Laborator 6 și 7

1 Obiective

Acest laborator prezintă principiile de funcționare ale algoritmilor de decupare a primitivelor grafice. Doi algoritmi pentru decuparea liniilor sunt prezentați: Cyrus-Beck și Cohen-Sutherland.

2 Algoritmi pentru decuparea liniilor

Algoritmii de decupare sunt folosiți pentru a elimina valorile din afara unui anumit domeniu, adică părți ale segmentelor sau poligoanelor care se află în afara zonei de afișare. În majoritatea cazurilor, zona de afișare este definită ca un dreptunghi și se numește fereastră de decupare. În raport cu această fereastră de decupare, o primitivă (punct, linie sau poligon) poate fi în una dintre următoarele relații:

- În întregime în interiorul ferestrei de decupare - nu decupăm primitiva, se continuă afișarea.
- În întregime în exteriorul ferestrei de decupare - nu decupăm primitiva, se renunță la aceasta.
- Intersectează fereastra de tăiere – se calculează punctele de intersecție și se actualizează primitiva, continuă afișarea.

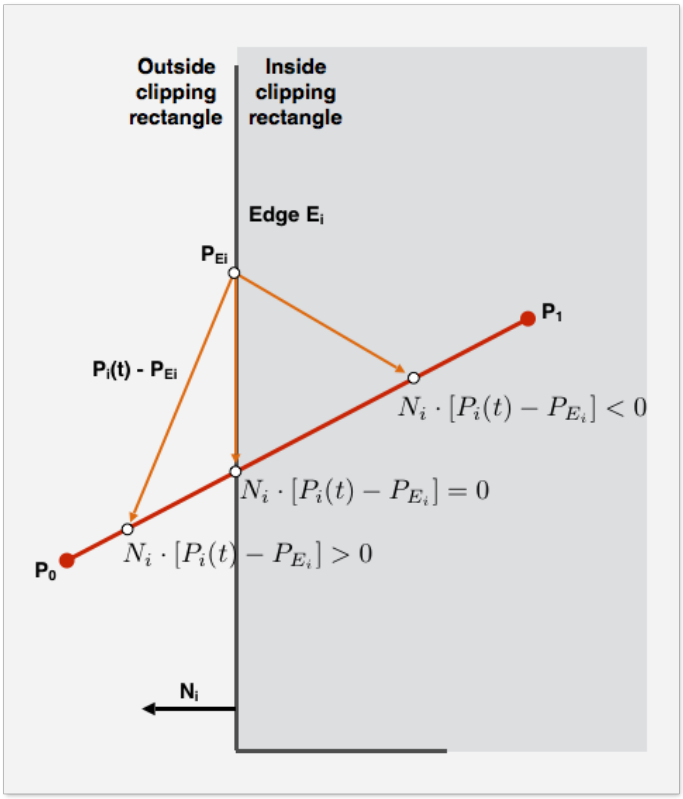
3 Algoritmul de decupare Cyrus-Beck

Algoritmul Cyrus-Beck este folosit pentru a decupa un segment de dreaptă după conturul unui poligon convex. Segmentul de dreaptă este definit parametric sub forma:

$P(t) = P_0 + (P_1 - P_0)t$, where $t \in [0, 1]$

Pentru fiecare muchie E_i calculăm vectorul normală astfel încât acesta să fie orientat înspre exterior (precum în figura alăturată). Pentru fiecare muchie se alege un punct P_{Ei}. Calculând produsul scalar $N_i \cdot (P(t) - P_{Ei})$ putem identifica poziția relativă a fiecărui punct din segmentul de dreaptă (definit pentru o anumită valoare a parametrului **t**):

- Dacă valoarea este **negativă**, punctul se află în semiplanul interior
- Dacă valoarea este **pozitivă**, punctul se află în semiplanul exterior
- Dacă valoarea este **zero**, punctul se află pe muchia ferestrei de decupare



Suntem inteeresați de valorile lui **t** pentru care $N_i \cdot (P(t) - P_{Ei}) = 0$.

Mai întâi înlocuim P(t) și alegem un factor comun:

$$N_i \cdot (P_0 + (P_1 - P_0)t - P_{Ei}) = 0$$

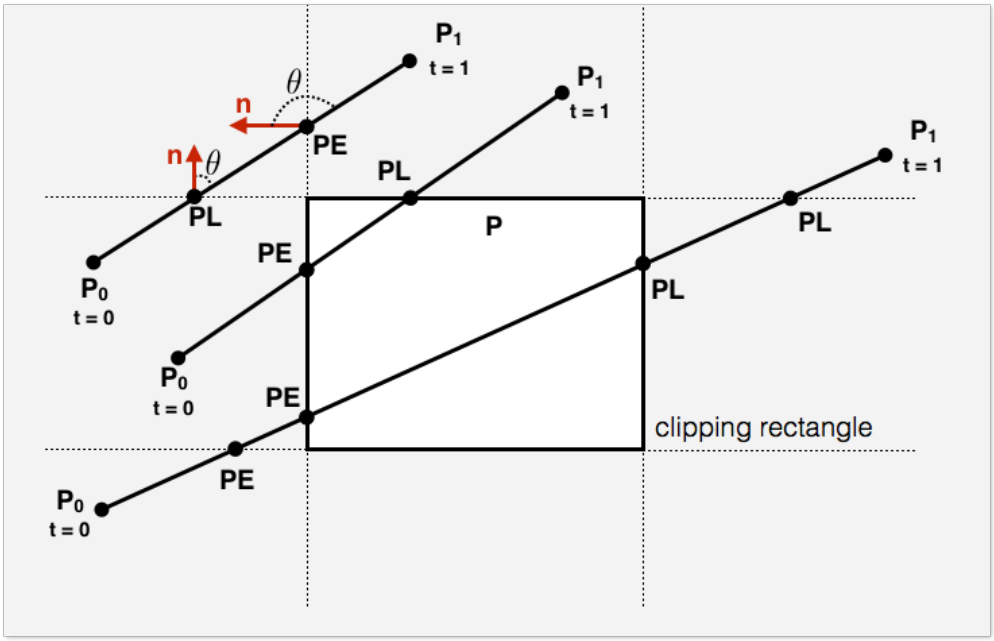
$$N_i \cdot (P_0 - P_{Ei}) + N_i \cdot (P_1 - P_0)t \ = 0$$

Fie $D = P_1 - P_0$ vectorul definit de punctele P_0 și P_1. În acest caz, putem calcula **t** ca:

$$t = \frac{N_i \cdot (P_0 - P_{Ei})}{-N_i \cdot D}$$

Trebuie sa calculăm valoarea lui **t** (a punctului de intersecție) pentru fiecare muchie a poligonului de decupare. Valorile lui **t** din afara intervalului [0, 1] sunt ignorate. Fiecare intersecție este caracterizată prin calculul unghiului dintre P_0P_1 și N_i ca:

- "potențial intrare" (PE) – dacă unghiul > 90°
- "potențial ieșire" (PL) – dacăunghiul < 90°



3.1 Pseudocod

```
precalculeaza Ni si selecteaza un PEi pentru fiecare muchie;
if (P1 = P0)
   linia degenereaza intr-un punct, deci facem decuparea unui punct;
else
   tE = 0; tL = 1;
   for (fiecare candidat calculeaza intresectia cu o muchie de decupare){
      if (Ni * D != 0){
         calculeaza t;
```

```
         foloseste semnul expresiei Ni * D pentru a categoriza punctul ca PE
(potential intrare) sau PL (potential iesire);
         if (PE)
            tE = max(tE, t);
         if (PL)
            tL = min(tL, t);
      }
   }
   if (tE > tL)
      return -1
   else
       return P(tE ) si P(tL ) ca intersectii de decupare
```

4 Algoritmul de decupare Cohen-Sutherland

4.1 Determină daca un punct P(x,y) este vizibil

Considerând P1(x_{min}, y_{min}), și P2(x_{max}, Y_{max}) punctele care definesc dreptunghiul zonei vizibile, punctul P(x,y) este vizibil numai dacă sunt îndeplinite următoarele condiții:

$$x_{min} \leq x \leq x_{max}$$

$$y_{min} \leq y \leq y_{max}$$

4.2 Determină daca un segment este vizibil

Pentru a determina dacă un segment de dreaptă este vizibil, avem nevoie de algoritmi de o complexitate putin mai ridicată. O idee ar fi să testăm vizibilitatea fiecărui punct al segmentului, înainte de a fi afișat pe ecran. Dar această metodă va necesita mult timp și foarte multe calcule. Metoda poate fi ușor îmbunătățită prin testarea mai întâi a capetelor segmentului. Dacă ambele puncte sunt în zona vizibilă, întregul segment va fi vizibil. Acest caz se numește "acceptare simplă". Pe aceeași logică, dacă ambele puncte sunt în afara și pe aceeași parte a zonei vizibile, nici o parte a segmentului nu va fi vizibilă. Acest caz este numit "respingere simplă". În toate celelalte cazuri, trebuie să folosim alți algoritmi pentru a stabili care parte a segmentului (dacă există) este vizibilă.

