Problema liniilor si suprafetelor ascunse în grafica 3D. Algoritmul orizontului mobil

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

Algoritmul orizontului mobil este un algoritm de tip linii ascunse. Prezentarea facuta în continuare este adaptata la suprafete construite prin functii de doua variabile Z = f(x,y). Ideea de baza a algoritmului este aceea de a descompune suprafata într-un anumit numar de linii care ar putea fi obtinute si prin sectionare succesiva. Fie-care linie este reprezentata la rândul ei, printr-o succesiune de puncte.
Keywords: *alghorithm, hiden lines, variables, segments, hiden surface.*

Se noteaza cu *L*0 numarul de linii în care se descompune suprafata si cu *P*0 numarul de puncte din care este construita o linie. Cu cât *L*0 si *P*0 au valori mai mari, cu atât reprezentarea va fi mai exacta. Construirea suprafetei începe cu trasarea liniei aflate cel mai aproape de observator, apoi urmatoarele, în ordine. Suprafata construita va împartii ecranul (fereastra de vizualizare) în orice moment al reprezentarii în trei parti:

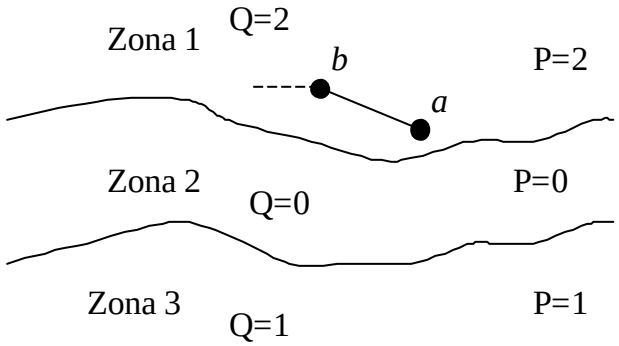
1. O parte care corespunde zonei vizibile aflate deasupra suprafetei reprezentate pâna în acel moment - *zona* 1
2. O parte mascata de suprafata reprezentata pâna în acel moment - *zona* 2 - care este invizibila
3. O parte ce corespunde zonei vizibile aflate sub suprafeta reprezentata pâna în acel moment - *zona* 1

O linie va fi contruita din *P*0-1 segmente mici care se vor trasa consecutiv. Un segment se traseaza integral, partial sau deloc dupa cum cele doua puncte care-l definesc se afla într-o zona vizibila sau în zona invizibila. Se noteaza cu *P* codul punctului final al unui segment. Valorile atasate punctului *P* sunt:

$$P = \begin{cases} 2, & \text{Daca punctul este în zona 1} \\ 0, & \text{Daca punctul este in zona 2} \\ 1, & \text{Daca punctul este in zona 3} \end{cases}$$

Analog, se definesc valorile atasate punctului *Q*, punctul initial al unui segment:

$$Q = \begin{cases} 2, & \text{Daca punctul este în zona 1} \\ 0, & \text{Daca punctul este in zona 2.} \\ 1, & \text{Daca punctul este in zona 3} \end{cases}$$



Linia orizontului superior - LCS

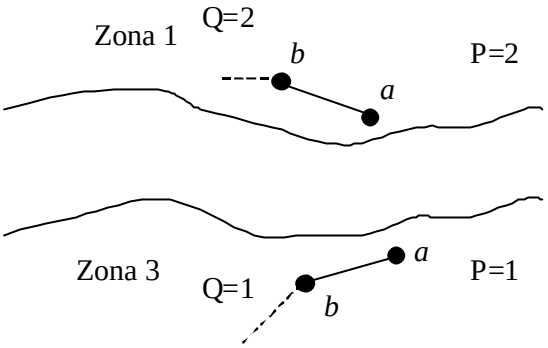
Linia orizontului inferior - LCI

Punctul *b* este deja trasat, are codul asociat *Q* si coordonatele (*A*2,*B*2). Punctul *a* urmeaza a fi trasat odata cu segmentul *ba*, are codul asociat *P* si coordonatele (*A*1,*B*1). Frontiera dintre zona 2 si zona 1 se numeste *orizontul superior* iar frontiera dintre zona 2 si zona 3 se numeste *orizontul inferior*, denumiri care dau titulatura metodei.

