RAČUNARSKA GRAFIKA

Oznaka predmeta: RG

Predavanje broj: 09

Nastavna jedinica: 3D transformacije i projekcije. Skrivene linije, morfing.

Nastavne teme: 3D transformacije: translacija, rotacija, skaliranje. Složene transformacije. Rotacija oko proizvoljne ose. Osnovne klase projekcija: perspektiva, paralelna. Uklanjanje skrivenih linija i površina. Geometrijsko uređivanje. Min-maks provera. Provera odnosa tačke i poligona. Proširenje algoritma Min-maks na 3D. Warnock-ov postupak. Podela prostora (quad tree). Watkins-ov postupak (scan line method). BSP. Uklanjanje poligona (culling). Ćelije i portali. PVS potencijalno vidljivi skup (Potentially visible set). LOD, morfing.

Predavač: prof. dr Perica S. Štrbac, dipl. ing.

Literatura:

James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes: "Computer Graphics: Principles and Practice", 2nd ed. in C, Addison-Wesley, 1996.

Koordinatni sistemi

- Svaki grafički paket ima bar jedan koordinatni sistem
  - Dekartov
  - Cilindrični
  - Polarni
  - Sferni
- World coordinates (WCS – World Coordinate System): prirodne koordinate, prevode se u normalizirane koordinate.

  $(-\infty,+\infty) \rightarrow [-1,1]$
- Javlja se problem prikaza na ekranu koji ima prebrojivo tačaka
- Vrši se prevođenje normaliziranih koordinata u ekranske koordinate
- Moguće je i izvršiti prevođenje WCS u ekranske koordinate
- Da bi se složena transformacija izvela kao proizvod osnovnih transformacija koriste se afine koordinate

Transformacije u 3D

- Pretpostavlja se konvencija pokretne virtuelne kamere
- Postoji formalna sličnost sa transformacijama u 2D grafici:
  - dodaje se jedan član jednačina (za koordinatu z),
  - dodaje se jedna jednačina (za z')
  - posledica je da matrica transformacije postaje 4x4

$$x' = A1 \cdot x + B1 \cdot y + C1 \cdot z + D1 \cdot 1$$

$$y' = A2 \cdot x + B2 \cdot y + C2 \cdot z + D2 \cdot 1$$

$$z' = A3 \cdot x + B3 \cdot y + C3 \cdot z + D3 \cdot 1$$

$$1 = 0 \cdot x + 0 \cdot y + 0 \cdot z + 1 \cdot 1$$

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \cdot \begin{bmatrix} A1 & A2 & A3 & 0 \\ B1 & B2 & B3 & 0 \\ C1 & C2 & C3 & 0 \\ D1 & D2 & D3 & 1 \end{bmatrix}$$

Translacija

- Koordinatni početak O(0,0,0) se translatorno pomera u tačku O'(x1,y1,z1):

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x1 & -y1 & -z1 & 1 \end{bmatrix}$$

- Translacija za dx,dy,dz bez pomeranja koordinatnog početka

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{bmatrix}$$

Rotacije

- U 2D grafici
  - jedna elementarna rotacija (oko koordinatnog početka)
- U3D grafici
  - 3 elementarne rotacije (oko svake ose koordinatnog sistema)
- Matrice rotacije koordinatnog početka oko X ose (Rx) za ugao α, oko Y ose (Ry) za ugao $-\beta$ i oko Z ose (Rz) za ugao γ:

$$Rx = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Ry = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Rz = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matrice rotacije tačke oko X ose (Rx) za ugao α, oko Y ose (Ry) za ugao $-\beta$ i oko Z ose (Rz) za ugao γ:

$$Rx = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Ry = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Rz = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 & 0 \\ -\sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Skaliranje

- Faktori skaliranja za ose X,Y,Z su Sx, Sy i Sz, respektivno
- Matrica skaliranja po sve tri ose:

$$S = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matrica skaliranja oko referentne tačke (Xr,Yr,Zr) po sve tri ose:

$$S = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ (1-Sx)Xr & (1-Sy)Yr & (1-Sz)Zr & 1 \end{bmatrix}$$

Složene transformacije

- Kao i kod transformacija u 2D grafici
  - složena transformacija se može dekomponovati na elementarne transformacije
  - određuje se kompozitna matrica složene transformacije množenjem matrica elementarnih transformacija
- Redosled elementarnih transformacija u složenoj transformaciji je bitan, množenje matrica nije komutativno

  Šta je sledeća matrica 3D transformacije:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -Xr & -Yr & -Zr & 1 \end{bmatrix} \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Xr & Yr & Zr & 1 \end{bmatrix}$$

Rotacija oko proizvoljne ose

- Problem:
  - izvršiti rotaciju za ugao α oko ose koja spaja tačke P1(x1,y1,z1) i P2(x2,y2,z2).

