Problema liniilor si suprafetelor ascunse în grafica 3D. Algoritmul *PAINTER*

Conf.dr. Felix FURTUNA
Catedra de Informatica Economica, A.S.E. Bucuresti

The painter's algorithm gets its name from the manner in which an oil painting is created. The artist begins with the background and fills the entire canvas with the background scene. Hidden surfaces can be covered up by choosing the correct order to draw them. We must compare each surface with all of the rest to see which is in front of which. We must, in effect, sort the surfaces to determine a priority for their display. The sorting will determine the order in which they will be drawn. The goal of this paper is an implementation of this algorithm using sample MFC graphic functions (GDI). The mathematical compute is based by vector cross product and vector dot product.
***Keywords**: hidden surfaces, polygon, topological sorting, cross product, dot product.*

Numele algoritmului provine de la similitudinea care exista între ideea algoritmului si modul în care un pictor zugraveste o scena, desenând mai întâi planurile îndepartate apoi pe cele apropiate. În acest fel primele desene sunt acoperite de cele efectuate ulterior, ramânând la vedere doar portiunile vizibile din locul din care este privita scena. Etapele în care se desfasoara algoritmul sunt:
1. eliminarea suprafetelor ascunse;
2. stabilirea unor prioritati pentru suprafetele vizibile;
3. trasarea suprafetelor conform prioritatilor.

Eliminarea suprafetelor ascunse
Pentru eliminarea suprafetelor ascunse se utilizeaza metoda testului de vizibilitate [2]. Sunt vizibile acele suprafete pentru care produsul scalar dintre vectorul normal la suprafata si vectorul de vizualizare este pozitiv.

Stabilirea prioritatilor pentru suprafetele vizibile
Stabilirea prioritatilor se face dupa cota *Z* a fiecarei suprafete. Cota *Z* este data de cea mai mare si de cea mai mica valoare a coordonatelor dupa axa O*Z*:

$$\text{MinZ} = Min\{z_i, i = 1, n\}$$
$$\text{MaxZ} = Max\{z_i, i = 1, n\}$$

unde *n* este numarul de puncte al suprafetei pentru care se calculeaza cota.

Prioritatile vor juca rolul unor ranguri cu ajutorul carora suprafetele vor fi sortate fie utilizând un algoritm de sortare prin numarare, fie un algoritm de sortare topologica. Astfel, pentru oricare doua suprafete P si Q se va efectua o succesiune de teste în urma carora rezulta care din fete are prioritatea (rangul) mai mare:
Testul 1. Se noteaza cu MaxZ(P), MinZ(P), MaxZ(Q) si MinZ(Q) cotele maxime si minime *z* pentru cele doua suprafete. Daca MaxZ(Q)<MinZ(P), atunci fata Q este prioritara, iar daca MaxZ(P)<MinZ(Q), fata *P* este prioritara (figura 1).

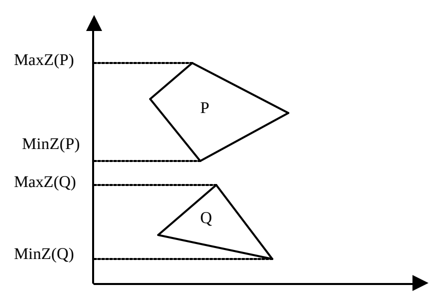


Fig.1. Testul 1

Testul 2. Daca în urma aplicarii testului 1 nu se poate stabili clar prioritatea unei fete se trece la testul 2, care indica existenta suprapunerilor. Se încadreaza fetele în câte o zona rectangulara dupa cotele maxime si minime pe axele OX si OY, (figura 2):

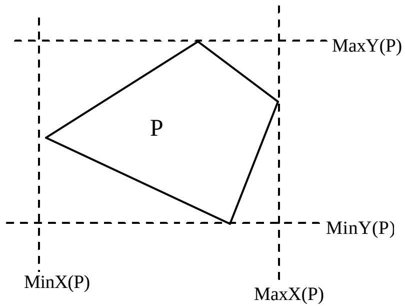


Fig. 2. Zona rectangulara a unei suprafete

Exista suprapuneri atunci când zonele rectangulare ale celor doua suprafete se întrepatrund (figura 3). Daca zonele rectangulare nu se întrepatrund (figura 4), atunci suprafetele pot fi desenate în orice ordine.

