Problema liniilor si suprafetelor ascunse în grafica 3D. Metoda testului de vizibilitate

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

Problema eliminarii liniilor si suprafetelor ascunse s-a pus pentru prima data la mijlocul anilor '60. Dupa aceasta perioada s-au dezvoltat numerosi algoritmi, fiecare cu avantajele si dezavantajele sale. Orice algoritm este influentat de modul în care sunt descrise obiectele. Este importanta alegerea unei bune metode de reprezentare a unui obiect, care sa permita usor adaugarea sau eliminarea unor parti componente. Importanta este si maniera de memorare a datelor despre obiect, rapiditatea trasarilor fiind direct influentata de acest aspect.
***Cuvinte cheie**: grafica 3D, obiecte, test de vizibilitate.*

Majoritatea algoritmilor care s-au impus folosesc una din urmatoarele metode de reprezentare:

A. ***Reprezentarea unui obiect prin poligoane plane***. Ofera o buna manipulare a obiectelor simple compuse din poliedre convexe precum paralelipipede, prisme, piramide etc. Pentru fiecare obiect se memoreaza coordonatele vârfurilor si componenta fiecarei fete.

B. ***Reprezentarea prin suprafete geometrice***. Daca se doreste reprezentarea unei suprafete curbate, se observa ca reprezentarea prin poligoane nu convine, deoarece o buna calitate a reprezentarii implica un numar prea mare de poligoane. Suprafetele sunt reprezentate prin portiuni ce pot fi definite cu ajutorul unor ecuatii. În aceasta categorie intra:

1) *Suprafete descrise prin functii de doua variabile*. Suprafata este definita printr-un ansamblu de puncte ale caror coordonate sunt functii de doua variabile independente: x=f(u,v), y=g(u,v), z=h(u,v), unde x,y,z sunt coordonatele punctelor. Daca se alege pentru u si v valori discrete se obtine o multime de carouri marginite de linii drepte. Cu cât discretizarea valorilor u si v este mai mare cu atât reprezentarea este mai buna. Principalul dezavantaj al acestei metode consta în dificultatea identificarii functiilor f, g si h pornind de la obiectele de reprezentat.

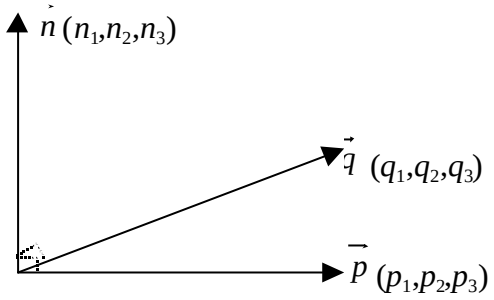
2) *Suprafete Bezier*. Metoda se bazeaza pe functii Bernstein care permit modificarea foarte usoara a curbelor, deci si a suprafetelor. Suprafetele astfel obtinute se numesc suprafete Bezier.

3) *Suprafete Splines*. Functiile Splines au fost propuse în 1973 de matematicianul Reisenfeld. Utilizarea curbelor Splines este o extensie a metodei precedente.

Metoda testului de vizibilitate

Aceasta metoda permite determinarea acelor suprafete ale unui obiect convex care sunt vizibile dintr-un anumit punct de observare. Daca privim un obiect de tip paralelipipedic, de exemplu, acesta va avea suprafete vizibile si suprafete ascunse. Notiunile de *produs scalar* si de *produs vectorial* sunt cele care ne vor ajuta sa determinam acele suprafete care sunt vizibile.

Consideram doi vectori $\vec{p}=(p_1,p_2,p_3)$ si $\vec{q}=(q_1,q_2,q_3)$ din spatiu. Acesti doi vectori sunt raportati la un sistem de axe direct. Prin definitie, produsul vectorial al doi vectori este un al treilea vector, perpendicular pe planul determinat de cei doi vectori. *Norma* vectorului rezultat este egala cu produsul normelor celor doi vectori dati si a sinusului unghiului format de vectorii respectivi.



$$\left\|\vec{n}\right\| = \left\|\vec{p}\right\| \cdot \left\|\vec{q}\right\| \cdot \sin q$$

Observatii:

1. Norma unui vector $\vec{p}$ se noteaza $\left\|\vec{p}\right\|$.

2. (p_1,p_2,p_3), (q_1,q_2,q_3) si (n_1,n_2,n_3) sunt valori scalare si corespund proiectiilor celor trei vectori pe axele sistemului de coordonate din care acestia fac parte.