Situatiile care se pot întâlni sunt urmatoarele:

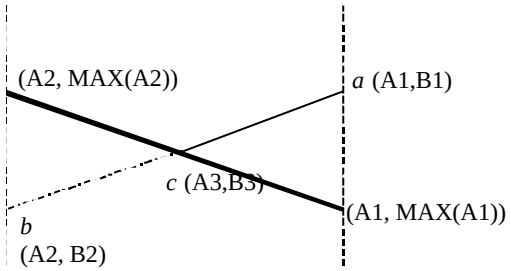
A. P = 2 si Q = 2 sau P=1 si Q=1

Ambele puncte sunt fie în zona 1 fie în zona 3 si segmentul ba este vizibil:



B. P=2 si Q=0 sau P=1 si Q=0 sau P=0 si Q=1 sau P=0 si Q=2

Un punct se afla în zona 2 si celalalt în zona 1 sau zona 3. Segmentul *ba* este partial vizibil si va taia fie orizontul superior fie orizontul inferior. Se calculeaza coordonatele (*A*3,*B*3) ale punctului de intersectie *c* cu una din cele doua linii orizont.



Fiecare punct din liniile ce compun suprafata de reprezentat are un rang de la 1 la P0. Pentru un rang oarecare, *i*, se cunosc coordonatele punctului aflat pe orizontul superior si pe orizontul inferior. În figura de mai sus punctele *b* si *a* care au rangurile *i* si *i*-1 corespund punctelor de frontiera (A2,MAX(A2)) , (A1,MAX(A1)) de pe orizontul superior.

Efectuând calculele legate de intersectia a doua segmente rezulta:

$$A3 = \frac{B1A2 - B2A1 - MAX(A1)A2 + MAX(A2)A1}{MAX(A2) - MAX(A1) - B2 + B1}$$

$$B3 = \frac{B1MAX(A2) - B2MAX(A1)}{MAX(A2) - MAX(A1) - B2 + B1}.$$

Observatie. Daca MAX(A2)-MAX(A1)-B2+B1=0 atunci segmentul *ba* este paralel cu unul din orizonturi, mai precis se afla pe linia orizontului respectiv.

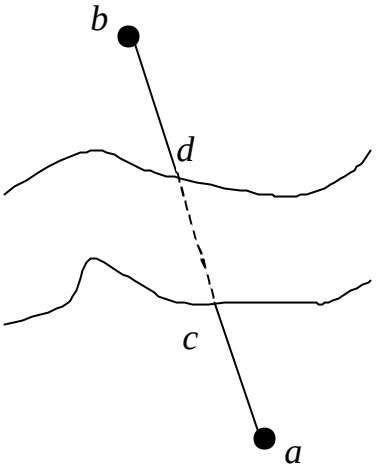
C. P=0 si Q=0

```
void CAomView::OnDraw(CDC* pDC)
{
        CAomDoc* pdoc = GetDocument();
```

Segmentul *ba* se afla în zona 2 si nu se traseaza.

D. P=1 si Q=2 sau P=2 si Q=1

Segmentul traverseaza zona 2, intersectând ambele orizonturi. În aceasta situatie se calculeaza doua intersectii: *c* si *d*.



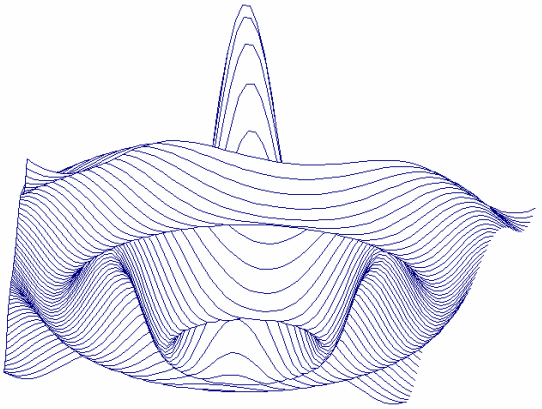
Se utilizeaza pentru trasarea segmentului *ab* interpolarea liniara. Fie *Y* = *MX*+*N*, ecuatia dreptei ce trece prin punctele *a* si *b*. Parametrii *M* si *N* sunt:

$$M = \frac{B2 - B1}{A2 - A1},\ N = \frac{B1A2 - B2A1}{A2 - A1}.$$

Trasarea se va face din aproape în aproape tinând cont de coordonatele punctelor de pe cele doua orizonturi.