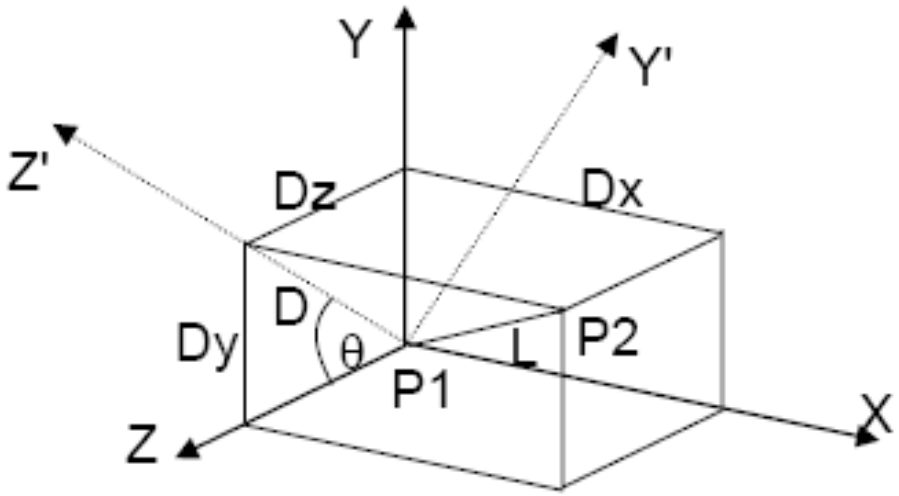
Rešenje:

- Translacija koordinatnog sistema u tačku P1(x1,y1,z1).
- Rotacija sistema za ugao $-\theta$ (u smeru kazaljke na časovniku) oko ose X tako da se data linija P1P2 dovede da leži u XoZ ravni (sada XoZ')

Dx=x2-x1

Dy=y2-y1

Dz=z2-z1



$$D = \sqrt{Dy^2 + Dz^2}$$

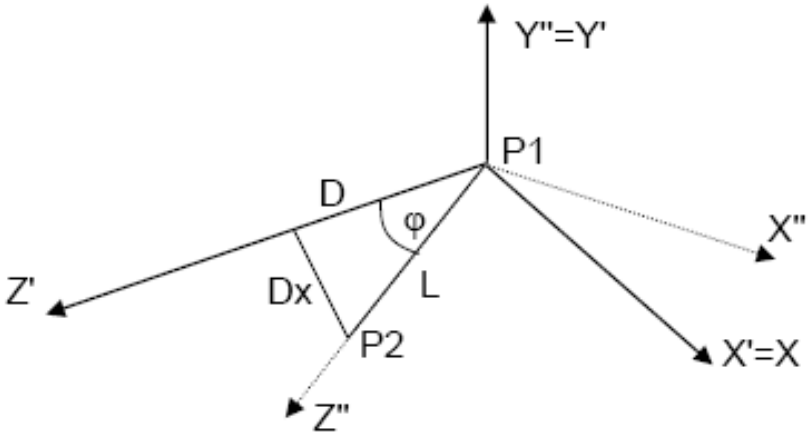
$$\sin\theta = Dy/D \qquad \cos\theta = Dz/D$$

Rotacija oko proizvoljne ose

- Rotacija sistema oko Y‘ za ugao ϕ tako da se data linija P1P2 poklopi sa Z’-osom (sada Z’’)

$$L = \sqrt{D^2 + Dx^2} = \sqrt{Dy^2 + Dz^2 + Dx^2}$$

$$\sin\varphi = Dx/L \qquad \cos\varphi = D/L$$

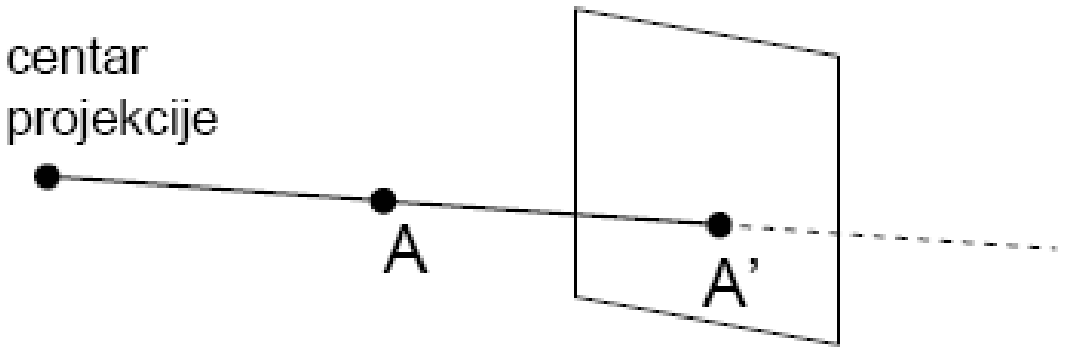


- Rotacija sistema oko Z'' za ugao α.
- Rotacija sistema oko Y za ugao $-\phi$.
- Rotacija sistema oko X za ugao θ.
- Translacija u originalni koordinatni početak.

