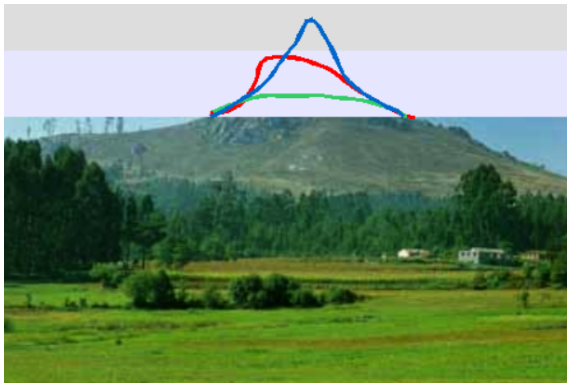
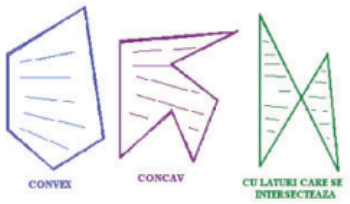
CURS 8 -- Clipping. Rasterizarea poligoanelor

Până acum am ignorat orice legătură între rasterizare și dimensiunile monitorului sau ferestrei destinate aplicației grafice. Clipping este una dintre cele mai importante probleme în Grafica pe calculator și ideea de bază este scrierea primitivelor grafice folosind coordonatele zonei de lucru. Clipping înseamnă a indentifica porțiunile primitivei care sunt vizibile și care nu sunt vizibile. În consecință, clipping înseamnă a ilumina doar pixelii care se află în interiorul suprafeței de desenat, care, în mod frecvent, este un dreptunghi cu axele paralele cu axele de coordonate. A face clipping înseamnă că anumită parte din imagine nu se încadrează în ecran. De exemplu în imaginea de mai jos: Este ca și cum ai fotografia un munte căruia îi lipsește vârful din imagine: nu mai poți spune care a fost forma lui reală, sau dimensiunea lui reală, doar inspectând partea inferioară a imaginii. Care este profilul real al acestui munte?



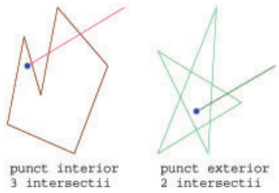
O caracteristică importantă a graficii raster față de grafica vectorială este posibilitatea utilizării elementelor grafice pline, adică solide. Procesul se bazează pe o carență a sistemului vizual uman și anume: integrarea spațială între obiecte pe care ochiul nu este în stare să le distingă din cauza absenței rezoluției. Numărul receptorilor vizuali fiind limitat, dacă se desenează elemente grafice distincte dar suficient de aproape (cum ar fi doi pixeli pe un monitor CRT) - ochiul va putea să le perceapă ca pe un singur obiect. O mulțime de pixeli de aceeași culoare, unul lângă altul, va fi așadar văzută, de către ochiul uman, ca o singură figură colorată uniform. Se consideră un poligon dat prin intermediul vârfurilor sale. Să se deseneze pixelii care aparțin acestuia. Va trebui să individualizăm secvențele orizontale de pixeli conținute în primitivă, algoritm numit de scan-line. Ideea este de a deplasa o linie orizontală din poziția cea mai de jos care intersectează poligonul către poziția cea mai de sus (cu o unitate la fiecare pas) și de a aplica un algoritm de scan-conversion pe linie, de fiecare dată. Pentru a desena un dreptunghi plin, cu laturile paralele cu axele de coordonate, vor trebui iluminați

pixelii interiori dreptunghiului
for y from y_{min} to y_{max} of the rectangle (y scan-line)
for x from x_{min} to x_{max} (pixel cu pixel)
putpixel (x, y, color)
Ne dorim să construim un algoritm generic capabil să facă 'filling' pentru poligoane de orice tip: concave, convexe, cu laturi care se intersectează:

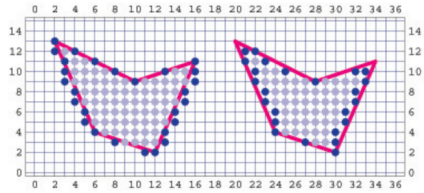


Se află un punct în interiorul unui poligon?

- dintr-un punct arbitrar P se desenează o semidreaptă
- se numără intersecțiile cu laturile
- punctul P este interior dacă numărul de intersecții este impar.



Modalitatea cea mai la îndemână de a calcula intersecțiile este să se utilizeze scan-conversion pentru liniile corespunzătoare fiecărei laturi a poligonului. Această aproximație poate să producă rezultate incorecte deoarece algoritmul de rasterizare pentru linii nu are noțiunea conceptului de înăuntru (interior) și în afară (exterior) și astfel pot fi selecționați pixeli care se afle în afara poligonului. Aceasta este în particular de nedorit când avem de convertit poligoane de culori diferite cu laturi comune: în această manieră vor exista pixeli ai unui poligon care invadează alt poligon generând astfel un efect vizual incorect.