Exemplu. Pentru functia:

$Z = f(x,y) = \sqrt{x^2 + y^2}$, suprafata desenata va fi urmatoarea:



Codul functiei `OnDraw` din MFC pentru tra-sarea suprafetei este redat mai jos. În fun-ctia `Scalare` este realizata transformarea de vizualizare.

```
ASSERT_VALID(pdoc);
// TODO: add draw code for native data here
pdoc->pf=fopen("aom.txt","rt");
if(!pdoc->pf){MessageBox("Nu exista fisierul de intrare!");return;}
double a1,b1,a2,b2,tx[100][100],ty[100][100];
int LCS[5000],LCI[5000],VALMAX=10000,VALMIN=-10000;
int nl=50,np=50;
RECT r;
GetClientRect(&r);
int vl=r.left+10,vt=r.top+10,vr=r.right-10,vb=r.bottom-10;
fscanf(pdoc->pf,"%lf",&a1);fscanf(pdoc->pf,"%lf",&b1);
fscanf(pdoc->pf,"%lf",&a2);fscanf(pdoc->pf,"%lf",&b2);
fclose(pdoc->pf);
double z,x,y,pasl=(b1-a1)/nl,pasp=(b2-a2)/np;
double wl=1e20,wr=-1e20,wt=-1e20,wb=1e20,ax,ay,bx,by,xo,yo,zo;
double xr,yr,R=10*sqrt((a1-b1)*(a1-b1)+(a2-b2)*(a2-b2));
double ct=cos(pdoc->teta),st=sin(pdoc->teta);
double cf=cos(pdoc->fi),sf=sin(pdoc->fi);
int i,j,li=r.right-r.left;
for(i=0;i<=r.right;i++){LCS[i]=VALMIN;LCI[i]=VALMAX;}
x=a1;i=0;
while(x<=b1)
{
	y=a2;j=0;
	while(y<=b2)
	{
		z=f(x,y);
		xo=-x*st+y*ct;
		yo=-x*ct*sf-y*st*sf+z*cf;
		zo=-x*ct*cf-y*st*cf-z*sf+R;
		xr=R*xo/zo;yr=R*yo/zo;
		tx[i][j]=xr;ty[i][j]=yr;
		if(xr<wl)wl=xr;
		if(xr>wr)wr=xr;
		if(yr<wb)wb=yr;
		if(yr>wt)wt=yr;
		y+=pasp;
		j++;
	}
	x+=pasl;i++;
}
nl=i;np=j;
Scalare(wl,wr,wt,wb,vl,vr,vt,vb,ax,ay,bx,by);
int xe,ye,cod1,cod2;
CPoint e[100][100];
for(i=0;i<nl;i++)
	for(j=0;j<np;j++)
	{
		xe=(int)(ax*tx[i][j]+bx+0.5);
		ye=(int)(ay*ty[i][j]+by+0.5);
		e[i][j].x=xe;e[i][j].y=ye;
	}
int x1,y1,x2,y2;
for(i=0;i<nl;i++)
{
	x1 = e[i][0].x;y1=e[i][0].y;
	if(LCS[x1]<=y1)cod1=1;
	else if(LCI[x1]>=y1)cod1=3;
		else cod1=2;
	pDC->MoveTo(x1,y1);
	for(j=1;j<np;j++)
	{
		x2=e[i][j].x;y2=e[i][j].y;
		if(LCS[x2]<=y2)cod2=1;
		else if(LCI[x2]>=y2)cod2=3;
			else cod2=2;
		double a=y2-y1, b=x2-x1;
		double c=y1*x2-x1*y2;
		if(b!=0)
		{
			a=a/b;b=c/b;
			if(cod1==1&&cod2==1)
			{
				pDC->LineTo(x2,y2);
				for(int k=x1;k<=x2;k++)
				{
					LCS[k]=(int) (a*k+b+0.5);
```

```
                if(LCI[k]==VALMAX)LCI[k]=LCS[k];
        }
}
if(cod1==3&&cod2==3)
{
        pDC->LineTo(x2,y2);
        for(int k=x1;k<=x2;k++)
        {
                LCI[k]=(int) (a*k+b+0.5);
                if(LCS[k]==VALMIN)LCS[k]=LCI[k];
        }
}
if(cod1==1&&cod2==2)
{
        int kx=x1;
        int ky=y1;
        while(ky>=LCS[kx])
        {
                LCS[kx]=ky;
                if(LCI[kx]==VALMAX)LCI[kx]=LCS[kx];
                kx++;ky=(int) (a*kx+b+0.5);
        }
        pDC->LineTo(kx-1,LCS[kx-1]);
        pDC->MoveTo(x2,y2);
}
if(cod1==3&&cod2==2)
{
        int kx=x1;
        int ky=y1;
        while(ky<=LCI[kx])
        {
                LCI[kx]=ky;
                if(LCS[kx]==VALMIN)LCS[kx]=LCI[kx];
                kx++;ky=(int) (a*kx+b+0.5);
        }
        pDC->LineTo(kx-1,LCI[kx-1]);
        pDC->MoveTo(x2,y2);
}
if(cod1==2&&cod2==1)
{
        int kx=x2;
        int ky=y2;
        while(ky>=LCS[kx])
        {
                LCS[kx]=ky;
                if(LCI[kx]==VALMAX)LCI[kx]=LCS[kx];
                kx--;ky=(int) (a*kx+b+0.5);