- Kompozitna matrica totalne transformacije: R=T*Rx*Ry*Rz*Ry'*Rx'*T‘ (ovde je M' označeno kao inverzne matrica transformacije od M)

Projekcija

- Generalno, projekcija je transformacija koja preslikava tačku
  - iz koordinatnog sistema sa N dimenzija
  - u koordinatni sistem sa manje od N dimenzija
- Ovde
  - ograničenje na preslikavanje 3D objekta na 2D površinu
  - ograničenje na planarne geometrijske projekcije
- Planarna geometrijska projekcija 3D tačke se dobija tako što se:
  - *projekcioni zrak*(*projektor*) emituje iz *centra projekcije* kroz željenu 3D tačku
  - odredi se presek projektora sa *projekcionom ravni*



Osnovne klase projekcija

- 3D objekti se moraju prikazati na 2D uređajima – potreba za projekcijama
- Povoljna okolnost je što je projekcija linije takođe linija, te se projektovanje svodi na krajnje tačke
- Dve osnovne klase planarnih geometrijskih projekcija su
  - *perspektiva*
  - *paralelna*
- Kod perspektive
  - centar projekcije i projekciona ravan su na konačnom rastojanju
  - zadržavaju se relativne proporcije objekta
- Kod paralelne projekcije
  - centar projekcije je beskonačno udaljen od projekcione ravni
  - sve pozicije koordinata prenose se na ravan paralelno
    - projektovanje pod uglom ima zrake projekcije koje su međusobno paralelne, ali nisu okomite na ravan pogleda

Perspektiva

- Prirodan vizuelni efekat sličan fotografiji i čovekovom vizuelnom sistemu
- Efekat se naziva perspektivnim skraćenjem (*perspective foreshortening*):
  - veličina projekcije objekta se menja inverzno sa rastojanjem objekta od centra projekcije
- Mane su što se na projekciji ne mogu meriti dužine i uglovi
  - uglovi su realni samo za stranice objekta koje su paralelne sa projekcionom ravni
- Karakteristika perspektivne projekcije svakog skupa paralelnih linija (koje nisu paralelne projekcionoj ravni)
  - skup konvergira u zajedničku "iščezavajuću" (*vanishing*) tačku
    - koristi se izraz i nedogled

Osna iščezavajuća tačka

- *Osna iščezavajuća tačka*(AVP – Axis Vanishing Point)
  - tačka iščezavanja linija paralelnih osi koordinatnog sistema
- Broj osnih iščezavajućih tačaka je jednak broju osa koje preseca projekciona ravan
  - čest slučaj: centar projekcije na Z-osi a projekciona ravan XoY
    - postoji samo jedna AVP
- U arhitekturi i inženjeringu
  - projekcija sa 2 AVP se koristi često
  - projekcija sa 3 AVP se koristi ređe

