Računarska grafika

Uklanjanje skrivenih površi

Bafer dubine (Z-bafer)

- Pripada *image-space* algoritmima
  - radi nad slikom koja se prikazuje
- *Screen buffer*
  - memorija kod koje svaka lokacija opisuje boju piksela
- *Depth buffer (Z-buffer)*
  - memorija kod koje svaka lokacija služi za smeštanje dubine (z koordinate) 3D tačke čija se projekcija prikazuje na odgovarajućem pikselu
- Algoritam:
  - bafer dubine se inicijalizuje na vrednost z koordinate zadnje odsecajuće ravni
  - bafer ekrana se inicijalizuje na vrednost pozadine
  - za vreme iscrtavanja popunjenog poligona poziva se modifikovani `SetPixel`
    - proverava da li je tačka (x,y,z) bliža posmatraču od odgovarajuće koja je već prikazana
    - ako jeste, upisuje se boja u lokaciju bafera ekrana, a z koordinata tačke u bafer dubine
    - ako nije, ne menja se vredost ni u baferu ekrana ni u baferu dubine

Z-bafer algoritam

- Pretpostavka: Z-osa usmerena prema posmatraču (kameri)
- Inicijalizacija:

```
For i:=0 to Xmax do
  For j:=0 to Ymax do
    Begin
      depth[i,j]:=Zmin; screen[i,j]:= background
    End;
```

- Procedura za postavljanje piksela:

```
Procedure SetPixel(x:Xres;y:Yres;z:Zres;v:Value);
  Begin
    If z>depth[x,y] then
      Begin
        depth[x,y]:=z; screen[x,y]:=v
      End;
  End;
```

Z-bafer – računanje z koordinate

- Osnovni problem z-bafer algoritma:
  - izračunavanje z koordinate za svaku tačku 3D scene koja odgovara pikselu koji se prikazuje pri rasterizaciji (sken-konverziji) projekcije poligona
- Polazi se od jednačine ravni kroz 3 tačke:

$$\det\begin{bmatrix} x-x1 & y-y1 & z-z1 \\ x2-x1 & y2-y1 & z2-z1 \\ x3-x1 & y3-y1 & z3-z1 \end{bmatrix}=0 \quad\Rightarrow\quad Ax+By+Cz+D=0 \quad\Rightarrow\quad z=\frac{-D-Ax-By}{C}$$

- Pošto se crta tačka po tačka idući duž X-pravca (x se inkrementira za 1)
  - pod pretpostavkom ortogonalne projekcije
    - ako je u tački $(x,y) \Rightarrow z$, tada je u tački $(x+1,y) \Rightarrow z(x+1,y)=z(x,y)-A/C$
    - A/C je konstanta za ceo poligon i računa se samo jednom
  - ako se radi sa projekcijom sa perspektivom

$$\frac{x'}{x}=\frac{d}{d-z},\quad \frac{y'}{y}=\frac{d}{d-z} \quad\Rightarrow\quad \text{(iz jednačine ravni)} \quad z=\frac{Ax'+By'+D}{Ax'+By'-Cd}\cdot d$$

    - međusobni odnos z koordinata 2 tačke P i R određuje koja se vidi: $z_P>z_R \Rightarrow$ P se vidi

Z-bafer – prednosti i nedostaci

- Prednosti:
  - Jednostavnost, primenljiv na svaku scenu, mogućnost HW realizacije
- Nedostaci:
  - veličina z-bafera, izračunavanje za svaku tačku prilikom rasterizacije poligona
- Primer
  - ako ekran ima rezoluciju 2048x1024 piksela, odrediti:

    a) kapacitet memorije za Z-bafer (*depth buffer*) ako je potrebna rezolucija od 256 nivoa dubine;

    b) kapacitet video-memorije (*screen buffer*) za (istovremeni) prikaz 2^{24} različitih nijansi boje;
- Rešenje

  a) Za 256 nivoa dubine potrebno je 8 bita (1Byte) po pikselu za Z-bafer ($256 = 2^8$)
  Kapacitet memorije za Z-bafer je: $2^{11}x2^{10}B = 2^{21}B = 2MB$