Operația de filling pentru o singură scan-line constă în 3 pași:
1. Se calculeaza intersectiile dintre scan-line şi toate laturile poligonului.
2. Se ordonează intersecţiile după abscisa x.
3. Se aprind pixelii între perechi de intersecţii care sunt interioare poligonului, utilizând, pentru a determina care pixel este interior, aşa-numita regulă odd-parity care spune:

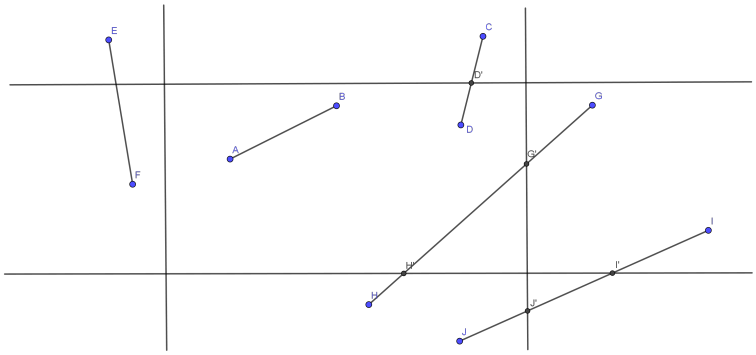
1. paritatea iniţială este pară;
2. fiecare intersecţie schimbă paritatea;
3. se desenează pixelii când paritatea este impară;
4. nu se desenează nimic când paritatea este pară.

1 Clipping pentru un punct

Presupunem că fereastra dreptunghiulară are vârful din stânga-jos de coordonate (x_{min}, y_{min}) iar pe cel din dreapta-sus de coordonate (x_{max}, y_{max}). Pentru ca un punct $P(x, y)$ să fie vizibil în fereastră trebuie satisfăcute relațiile:

$$x_{min} \leq x \leq x_{max}, y_{min} \leq y \leq y_{max}.$$

Un segment de dreapta poate fi față de o fereastră situat astfel:



În această figură, se pot observa segmentele de dreaptă (sau fragmentele din ele) care sunt încadrate în interior respectiv exterior. Astfel,
—în interior avem segmentul AB,
— în exterior avem segmentele EF şi IJ,
— un capăt în interior și unul în exterior CD,
— ambele capete în exterior dar o parte a segmentului în interior GH.
Un algoritm simplu este următorul:
• dacă ambele capete ale segmentului sunt în interior, se trasează segmentul;
• dacă numai un capăt este în interior (și celălalt în exterior) se determină punctul de intersecție cu marginea ferestrei și se trasează subsegmentul de interior;
• altfel, se calculează intersecțiile cu liniile ce definesc marginea ferestrei și, dacă punctele de intersecție sunt pe segmentele de dreaptă ce definesc frontiera ferestrei, se trasează subsegmentul dintre acestea.

Dacă o linie nu este nici respinsă și nici acceptată atunci se calculează, într-o ordine prestabilită, intersecțiile liniei cu dreptele suport ale laturilor ferestrei de lucru: $y = y_{max}, x = x_{max}, y = y_{min}, x = x_{min}$, îndepărtând porțiunile de segment care se situează în afara ferestrei dreptunghiulare. Rezultă astfel un nou segment, iar procedura se bucleză asupra ei însă și până când un segment rezultat este acceptat sau respins. Algoritmul care urmează propune ca ecranul să se împartă în 9 regiuni determinate de marginile ferestrei. În prezentarea clasică a algoritmului Cohen - Sutherland se asociază fiecărui capăt al unui segment un cod de 4 cifre și anume:

1001	**1000**	**1010**
0001	**0000**	**0010**
0101	**0100**	**0110**

a) prima cifră este → 1 dacă punctul este deasupra ferestrei
→ 0 în rest

b) a doua cifră este → 1 dacă punctul este sub fereastră
→ 0 în rest

c) a treia cifră este → 1 dacă punctul este în dreapta ferestrei
→ 0 în rest

d) a patra cifră este → 1 dacă punctul este în stânga ferestrei
→ 0 în rest .

Puterea algoritmului constă în faptul că, odată determinate codurile asociate capetelor segmentului, testele de acceptare sau respingere se realizează prin operații logice rapide. Dacă ambele capete ale segmentului au codul 0000 atunci, în mod

banal, se poate decide că segmentul se află în întregime în interiorul ferestrei. Acesta este cazul segmentului (AB) din figură. De asemenea, în mod banal se poate decide că un segment este în întregime în exterior când operația AND logic între codurile celor două puncte conduce la un rezultat diferit de 0. Într-adevăr, în acest caz, ambele puncte se află într-un același semiplan (cel indentificat de bitul 1 al rezultatului) și astfel segmentul nu intersectează dreptunghiul de clipping. Este cazul segmentului (EF) din figură. Avem: $F(0001)$ și E(1001) și astfel

$$0001 \wedge 1001 = 0001 \neq 0000.$$

Dacă însă rezultatul operației AND este 0000 se testează care din cele două puncte extreme se află în exteriorul dreptunghiului (cel puțin unul este!!!) și se împarte segmentul în două părți, din care una poate fi înlăturată (deoarece se află în exteriorul dreptunghiului). Să luăm cazul segmentului (CD) din figură: D(1000) și C(0000). Dreapta cu care se calculează intersecția este cea care corespunde bitului 1 din codul punctului. Pentru D se calculează intersecția D' cu dreapta $y = y_{max}$. Noul punct D' îl va substitui pe precedentul D. Se scrie din nou codul și se reia procedeul.

2 BIBLIOGRAFIE

1. Piscoran Laurian, Piscoran Ioan, Elemente de geometrie analitica si diferentiala, Ed. Risoprint, 2010, Cluj Napoca.
2.https://mathworld.wolfram.com/BarycentricCoordinates.html