Paralelna projekcija

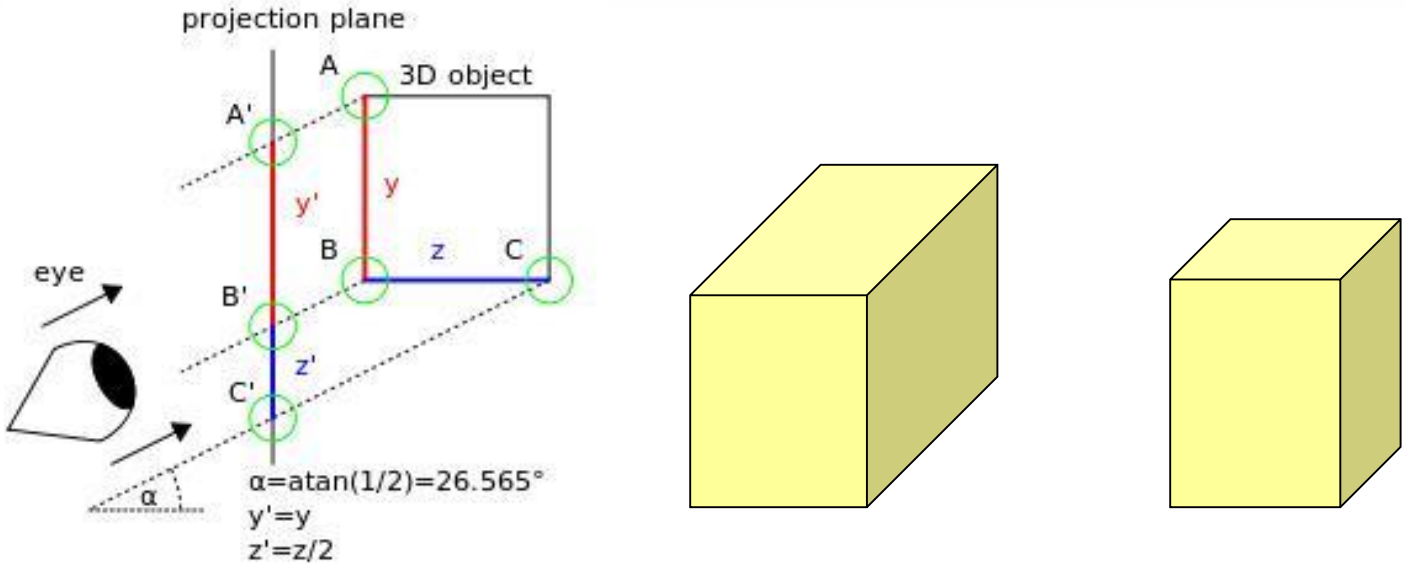
- Projekcioni zraci su paralelni.
- Određen je samo pravac i smer projekcionih zraka
  - vektor: *smer projekcije*(*direction of projection*)
- Generalno, vektor je razlika tačaka u homogenom sistemu:

  $v=(x,y,z,1) -(x',y',z',1) = (a, b, c, 0)$
- Paralelna projekcija je perspektiva čiji je centar u beskonačnosti.
- Paralelna projekcija ima sledeće osobine:
  - na njoj se mogu meriti rastojanja
    - iako i ovde mogu biti različita (ali konstantna) skraćenja po svakoj od osa (različiti metodi prikaza)
  - paralelne linije ostaju paralelne (ne postoje iščezavajuće tačke)
  - uglovi su očuvani samo na stranicama tela koje su paralelne projekcionoj ravni

Klasifikacija paralelnih projekcija

- Paralelne projekcije se dele na:
  - ortogonalne (ortografske)
    - smer projekcije normalan na projekcionu ravan koja je koordinatna ravan
    - pogled odozgo (*top, plan view*), spreda (*front view*), sa strane (*side view*)
  - aksonometrijske
    - koriste projekcionu ravan koja nije normalna na ose koordinatnog sistema
    - izometrijske
      - normala na projekcionu ravan zaklapa jednake uglove sa sve 3 ose
  - iskošene
    - *Oblique*, smer projekcije nije normalan na projekcionu ravan
    - *Kavaljeova*
      - smer projekcije zaklapa ugao od 45° sa projekcionom ravni, faktor skraćenja je nula
    - *Kabinet*
      - smer projekcije zaklapa ugao od arctg(2)=63.4°
      - linije normalne na projekcionu ravan se skraćuju faktorom 2

Paralelne projekcije pod uglom



- Kosa paralelna projekcija, *Oblique projection*
  - putanja projekcije nije okomita na ravan pogleda
- Cavalierova projekcija
  - tg(β) = 1, β = 45°
  - sve linije koje su normalne na ravan pogleda projektuju se bez promene dužine
- Kabinetska projekcija
  - tg(β) = 2, β ≈ 63,4°
  - sve linije koje su normalne na ravan pogleda projektuju se na polovinu svoje dužine

Matematičko modeliranje projekcija

- Ovde se uzima da je projekciona ravan XoY ravan
- Centar projekcije se nalazi na pozitivnoj Z osi i to:
  - ako je centar u tački P(0,0,+∞) projekcija je ortogonalna (paralelni zraci, normalni na ravan)
  - ako je centar u tački P(0,0,d≠+∞) projekcija je sa perspektivom
- Ideja je da se i projekcija tretira kao elementarna transformacija, kako bi matematički aparat bio jednoobrazan

Ortogonalna projekcija na XoY

Svodi se na ignorisanje z komponente tačke (projekciona ravan je z'=0):

$$x' = x$$
$$y' = y$$
$$z' = 0$$
$$x' = 1\cdot x + 0\cdot y + 0\cdot z + 0\cdot 1$$
$$y' = 0\cdot x + 1\cdot y + 0\cdot z + 0\cdot 1$$
$$z' = 0\cdot x + 0\cdot y + 0\cdot z + 0\cdot 1$$
$$1 = 0\cdot x + 0\cdot y + 0\cdot z + 1\cdot 1$$

$$Po = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Odnosno pošto slika tačke treba da bude izražena u 2D:

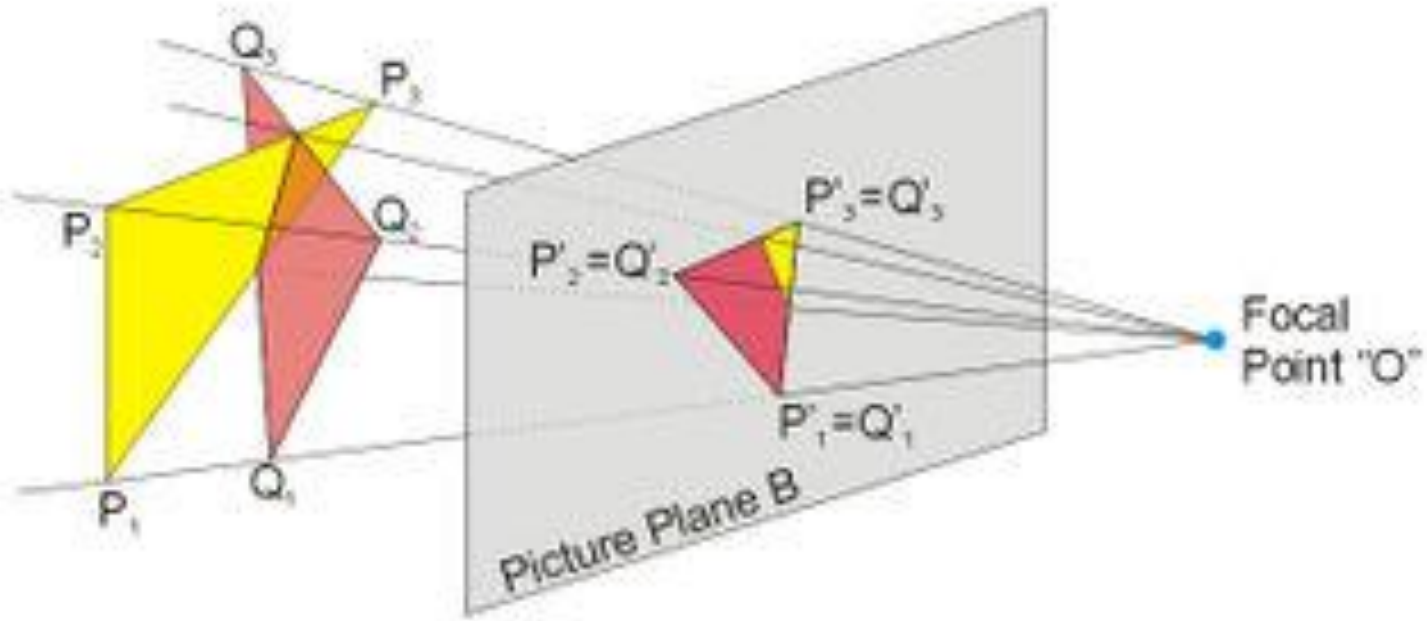
$$Po = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Konačno transformacija je:

$$Q_{2D} = P_{3D} * Po = [x' \quad y' \quad 1] = [x \quad y \quad z \quad 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Perspektiva

- Zrake projekcije nisu paralelne nego se seku u zamišljenoj tački (projekciona referentna tačka, centar projekcije)
- Dalji objekti su manji, bliži su veći

