN
```

```
    x[0] := x0;  x[1] := x1;
    y[0] := y0;  y[1] := y1;
    ComputeOutCode(x[0], y[0], outCode[0]);
    ComputeOutCode(x[1], y[1], outCode[1]);
    visible := (outCode[0] = []) AND (outCode[1] = []);

    WHILE NOT visible AND (outCode[0] * outCode[1] = []) DO BEGIN
        (* odredimo krajnju tacku duzi koja je izvan vidljivog dela *)
        IF outCode[0] <> [] THEN  i := 0
        ELSE                      i := 1;

        (* skratimo duz *)
        IF TOP IN outCode[i] THEN BEGIN
            x[i] := x[0] + (x[1] - x[0])*(ryMax - y[0])/(y[1] - y[0]);
            y[i] := ryMax
        END
        ELSE IF BOTTOM IN outCode[i] THEN BEGIN
            x[i] := x[0] + (x[1] - x[0])*(ryMin - y[0])/(y[1] - y[0]);
            y[i] := ryMin
        END
        ELSE IF RIGHT IN outCode[i] THEN BEGIN
            y[i] := y[0] + (y[1] - y[0])*(rxMax - x[0])/(x[1] - x[0]);
            x[i] := rxMax
        END
        ELSE (* LEFT *) BEGIN
            y[i] := y[0] + (y[1] - y[0])*(rxMin - x[0])/(x[1] - x[0]);
            x[i] := rxMin
        END;
        ComputeOutCode(x[i], y[i], outCode[i])
    END;

    IF visible THEN BEGIN
        x0 := ROUND(x[0]); x1 := ROUND(x[1]);
        y0 := ROUND(y[0]); y1 := ROUND(y[1])
    END
END;
```

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