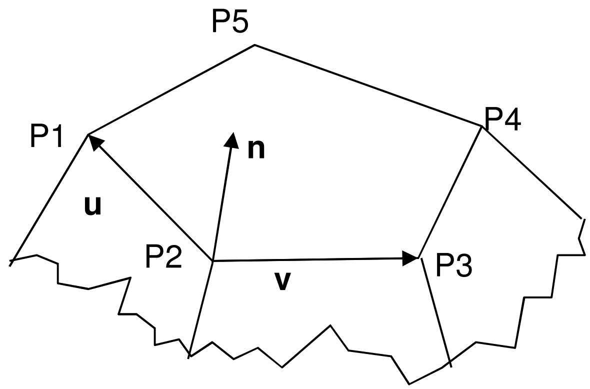
  b) Za prikaz 2^{24} različitih nijansi boje, potrebno je 24/3=8 bita po osnovnoj boji,
  Kapacitet video-memorije je jednak trostrukom kapacitetu Z-bafera (6MB)

BR algoritam

- BR ili BC algoritam (*Backface Removal*, *Backface Culling*)
  - uklanjanje površi posmatranih sa naličja
- Primenjivo na prikazivanje pojedinačnih neprozirnih objekata sa ravnim konveksnim površima
- Cilj algoritma je da utvrdi koje površi samo telo zaklanja od posmatrača
  - to su one površi koje su okrenute naličjem prema posmatraču
  - te površi se proglašavaju skrivenim, odnosno ne iscrtavaju se
- Najpre je potrebno odrediti pravac i smer u kojem je okrenuto lice površi
- Računa se normala na površ (*Face Normal*, *FN*)
  - *FN* - vektor normalan na površ usmeren od objekta prema spoljašnjosti
  - *FN* se dobija tako što se nađe vektorski proizvod dva vektora koji leže u ravni površi

BR algoritam – određivanje normale

- U ravni površi leže dve susedne ivice jedne stranice poliedra
- Da bi se dobio odgovarajući smer poštuje se sledeća konvencija:
  - temena poligona posmatrane površi objekta se označe redom u smeru nasuprot kretanja kazaljke časovnika posmatrano izvan tela



- Normala na površ se dobija kao:

$$\vec{v} = P2P3; \quad \vec{u} = P2P1; \quad \vec{n} = \vec{v} \times \vec{u}$$

BR algoritam – vektorski proizvod

- Predstavljeno matrično, najpre se izračuna matrica translacije koordinatnog početka u P2 (u bazu vektora ***v*** i ***u***):

$$T=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ -P2.x & -P2.y & -P2.z & 1\end{bmatrix}$$

- Zatim se odrede vektori ***v*** i ***u***:

$$\vec{v}=[x_v \quad y_v \quad z_v] \qquad [P3.x \quad P3.y \quad P3.z \quad 1]*T=[x_v \quad y_v \quad z_v \quad 1]$$

$$\vec{u}=[x_u \quad y_u \quad z_u] \qquad [P1.x \quad P1.y \quad P1.z \quad 1]*T=[x_u \quad y_u \quad z_u \quad 1]$$

- Normala na površ se računa kao vektorski proizvod ***v*** x ***u*** :

$$\vec{n}=\vec{v}\times\vec{u}=\det\begin{bmatrix}\vec{x} & \vec{y} & \vec{z}\\ x_v & y_v & z_v\\ x_u & y_u & z_u\end{bmatrix}=\left[\det\begin{bmatrix}y_v & z_v\\ y_u & z_u\end{bmatrix} \quad -\det\begin{bmatrix}x_v & z_v\\ x_u & z_u\end{bmatrix} \quad \det\begin{bmatrix}x_v & y_v\\ x_u & y_u\end{bmatrix}\right]=[x_n \quad y_n \quad z_n]$$

BR algoritam – vektor pogleda

- Sledeći korak je određivanje vektora pogleda (*View Vector, VV*)
- *VV* je vektor usmeren od površi prema posmatraču
- Uzima se iz posmatranog temena (P2)
- Ako se posmatrač nalazi u tački D(0,0,d) originalnog koordinatnog sistema, vektor pogleda ***w*** je:

$$\vec{w} = [x_w \quad y_w \quad z_w] \qquad [0 \quad 0 \quad d \quad 1] * T = [x_w \quad y_w \quad z_w \quad 1]$$