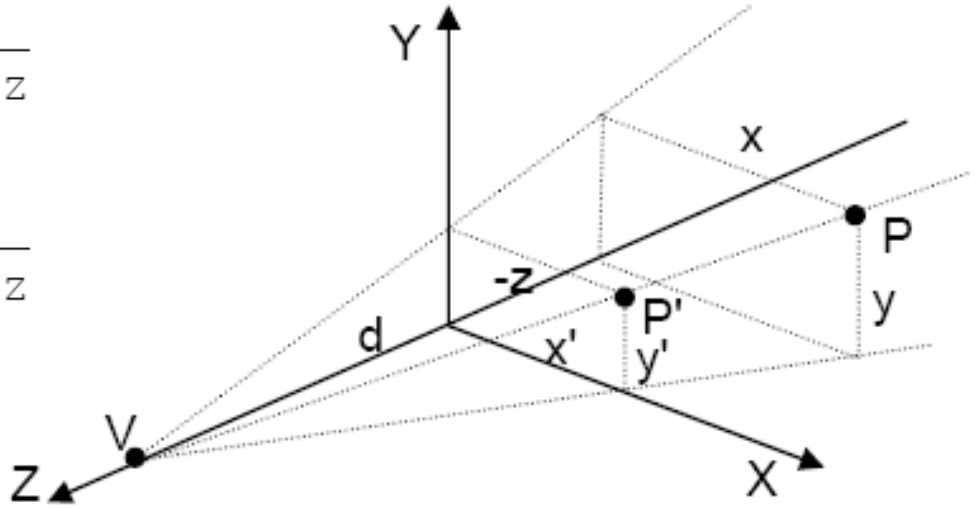


Projekcija sa perspektivom

- Centar projekcije V(0,0,d) leži na pozitivnoj Z osi

$$\frac{x'}{d} = \frac{x}{d+(-z)} \quad \Rightarrow \quad x' = x \cdot \frac{d}{d-z}$$

$$\frac{y'}{d} = \frac{y}{d+(-z)} \quad \Rightarrow \quad y' = y \cdot \frac{d}{d-z}$$



$$\left.\begin{aligned} x' &= \frac{d}{d-z}\cdot x + 0\cdot y + 0\cdot 1 \\ y' &= 0\cdot x + \frac{d}{d-z}\cdot y + 0\cdot 1 \\ 1 &= 0\cdot x + 0\cdot y + 1\cdot 1 \end{aligned}\right\} \cdot \frac{d-z}{d} = w$$

$$x'' = x' \cdot \frac{d-z}{d} = 1\cdot x + 0\cdot y + 0\cdot z + 0\cdot 1$$

$$y'' = y' \cdot \frac{d-z}{d} = 0\cdot x + 1\cdot y + 0\cdot z + 0\cdot 1$$

$$w = 1\cdot \frac{d-z}{d} = 0\cdot x + 0\cdot y + \left(-\frac{1}{d}\right)\cdot z + 1\cdot 1$$

Projekcija sa perspektivom

- U matričnom obliku:

$$[x'' \quad y'' \quad w] = [x \quad y \quad z \quad 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{d} \\ 0 & 0 & 1 \end{bmatrix}$$

- Matrica projekcije sa perspektivom:

$$Pp = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{d} \\ 0 & 0 & 1 \end{bmatrix}$$

- Koordinate tačke projekcije se dobijaju:

$$[x' \quad y' \quad 1] = \left[\frac{x''}{w} \quad \frac{y''}{w} \quad 1\right]$$

Napomena: za d=0 (centar projekcije u koordinatnom početku)

  - projekcija je nedefinisana

Primer

- Tačka P(3,5,7) se projektuje u tačku Q
- Centar projekcije je u tački V(0,0,10), a projekciona ravan je XoY

$$\begin{bmatrix}3 & 5 & 7 & 1\end{bmatrix}\cdot\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & -\frac{1}{10}\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}3 & 5 & 0.3\end{bmatrix}\Rightarrow$$

$$\begin{bmatrix}3 & 5 & 0.3\end{bmatrix}\ /0.3\ =\begin{bmatrix}10 & \frac{50}{3} & 1\end{bmatrix}\Rightarrow$$

- Rešenje je $Q(10,\frac{50}{3})$

Koje biste matrice transformacije koristili?

- Četvorostranu piramidu

  A(50,50,30)

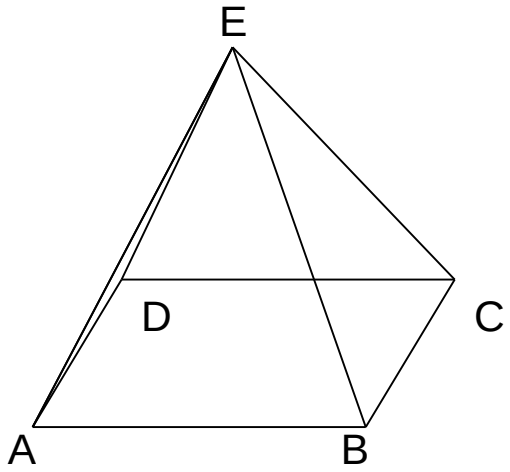
  B(150,50,30)

  C(150,150,30)

  D(50,150,30)

  E(100,100,100)

  projektovati na X0Y osu.



- Rotirati piramidu ABCDE oko x ose za 45 stepeni, skalirati dobijenu piramidu A'B'C'D'E' za Sx=Sy=Sz=0.5 i takvu projektovati na X0Y osu.
- Piramidu ABCDE projektovati na X0Y perspektivom za V(0,0,200).