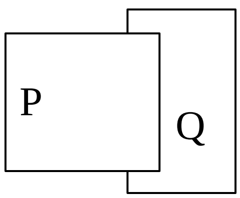


Fig. 3. Suprapuneri

Daca exista suprapuneri se trece la testul 3.

<u>Testul 3</u> Acest test depisteaza *suprapunerile reciproce*. Planul care contine o suprafata împarte spatiul în doua. Daca o suprafata se afla în acelasi spatiu cu observatorul atunci aceasta va avea o prioritate mai mare decât suprafata care face împartirea si invers, când suprafata se afla în celalalt spatiu, ea va avea o prioritate mai mica. O suprafata se afla într-o singura parte a unui plan atunci când toate punctele ei se afla în acea parte. Atunci când are puncte de-o parte si de alta a planului, suprafata este intersectata de plan.

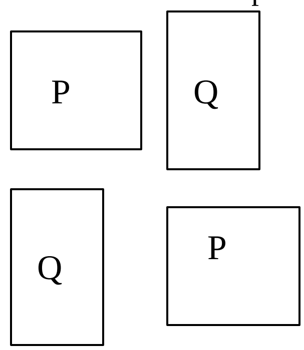


Fig. 4. Fara suprapuneri

Sa presupunem ca vrem sa vedem daca punctele suprafetei *Q* se afla de -o parte sau de alta a planului în care se afla suprafata *P*. Pentru aceasta se determina coeficientii A_P, B_P, C_P, D_P ale ecuatiei planului care contine *P*:

$$A_P x + B_P y + C_P z + D_P = 0 .$$

Forma determinant pentru ecuatia planului ce contine trei puncte de coordonate (x_1, y_1, z_1), (x_2, y_2, z_2) si (x_3, y_3, z_3) este:

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = x \begin{vmatrix} y_1 & z_1 & 1 \\ y_2 & z_2 & 1 \\ y_3 & z_3 & 1 \end{vmatrix} - y \begin{vmatrix} x_1 & z_1 & 1 \\ x_2 & z_2 & 1 \\ x_3 & z_3 & 1 \end{vmatrix} + z \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} - \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix} = 0 .$$

Dezvoltând forma determinant si efectuând identificarea de termeni obtinem:

$$\begin{cases} A_P = y_1(z_2 - z_3) - y_2(z_1 - z_3) + y_3(z_1 - z_2) \\ B_P = -x_1(z_2 - z_3) + x_2(z_1 - z_3) - x_3(z_1 - z_2) \\ C_P = x_1(y_2 - y_3) - x_2(y_1 - y_3) + x_3(y_1 - y_2) \\ D_P = -x_1(y_2 z_3 - y_3 z_2) + x_2(y_1 z_3 - y_3 z_1) - x_3(y_1 z_2 - y_2 z_1) \end{cases}$$

Daca înlocuim cu coordonatele punctelor care alcatuiesc fata *Q* în ecuatia planului mai sus determinata si obtinem numai valori mai mari sau egale cu zero, sau mai mici sau egale cu zero, atunci fata *Q* se afla de o parte sau de cealalta a planului *P*. Pentru valorile 0 punctele sunt chiar pe plan. Daca sunt puncte si de o parte si de cealalta a planului, atunci acesta intersecteaza fata.

Daca nici una dintre cele doua fete nu se afla fie de o parte fie de cealalta a planului care contine cealalta fata, avem suprapuneri reciproce si trecem la testul 4.