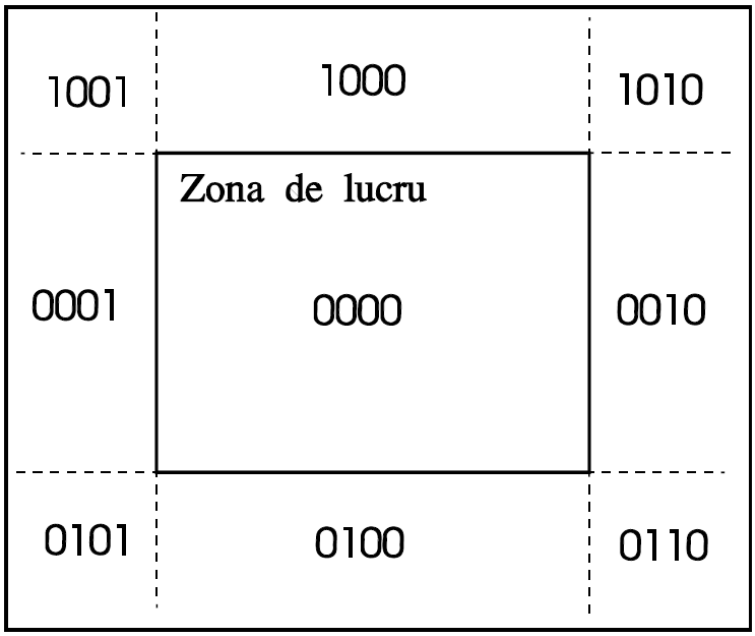
Dacă un segment de linie nu poate fi inclus în cazuri de “acceptare simplă” sau "respingere simplă", atunci trebuie să calculăm punctele sale de intersecție cu următoarele drepte:

$$y = y_{max}, x = x_{max}, y = y_{min}, x = x_{min}$$

și să eliminăm segmentele care sunt plasate în afara zonei vizibile. Ca rezultat, vom obține un nou segment de linie. Algoritmul se repetă până când segmentul rezultat poate fi inclus într-unul din cazurile de “acceptare simplă” sau "respingere simplă".

Algoritmul Cohen-Sutherland folosește un cod de patru cifre pentru a descrie fiecare capăt de segment. Codul are următoarea structură:

- prima cifră este 1 dacă punctul este deasupra zonei vizibile; altfel este 0
- a doua cifră este 1 dacă punctul este sub zona vizibilă; altfel este 0
- a treia cifră este 1 dacă punctul este în partea dreaptă a zonei vizibile; altfel este 0
- a patra cifră este 1 dacă punctul este în partea stângă a zonei vizibile; altfel este 0



4.3 Pseudocod

```
repeat until FINISHED = TRUE
{
   COD1 = computeCScode(x1, y1)    // calculeaza codul de 4 cifre pentru P(x1, y1)
   COD2 = computeCScode(x2, y2)    // calculeaza codul de 4 cifre pentru P(x2, y2)
   RESPINS = SimpleRejection(COD1, COD2)    // testeaza cazul de respingere simpla
   if RESPINS = TRUE
      FINISHED = TRUE
   else
   {
      DISPLAY = SimpleAcceptance(COD1, COD2)    // testeaza cazul de acceptare simpla
      if DISPLAY = TRUE
         FINISHED = true
      else
      {
         if(P(x1, y1) is inside the display area)
            invert(x1,y1,x2,y2,COD1,COD2)    // daca P(x1, y1) este in interiorul zonei de afisare, inverseaza
                                                P(x1, y1) si P(x2, y2) impreuna cu codurile de 4 cifre

         if(COD1[1] = 1) and (y2 <> y1)    // elimina segmentul aflat deasupra zonei de afisare
         {
            x1 = x1+(x2-x1)*(Ymax-y1)/(y2-y1)
            y1 = Ymax
         }
         elseif(COD1[2] = 1) and (y2 <> y1)    // elimina segmentul aflat sub zona de afisare
         {
            x1 = x1+(x2-x1)*(Ymin-y1)/(y2-y1)
            y1 = Ymin
         }
         elseif(COD1[3] = 1) and (x2 <> x1)    // elimina segmentul aflat la dreapta zonei de afisare
         {
```

```
            y1 = y1+(y2-y1)*(Xmax-x1)/(x2-x1)
            x1 = Xmax
        }
        elseif(COD1[4] = 1) and (x2 <> x1)    // elimina segmentul aflat la stanga zonei de afisare
        {
            y1 = y1+(y2-y1)*(Xmin-x1)/(x2-x1)
            x1 = Xmin
        }
    }
  }
}
```

5 Temă

- Implementați algoritmul Cyrus-Beck
- Implementați algoritmul Cohen-Sutherland
- Implementați (folosind SDL) o demonstrație interactivă a algoritmilor. Definiți, folosind mouse-ul, fereastra de decupare (un dreptunghi sau un poligon convex) și un segment de linie.