Legatura dintre parametrii p_1,p_2,p_3,q_1,q_2,q_3 si n_1,n_2,n_3 este urmatoarea:

$$\begin{cases} n_1 = p_2q_3 - q_2p_3 \\ n_2 = p_3q_1 - q_3p_1 \\ n_3 = p_1q_2 - q_1p_2 \end{cases}.$$

Ceea ce intereseaza mai mult în definitia produsului vectorial este faptul ca vectorul rezultat este perpendicular pe plan. Norma produsului vectorial nu are nici o importanta în algoritmul de fata. Sensul vectorului $\vec{n}$ se obtine aplicând regula mâinii drepte (regula surubului), folosita deja în determinarea sensului sistemelor de coordonate. Un lucru important de care trebuie sa se tina cont este ca produsul vectorial nu este comutativ.

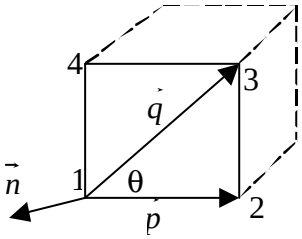


Fig.1.

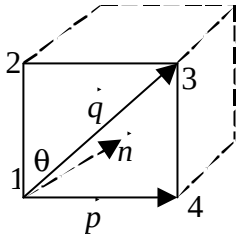


Fig.2.

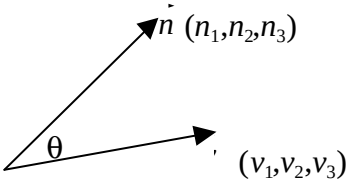
În figura 1, vârfurile primei suprafete a cubului sunt numerotate în sens trigonometric. Punctul de plecare în aceasta numerotare nu are nici o importanta. Daca desemnam prin latura (1,2) vectorul $\vec{p}$ si prin latura (1,3) vectorul $\vec{q}$, atunci vectorul rezultat $\vec{n} = \vec{p} \times \vec{q}$ va avea sensul catre exteriorul cubului, iar vectorul $\vec{q} \times \vec{p}$ va avea sens opus.

În figura 2, $\vec{n} = \vec{p} \times \vec{q}$ are sensul spre interior iar $\vec{q} \times \vec{p}$ spre exterior.

Testul de vizibilitate.

Definitie. Numim *produs scalar* al doi vectori $\vec{v}$ si $\vec{n}$, un numar real astfel determinat:

$$\vec{v} \cdot \vec{n} = v_1n_1 + v_2n_2 + v_3n_3$$



Cu ajutorul nurmelor, produsul scalar se scrie: $\vec{v} \cdot \vec{n} = \left\|\vec{v}\right\| \cdot \left\|\vec{n}\right\| \cdot \cos q$.

Deoarece o norma este intotdeauna pozitiva, sensul produsului scalar este determinat astfel:

$\vec{v} \cdot \vec{n} > 0$, atunci când $\cos q > 0$, deci $0 < q < 90°$, (figura 3)

$\vec{v} \cdot \vec{n} < 0$, atunci când $\cos q < 0$, deci $90° < q < 180°$ (figura 4).

Consideram figurile urmatoare:

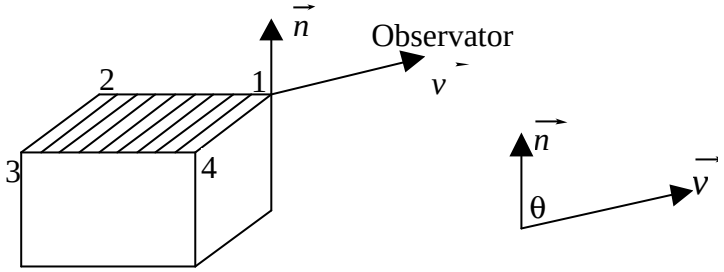


Fig.3. Suprafata vizibila, $0 < q < 90°$

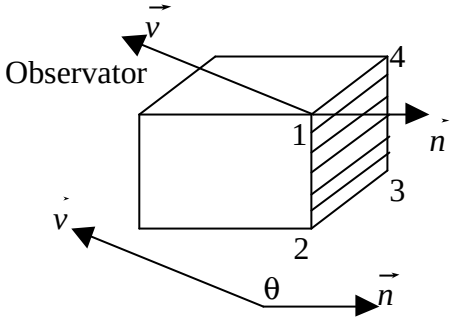


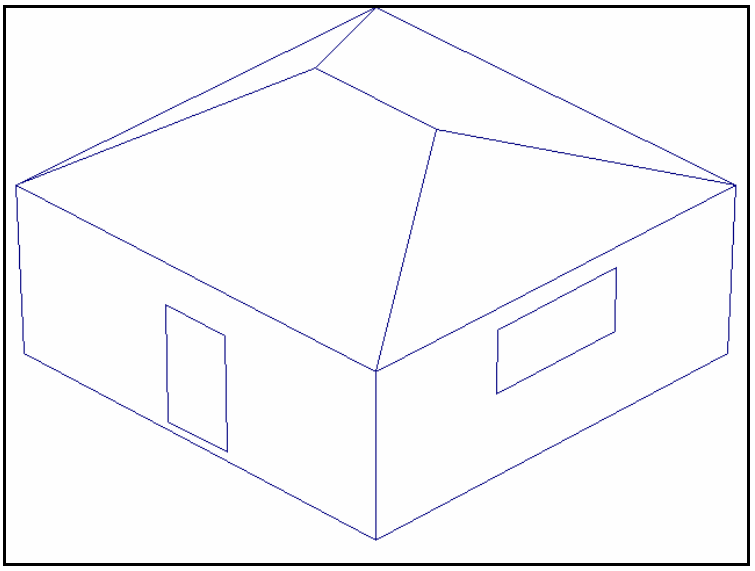
Fig.4. Suprafata invizibila, $90° < q < 180°$.

Pentru fiecare fata testata asociem vectorului normal la suprafata, $\vec{n}$, un al doilea vector, $\vec{v}$, care este *vectorul de vizualizare*. Acest vector porneste de la unul din vârfurile care definesc fata testata (nu are importanta care vârf) si puncteaza catre observator. Daca translatam cei doi vectori (sau unul dintre ei) într-o origine comuna si determinam marimea unghiului q dintre ei, se poate calcula sensul produsului scalar. Daca acesta este pozitiv suprafata este vizibila deoarece unghiul de vizualizare este mai mic decât 90°, iar daca este negativ, suprafata este invizibila, unghiul de vizualizare fiind mai mare decât 90°.

Concluzii. Pentru determinarea fetelor ascunse ale unui obiect poliedral convex prin metoda testului de vizibilitate se procedeaza astfel:

1. se determina vectorul normal la suprafata, $\vec{n}$,
2. se determina vectorul de vizualizare, $\vec{v}$,
3. se calculeaza produsul scalar $\vec{v} \cdot \vec{n}$,
4. daca produsul scalar este strict negativ, atunci fata testata este invizibila si nu se traseaza, daca este strict pozitiv atunci fata testata este vizibila si se traseaza, daca produsul scalar este 0 atunci cei doi vectori sunt ortogonali iar fata se va vedea din profil aparând ca o linie.

Exemplu. Sa presupunem ca dorim reprezentarea grafica a obiectului din figura urmatoare:



Datele sunt preluate dintr-un fisier text cu structura:

n - numar de puncte
m - numar de fete
x_1 y_1 z_1
...
x_n y_n z_n
n_1 - numar puncte fata 1
$p_1, p_2, ..., p_{n_1}$
...
n_m - numar puncte fata *m*
$p_1, p_2, ..., p_{n_m}$

Pentru exemplul de mai sus datele de intrare sunt:

```
18
11
4 1 0
4 4 0
1 4 0
1 1 0
4 4 2
1 4 2
1 1 2
4 1 2
2.5 2 3
2.5 3 3
4 2.25 0.1
4 2.75 0.1
4 2.75 1.5
4 2.25 1.5
3 4 1
2 4 1
2 4 1.75
3 4 1.75
4
1 4 3 2
4
1 2 5 8
4
2 3 6 5
4
3 4 7 6
4
1 8 7 4
4
8 5 10 9
3
5 6 10
4
6 7 9 10
3
7 8 9
4
11 12 13 14
4
15 16 17 18
```