<u>Testul 4</u>. În cazul suprapunerilor reciproce se recurge la scindarea uneia dintre fete. Împartirea se va face dupa intersectia dintre plane *P* si *Q* (figura 5).

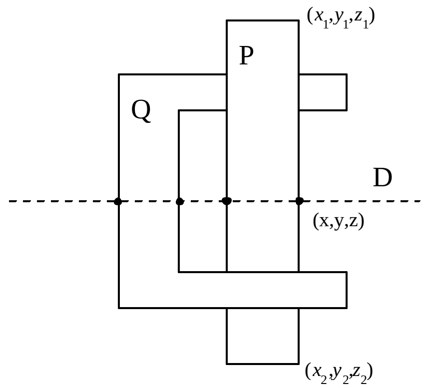


Fig. 5. Suprapunere reciproca

În aceste situatii fie se împarte fata P fie fata Q. Prioritatile fetelor rezultate în raport de fata ramasa pot fi usor stabilite prin testele anterioare.

Intersectia dintre doua plane este o dreapta. Ecuatia carteziana a unei drepte în spatiu care contine doua puncte (x1,y1,z1) si (x2,y2,z2) este:

$$D: \frac{x-x_1}{x_2-x_1} = \frac{y-y_1}{y_2-y_1} = \frac{z-z_1}{z_2-z_1} = v,$$

$$D: \begin{cases} x = x_1 + v(x_2 - x_1) \\ y = y_1 + v(y_2 - y_1). \\ z = z_1 + v(z_2 - z_1) \end{cases}$$

Se determina intersectia acestei drepte cu planul de ecuatie $A_Q x + B_Q y + C_Q z + D_Q = 0$. Înlocuind relatiile D pentru x, y si z în ecuatia planului, obtinem:

$$v = \frac{-A_Q x_1 - B_Q y_1 - C_Q z_1 - D_Q}{A_Q(x_2 - x_1) + B_Q(y_2 - y_1) + C_Q(z_2 - z_1)}.$$

Revenind si înlocuind în ecuatiile D se obtine x, y si z.

Trasarea suprafetelor conform prioritatilor

În urma efectuarii testelor vor fi stabilite prioritati sub forma unor perechi de indici (i,j), cu semnificatia ca fata i este mai îndepartata decât fata j si trebuie trasata prima. Trasarea suprafetelor se va face astfel încât sa fie respectate toate prioritatile (i,j). Pentru aceasta, extragerea suprafetelor se va realiza printr-un algoritm de sortare topologica.

Aplicatie. Consideram urmatorul obiect 3D reprezentând o casa:

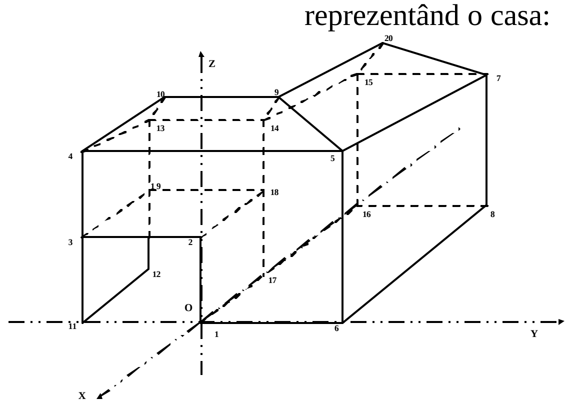


Fig. 6. Exemplu

Datele prin care este descris obiectul sunt memorate într-un fisier text, în felul urmator:

- pe prima linie este trecut n numarul de puncte din care este compus obiectul;
- în linia a doua este trecut m numarul de suprafete;
- pe urmatoarele n linii sunt trecute coordonatele fiecarui punct;
- pe ultimele $2m$ linii sunt trecute pentru fiecare suprafata numarul de puncte din care este compusa aceasta si punctele corespunzatoare.