Primer 1/3

```
#pragma comment( lib, "opengl32.lib")
#pragma comment( lib, "glu32.lib")
#pragma comment( lib, "glut32.lib")
#include <GL/glut.h>

GLdouble yeye = 0; GLfloat ugao = 45;
void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    gluLookAt(0, yeye, 8, 0, 0, 0, 0, 1, 0);

    glTranslatef(0.0, 0.0, 3.0);// Centralni cajnik
    glutSolidTeapot(1.0);   //glutSolidSphere(1.0,16,16);

    glTranslatef(2.0, 2.0, -1.0);
    glutSolidTeapot(1.0);// Gornji desni cajnik

    glTranslatef(-4.0, 0.0, -1.0);// Gornji levi cajnik
    glutSolidTeapot(1.0);

    glTranslatef(0.0, -4.0, -1.0);// Donji levi cajnik
    glutSolidTeapot(1.0);

    glTranslatef(4.0, 0.0, -1.0);// Donji desni cajnik
    glutSolidTeapot(1.0);
    glFlush();
}
```

Primer 2/3

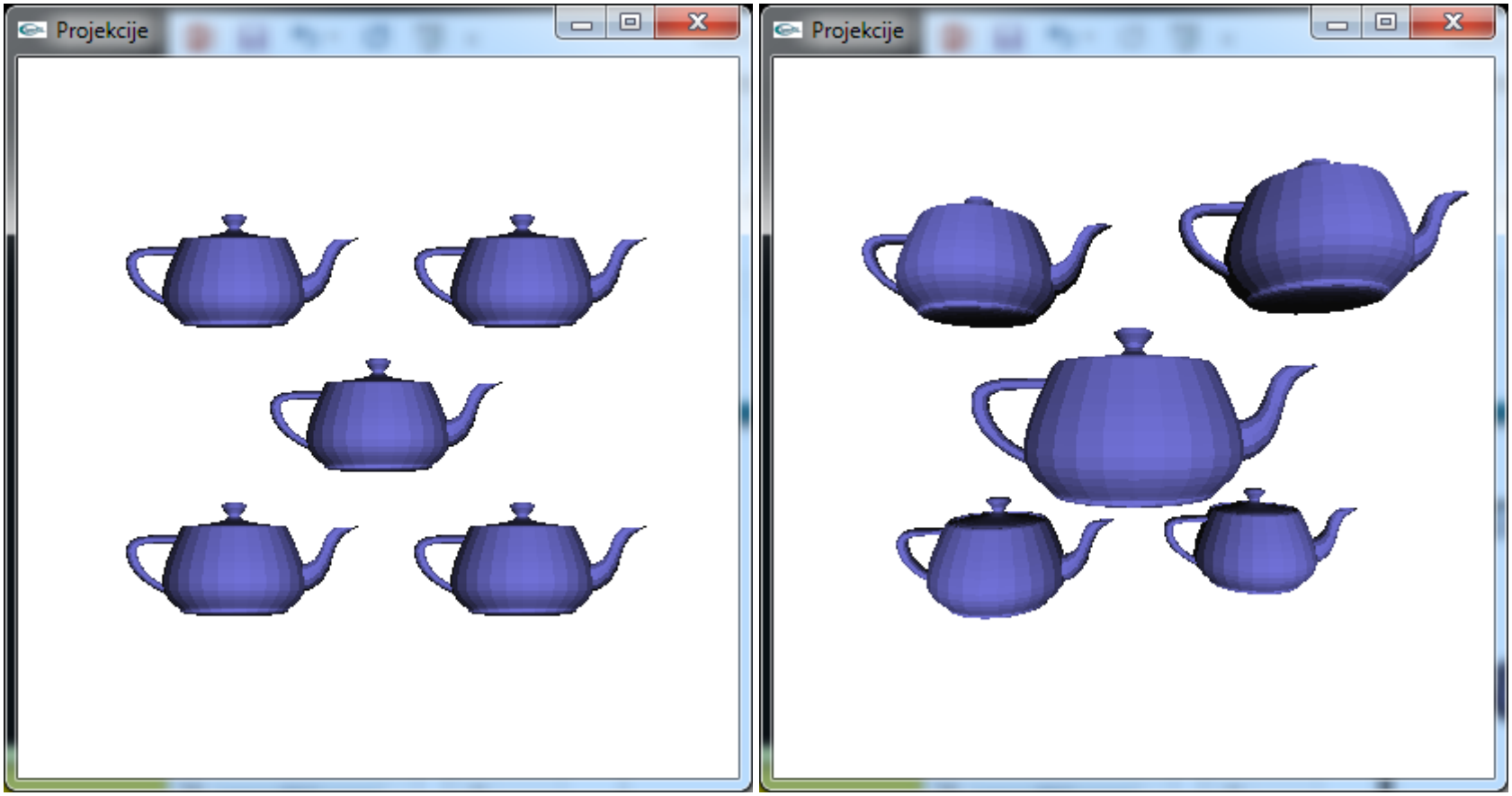
```
void kbd(unsigned char key, int x, int y) {
    switch (key) {
        case 27 :  exit(0);   break;
        case 'o' :
        case 'O' :
            glMatrixMode(GL_PROJECTION);
            glLoadIdentity();
            glOrtho(-5, 5, -5, 5, -10, 10);
            break;
        case 'f' :
        case 'F' :
            glMatrixMode(GL_PROJECTION);
            glLoadIdentity();
            glFrustum(-2, 2, -2, 2, 3.0, 10);
            break;
        case 'w' :
        case 'W' : ugao+=15.0; yeye+=0.2; break;
        case 's' :
        case 'S' : ugao-=15.0; yeye-=0.2; break;
    }
    glMatrixMode(GL_MODELVIEW);
    glutPostRedisplay();
}
// Callback funkcija za promenu velicine prozora
void reshape(GLint w, GLint h) {
    glViewport(0, 0, w, h);
}
```