        }
        pDC->MoveTo(kx+1,LCS[kx+1]);
        pDC->LineTo(x2,y2);
}
if(cod1==2&&cod2==3)
{
        int kx=x2;
        int ky=y2;
        while(ky<=LCI[kx])
        {
                LCI[kx]=ky;
                if(LCS[kx]==VALMIN)LCS[kx]=LCI[kx];
                kx--;ky=(int) (a*kx+b+0.5);
        }
        pDC->MoveTo(kx+1,LCI[kx+1]);
        pDC->LineTo(x2,y2);
}
if(cod1==1&&cod2==3)
{
        int kx=x1;
        int ky=y1;
        while(ky>=LCS[kx])
        {
                LCS[kx]=ky;
                if(LCI[kx]==VALMAX)LCI[kx]=LCS[kx];
                kx++;ky=(int) (a*kx+b+0.5);
        }
        pDC->LineTo(kx-1,LCS[kx-1]);
        kx=x2;ky=y2;
        while(ky<=LCI[kx])
```

```
                                        {
                                                LCI[kx]=ky;
                                                if(LCS[kx]==VALMIN)LCS[kx]=LCI[kx];
                                                kx--;ky=(int) (a*kx+b+0.5);
                                        }
                                        pDC->MoveTo(kx+1,LCI[kx+1]);
                                        pDC->LineTo(x2,y2);
                                }
                                if(cod1==3&&cod2==1)
                                {
                                        int kx=x1;
                                        int ky=y1;
                                        while(ky<=LCI[kx])
                                        {
                                                LCI[kx]=ky;
                                                if(LCS[kx]==VALMIN)LCS[kx]=LCI[kx];
                                                kx++;ky=(int) (a*kx+b+0.5);
                                        }
                                        pDC->LineTo(kx-1,LCS[kx-1]);
                                        kx=x2;ky=y2;
                                        while(ky>=LCS[kx])
                                        {
                                                LCS[kx]=ky;
                                                if(LCI[kx]==VALMAX)LCI[kx]=LCS[kx];
                                                kx--;ky=(int) (a*kx+b+0.5);
                                        }
                                        pDC->MoveTo(kx+1,LCS[kx+1]);
                                        pDC->LineTo(x2,y2);
                                }
                                if(cod1==2&&cod2==2)pDC->MoveTo(x2,y2);
                        }
                        cod1=cod2;
                        x1=x2;y1=y2;
                }
        }
}
void Scalare(double wl, double wr, double wt, double wb, int vl, int vr, int vt, int vb,
double &ax, double &ay, double &bx, double &by)
{
ax = (vr - vl) / (wr - wl);
bx = (vl*wr - wl*vr) / (wr - wl);
ay = (vb - vt) / (wb - wt);
by = (wb*vt - vb*wt) / (wb - wt);
}
```

Bibliografie

1. Bates, J., Topkins, T., *Utilizare Visual C++ 6*, Teora, Bucuresti, 2000

2. Cullens, C., Davidson, M., *Utilizare Visual C++* 4, Teora, Bucuresti, 1999

3. Cunningham, J., *Computer graphics and object oriented programming*, John Wiley & Sons, 1992

4. Dony, R., *Graphisme scientifique sur micro-ordinateur*, Masson, Paris, 1989