- Površ je okrenuta licem prema posmatraču
  - ako vektori ***n*** i ***w*** zaklapaju oštar ugao
  - ugao je oštar ako je kosinus ugla pozitivan
  - kosinus je pozitivan ako je skalarni proizvod vektora ***n*** i ***w*** pozitivan
- Površ je vidljiva ako:

$$\vec{n} \cdot \vec{w} = |\vec{n}| \cdot |\vec{w}| \cdot \cos(\angle(\vec{n}, \vec{w})) > 0$$

BR algoritam – skalarni proizvod

- Skalarni proizvod

$$\vec{n}\cdot\vec{w}=\begin{bmatrix}x_n & y_n & z_n\end{bmatrix}\cdot\begin{bmatrix}x_w & y_w & z_w\end{bmatrix}=x_n x_w+y_n y_w+z_n z_w=$$

$$x_w\cdot\det\begin{bmatrix}y_v & z_v\\ y_u & z_u\end{bmatrix}-y_w\cdot\det\begin{bmatrix}x_v & z_v\\ x_u & z_u\end{bmatrix}+z_w\cdot\det\begin{bmatrix}x_v & y_v\\ x_u & y_u\end{bmatrix}=$$

$$x_w(y_v z_u-y_u z_v)-y_w(x_v z_u-x_u z_v)+z_w(x_v y_u-x_u y_v)$$

$$\vec{u}=\begin{bmatrix}x_u & y_u & z_u\end{bmatrix}\quad \begin{bmatrix}P1.x & P1.y & P1.z & 1\end{bmatrix}*T=\begin{bmatrix}P1.x-P2.x & P1.y-P2.y & P1.z-P2.z & 1\end{bmatrix}$$

$$\vec{v}=\begin{bmatrix}x_v & y_v & z_v\end{bmatrix}\quad \begin{bmatrix}P3.x & P3.y & P3.z & 1\end{bmatrix}*T=\begin{bmatrix}P3.x-P2.x & P3.y-P2.y & P3.z-P2.z & 1\end{bmatrix}$$

$$\vec{w}=\begin{bmatrix}x_w & y_w & z_w\end{bmatrix}\quad \begin{bmatrix}0 & 0 & d & 1\end{bmatrix}*T=\begin{bmatrix}-P2.x & -P2.y & d-P2.z & 1\end{bmatrix}$$

BR algoritam – pseudokod (1)

- Pomoćna procedura za transformaciju proizvoljne tačke u 3D:

```
Procedure Transform(p: Point; Var q: Point; m: Matrix4x4);
  {p - original point,
   q - transformed point,
   m - matrix of transformation}
  Var
    v,w: Vector;
  Begin
    v[1]:= p.x;   v[2]:= p.y;   v[3]:= p.z;   v[4]:= 1.0;
    VxM(v,m,w);
    q.x:= w[1];   q.y:= w[2];   q.z:= w[3];
  End;
```

BR algoritam – pseudokod (2)

- Logička funkcija koja određuje da li je poligon `py` skriven:

```
Function HiddenPoly(py:Poly; p:Points; dist:Real):Boolean;
  Var
    t: Matrix4x4;
    p1,p3,d,dt: Point; {p1,p3,dt-tacke u transl.koord.sis.}
    mixprod: Real;     {[vect(p3) x vect(p1)] · vect(d)}
  Begin
    d.x:=  0.0;  d.y:= 0.0;  d.z:= dist;
    TranslInit(t,p[py[2]].x,p[py[2]].y,p[py[2]].z);
    Transform(p[py[1]],p1,t);
    Transform(p[py[3]],p3,t);
    Transform(d,dt,t);
    mixprod:= dt.x * (p3.y*p1.z - p1.y*p3.z) -
              dt.y * (p3.x*p1.z - p1.x*p3.z) +
              dt.z * (p3.x*p1.y - p1.x*p3.y);
    HiddenPoly:= (mixprod < 0);
  End;
```

BR algoritam – diskusija

- Algoritam je jednostavan za primenu
  ali nije univerzalan (zbog navedenih ograničenja)
- Praktično je veoma koristan
  - jer može da se primeni pre drugih algoritama veće složenosti
  - da se uštedi na ukupnom vremenu za uklanjanje skrivenih površi
- Poligoni, koje ovaj algoritam ukloni, sigurno se ne vide

Slikarev algoritam (sortiranje po dubini)