Pentru casa din figura 6 fisierul arata astfel:

```
20
15
0 0 0
0 0 3
0 -5 3
0 -5 6
0 5 6
```

```
0 5 0
-12 5 6
-12 5 0
-2.5 2.5 7.5
-2.5 -2.5 7.5
0 -5 0
-5 -5 0
```

```
-5 -5 6
-5 0 6
-12 0 6
-12 0 0
-5 0 0
-5 0 3
-5 -5 3
```

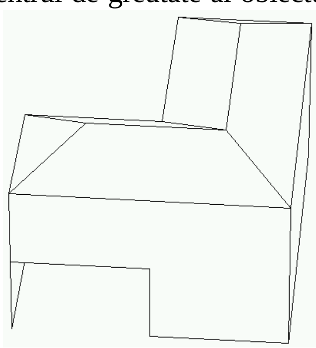
```
-9.5 2.5 7.5
6
6 5 4 3 2 1
4
8 7 5 6
4
16 8 6 1
4
11 4 13 12
4
5 9 10 4
3
4 10 13
4
7 20 9 5
3
15 20 7
4
18 2 3 19
4
13 10 9 14
4
9 20 15 14
6
14 15 16 1 2 18
4
13 14 18 19
4
3 11 12 19
4
15 7 8 16
```

Implementarea algoritmului. S-a ales mediul Visual C si s-au utilizat functii din biblioteca grafica GDI. Iata câteva din procedurile care cuprind elemente esentiale ale implementarii:

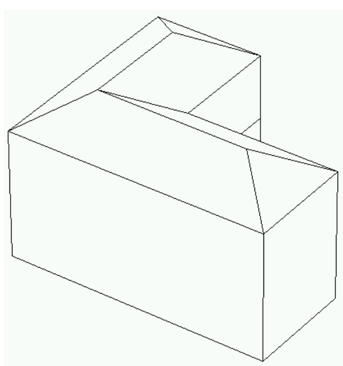
```
void EcuatiaPlanului(double x1,double y1,double z1,double x2,double y2,double z2,
 double x3,double y3,double z3,double &a,double &b,double &c,double &d)
{       a = y1*(z2-z3)-y2*(z1-z3)+y3*(z1-z2);
        b = -x1*(z2-z3)+x2*(z1-z3)-x3*(z1-z2);
        c = x1*(y2-y3)-x2*(y1-y3)+x3*(y1-y2);
        d = -x1*(y2*z3-y3*z2)+x2*(y1*z3-y3*z1)-x3*(y1*z2-y2*z1);
}

void IntersectieDreaptaPlan(double x1,double y1,double z1,double x2,double y2,
        double z2,double a,double b,double c,double d,
        double &x,double &y,double &z)
{       double v = (-a*x1-b*y1-c*z1-d)/(a*(x2-x1)+b*(y2-y1)+c*(z2-z1));
        x=x1+v*(x2-x1);y=y1+v*(y2-y1);z=z1+v*(z2-z1);
}

void Scindare(int i,int j,int np[],int f[][10],double a[],double b[],
        double c[],double d[],double punct[][3],int &m,int &n)
{       int r,ps[10],k=0;
        double x,y,z,x1,y1,z1,x2,y2,z2,v1,v2;
        for(r=0;r<np[i]-1;r++)
        {       x1=punct[f[i][r]][0];
                y1=punct[f[i][r]][1];
                z1=punct[f[i][r]][2];
                v1= a[j]*x1+b[j]*y1+c[j]*z1+d[j];
                x2=punct[f[i][r+1]][0];
                y2=punct[f[i][r+1]][1];
                z2=punct[f[i][r+1]][2];
                v2= a[j]*x2+b[j]*y2+c[j]*z2+d[j];
                if(v1*v2<=0)
                        {//Memorez primul punct din segmentul intersectat
                        ps[k]=f[i][r];
                        IntersectieDreaptaPlan(x1,y1,z1,x2,y2,z2,a[j],b[j],c[j],d[j],x,y,z);
                        punct[n+k][0]=x;punct[n+k][1]=y;punct[n+k][2]=z;
                        k++;
                        }
        }
        int p,l=0,l1=0,l2=0,f1[10],f2[10];
        ps[k]=f[i][0];
        for(p=0;p<k;p++)
                if(p%2==0)
                {       if(p!=0){f1[l1]=n+p-1;l1++;}
                        while(f[i][l]!=ps[p]){f1[l1]=f[i][l];l1++;l++;}
                        f1[l1]=n+p;l1++;
                }
                else
                {       f2[l2]=n+p-1;l2++;
                        while(f[i][l]!=ps[p]){f2[l2]=f[i][l];l2++;l++;}
                        if(p!=k){f2[l2]=n+p;l2++;}
                }
        np[i]=l1;np[m]=l2;m++;
        for(r=0;r<l1;r++)f[i][r]=f1[r];
        for(r=0;r<l2;r++)f[m][r]=f2[r];
}


void Teste(int i,int j,double MaxZf[],double MinZf[],double MaxXf[],double MinXf[],
                  double MaxYf[],double MinYf[],
                  int npi,int npj,int f[][10],
                  double a[],double b[],double c[],double d[],
```

```
                 double o1,double o2,double o3,double p[][3],
                 int &k1,int &k2)
{// in k1 se intoarce fata cea mai indepartata de observator
 // in k2 se intoarce fata cea mai apropiata de observator
       int r;
       double semn,x,y,z,v,ValoareI,ValoareJ;
       k1=-1;
       if(MaxZf[i]<=MinZf[j]){k1=j;k2=i;return;}
       if(MaxZf[j]<=MinZf[i]){k1=i;k2=j;return;}
       if(MaxXf[j]<=MinXf[i] || MaxXf[i]<=MinXf[j])
              if(MaxZf[i]>MaxZf[j]){k1=i;k2=j;return;}else{k1=j;k2=i;return;}
       if(MaxYf[j]<=MinYf[i] || MaxYf[i]<=MinYf[j])return;
              if(MaxZf[i]>MaxZf[j]){k1=i;k2=j;return;}else{k1=j;k2=i;return;}
       int DeOParteI,DeOParteJ;
// Se verifica daca toate punctele fetei i sunt intr-o parte a fetei j
       r=0;
       do
       {      x=p[f[i][r]][0];
              y=p[f[i][r]][1];
              z=p[f[i][r]][2];
              v = a[j]*x+b[j]*y+c[j]*z+d[j];
              r++;
       }
       while(v==0&&r<npi);
       if(r==npi)return;
       ValoareI=v;
       if(v<0)semn=-1.;else semn=1.;
       DeOParteI=1;
       while(r<npi)
       {      x=p[f[i][r]][0];
              y=p[f[i][r]][1];
              z=p[f[i][r]][2];
              v = a[j]*x+b[j]*y+c[j]*z+d[j];
              if(v!=0)
              {      v*=semn;
                     if(v<0){DeOParteI=0;break;}
              }
              r++;
       }
// Clarificare fata i in raport cu observatorul
       if(DeOParteI==1)
       {
              v = a[j]*o1+b[j]*o2+c[j]*o3+d[j];
              if(v!=0)
              {if(v*ValoareI>0){k1=j;k2=i;return;}else {k1=i;k2=j;return;}}
       }

// Se verifica daca toate punctele fetei j sunt intr-o parte a fetei i
       r=0;
       do
       {      x=p[f[j][r]][0];
              y=p[f[j][r]][1];
              z=p[f[j][r]][2];
              v = a[i]*x+b[i]*y+c[i]*z+d[i];
              r++;
       }
       while(v==0&&r<npj);
       if(r==npj)return;
       ValoareJ=v;
       if(v<0)semn=-1.;else semn=1.;
       DeOParteJ=1;
       while(r<npj)
       {      x=p[f[j][r]][0];
              y=p[f[j][r]][1];
              z=p[f[j][r]][2];
              v = a[i]*x+b[i]*y+c[i]*z+d[i];
              if(v!=0)
              {      v*=semn;
                     if(v<0){DeOParteJ=0;break;}
              }
              r++;}
// Se verifica pozitia observatorului in raport cu fata j
       if(DeOParteJ==1)
              {v = a[i]*o1+b[i]*o2+c[i]*o3+d[i];
               if(v!=0){if(v*ValoareJ>0){k1=i;k2=j;return;}else {k1=j;k2=i;return;}}
              }
       k1=-2;}
```