Sunt prezentate în continuare functiile Visual C pentru trasare, scalare si schimbarea pozitiei observatorului.

```
void COmView::OnDraw(CDC* pDC)
{       COmDoc* pdoc = GetDocument();
        ASSERT_VALID(pdoc);
        // TODO: add draw code for native data here
        double tx[100],ty[100],Punct[100][3];
        double wl=1e20,wr=-1e20, wt=-1e20,wb=1e20,ax,ay,bx,by,x,y,z,xo,yo,zo,xr,yr,R;
        double ct=cos(pdoc->teta),st=sin(pdoc->teta), cf=cos(pdoc->fi),sf=sin(pdoc->fi);
        pdoc->pf=fopen("piramida.txt","rt");
        if(!pdoc->pf)return;
        RECT r;
        GetClientRect(&r);
        int vl=r.left+10,vt=r.top+10,vr=r.right-10,vb=r.bottom-10;
```

```
	int m,i,j,n,NpFata[100],Fata[100][10];
	fscanf(pdoc->pf,"%d",&n);fscanf(pdoc->pf,"%d",&m);
	for(i=0;i<n;i++)
	{		fscanf(pdoc->pf,"%lf",&Punct[i][0]);
		fscanf(pdoc->pf,"%lf",&Punct[i][1]);
		fscanf(pdoc->pf,"%lf",&Punct[i][2]);
	}
	for(i=0;i<m;i++)
	{	fscanf(pdoc->pf,"%d",&NpFata[i]);
		for(j=0;j<NpFata[i];j++)
			{fscanf(pdoc->pf,"%d",&Fata[i][j]);Fata[i][j]--;}
	}
	fclose(pdoc->pf);
	Dmax(n,Punct,i,j,R);
	x=(Punct[i][0]+Punct[j][0])/2;
	y=(Punct[i][1]+Punct[j][1])/2;
	z=(Punct[i][2]+Punct[j][2])/2;
	R=10*R;
	for(i=0;i<n;i++)
	{	Punct[i][0]-=x;
		Punct[i][1]-=y;
		Punct[i][2]-=z;
	}
	double o1=R*cf*ct,o2=R*cf*st,o3=R*sf;
	for(i=0;i<n;i++)
	{	x=Punct[i][0];y=Punct[i][1];z=Punct[i][2];
		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]=xr;ty[i]=yr;
		if(xr<wl)wl=xr;
		if(xr>wr)wr=xr;
		if(yr<wb)wb=yr;
		if(yr>wt)wt=yr;
	}
	pDC->SetROP2(R2_MASKPEN);
	Scalare(wl,wr,wt,wb,vl,vr,vt,vb,ax,ay,bx,by);
	for(j=0;j<m;j++)
	{	int a1=Fata[j][0],a2=Fata[j][1],a3=Fata[j][2];
		double p1=Punct[a2][0]-Punct[a1][0], p2=Punct[a2][1]-Punct[a1][1],
			p3=Punct[a2][2]-Punct[a1][2];
		double q1=Punct[a3][0]-Punct[a1][0], q2=Punct[a3][1]-Punct[a1][1],
			q3=Punct[a3][2]-Punct[a1][2];
		double vo1=o1-Punct[a1][0],vo2=o2-Punct[a1][1], vo3=o3-Punct[a1][2];
		double ProdScal=vo1*(p2*q3-q2*p3)+vo2*(p3*q1-q3*p1)+vo3*(p1*q2-q1*p2);
		if(ProdScal>0)
		{	CPoint F[10];
			for(i=0;i<NpFata[j];i++)
			{	F[i].x = (int) (ax*tx[Fata[j][i]]+bx+0.5);
				F[i].y = (int) (ay*ty[Fata[j][i]]+by+0.5);
			}
			pDC->Polygon(F,NpFata[j]);
		}
	}
}

void COmView::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);
}

void COmView::Dmax(int n, double a[][3], int &i, int &j,double &dist)
{
double dst;
dist=0;
for(int k=0;k<n;k++)
        for(int l=k+1;l<n;l++)
        {       dst=sqrt( (a[k][0]-a[l][0])*(a[k][0]-a[l][0]) + (a[k][1]-a[l][1])*(a[k][1]-a[l][1])
                        + (a[k][2]-a[l][2])*(a[k][2]-a[l][2]) );
                if(dst>dist){dist=dst;i=k;j=l;}
        }
}

void COmView::OnLButtonDblClk(UINT nFlags, CPoint point)
{       // TODO: Add your message handler code here and/or call default
        COmDoc* pdoc = GetDocument();
        pdoc->fi+=20*PI/180;
        Invalidate();
        CView::OnLButtonDblClk(nFlags, point);
}

void COmView::OnRButtonDblClk(UINT nFlags, CPoint point)
{       // TODO: Add your message handler code here and/or call default

        COmDoc* pdoc = GetDocument();
        pdoc->teta+=20*PI/180;
        Invalidate();
        CView::OnRButtonDblClk(nFlags, point);
}
```

Bibliografie

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

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

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

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