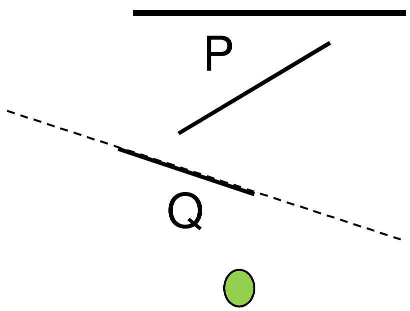
- Newell, Newell & Sancha 1972. godine
- Slikarev algoritam: prvo se "slika" najudaljeniji poligon koji ne sakriva druge, a zatim se preko njega slikaju poligoni koji ga (delimično) zaklanjaju
- Opseg (*extent*) koordinate
  - interval svih vrednosti odgovarajuće koordinate
- Koncept algoritma sortiranja po dubini (Z-osa usmerena ka posmatraču):
  - sortiraju se poligoni po rastućoj z-koord. najudaljenijeg temena poligona (z_{min})
    - dobijena lista: poligon sa najdaljim temenom – glava liste
  - kreće se od poligona sa najdaljim temenom (glava liste je “tekući” poligon)
    - za svaki poligon se proverava da li on zaklanja neki od narednih poligona u listi
      - sprovodi se 5 testova koji se ređaju od jednostavnijih ka složenijim proverama
      - ako su z-opsezi poligona različiti, nema potrebe analizirati naredne poligone
    - ako zaklanja – zaklonjeni poligon se prevezuje ispred zaklanjajućeg (tekućeg)
      - prebačeni poligon postaje tekući
  - rešavaju se uzajamna preklapanja
    - detektuje se potencijalna beskonačna petlja, pa se dele (seku) poligoni
  - crtaju se popunjeni poligoni počevši od najudaljenijih (od glave liste)

Slikarev algoritam – algoritam

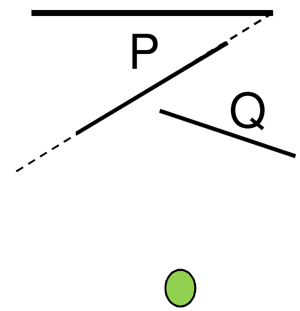
```
{Sortiranje poligona po najudaljenijem temenu,
P ukazuje na glavu liste - poligon sa najdaljim temenom}
Repeat
  Q:=P^.next;
  While (Q<>nil) and (z-opsezi P^ i Q^ se preklapaju) do
     Begin
       If (x-opsezi P^ i Q^ se ne preklapaju) or       {1}
           (y-opsezi P^ i Q^ se ne preklapaju) or       {2}
           (P^ je u potpunosti iza ravni Q^) or         {3}
           (Q^ je u potpunosti ispred ravni P^) or      {4}
           (projekc. P^ se ne preklapa sa projekc. Q^){5}
       Then { P^ ne sakriva ni deo Q^}
       Else Begin
             prevezivanje Q^ ispred P^ u listi; P:=Q
       End;
       Q:=Q^.next;
     End;
  crtanje popunjenog P^; P:=P^.next;
Until P=nil;
```

Slikarev algoritam – testovi

- Za slučaj poligona normalinih na horizontalnu ravan: pogled „odozgo“



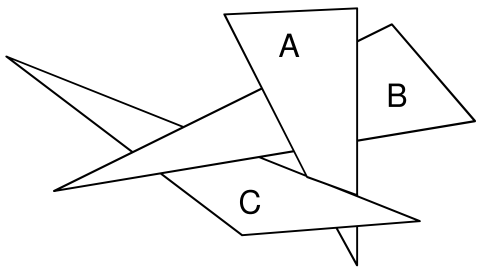
Poligon P iza ravni poligona Q



Poligon Q ispred ravni poligona P

Slikarev algoritam – problem i rešenje

- Problem:
  - u slučaju uzajamnog preklapanja (npr. A,B i C na slici) dolazi do beskonačne petlje



- Slučaj uzajamnog preklapanja se otkriva tako što se:
  - pri prebacivanju Q ispred P u listi, Q se obeleži
  - pri svakom prebacivanju vrši se provera da li je poligon Q već bio obeležen
  - ako se otkrije da je Q već bio obeležen - detektuje se uzajamno preklapanje
- U slučaju detektovanog preklapanja
  - poligon Q se deli na 2 (Q1 i Q2) pomoću ravni poligona P