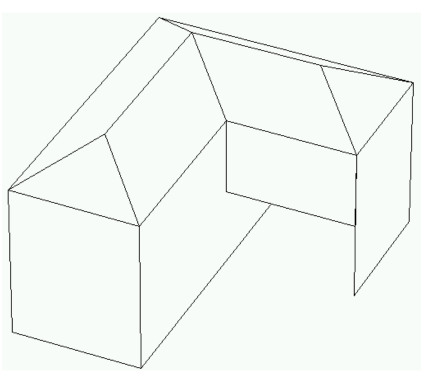
În figura 7 sunt prezentate câteva iesiri obtinute prin rularea aplicatiei. Pozitiile difera prin schimbarea componentelor q si F din coordonatele sferice ale observatorului. Valoarea R a fost aleasa astfel încât observatorul sa se plaseze la o distanta de 10 ori mai mare decât cele mai distantate doua puncte între ele care alcatuiesc obiectul. De asemenea obiectul este centrat printr-o translatie a punctului O, care initial coincide cu punctul 1, în centrul de greutate al obiectului.



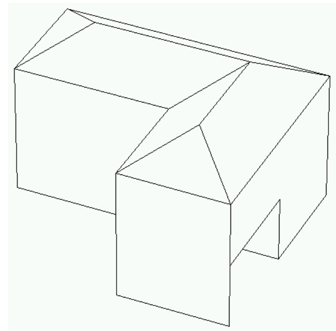
A. Vedere frontala 1 ($\Phi = 45°, \theta = 5°$)



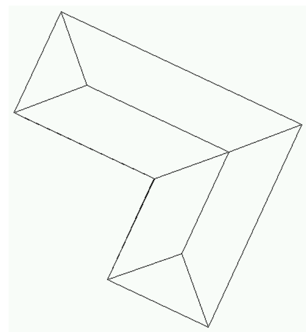
B. Vedere laterala 1 ($\Phi = 45, \theta=135°$)



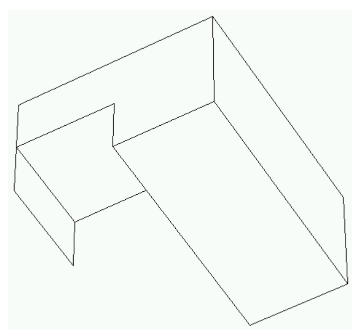
C. Vedere frontala 2 ($\Phi = 45°, \theta = 210°$)



D. Vedere laterala 2 ($\Phi =45°, \theta = 340°$)



E. Vedere de sus ($\Phi = 90°, \theta = 290°$)



F. Vedere de jos ($\Phi = 315°, \theta = 30°$)

Fig. 7.

Bibliografie

1. Furtuna, F., *Problema liniilor si suprafetelor ascunse în grafica 3D. Metoda testului de vizibilitate*, Informatica Economica, nr. 4, Ed. INFOREC, Bucuresti, 2001
2. Furtuna, F., *Problema liniilor si suprafetelor ascunse în grafica 3D. Algoritmul orizontului mobil*, Informatica Economica, nr.1, Ed. INFOREC, Bucuresti, 2002
3. White, D., Scribner, K., Olafsen, E., *MFC Programming with Visual* C++6, SAMS Publishing, USA, 1999
4. Robert Dony, *Calcul des parties cachees*, Masson, Paris, 1990
5. Cunningham, J., *Computer graphics and object oriented programming*, John Wiley & Sons, 1992