Primer 3/3

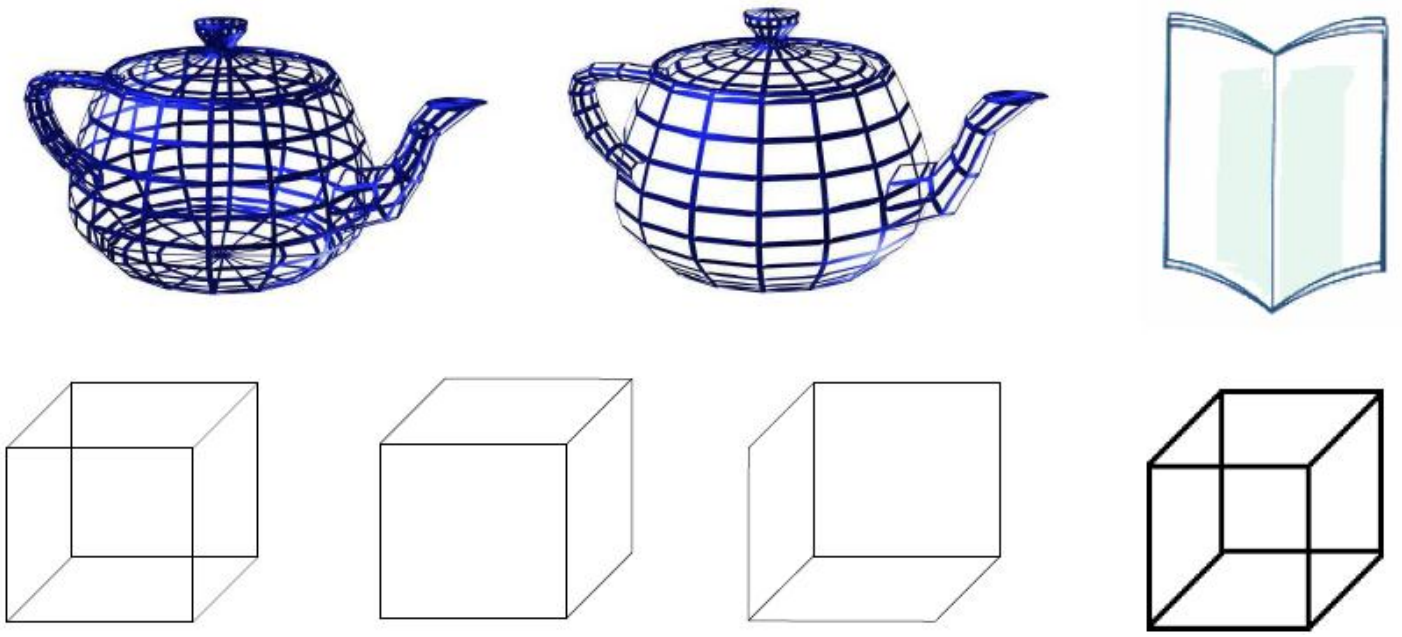
```
void init() {
    GLfloat light_position[] = {0.0, 0.0, 6.0, 0.0};
    GLfloat light_color[]    = {0.5, 0.5, 1.0, 0.0};
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glShadeModel(GL_FLAT);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glLightfv(GL_LIGHT0, GL_DIFFUSE,  light_color);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
    glColor3f(1.0, 1.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-5, 5, -5, 5, -10, 10);
    glMatrixMode(GL_MODELVIEW);
}
void main(int argc, char** argv) {
    glutInit(&argc,argv);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow("Projekcije");
    init();
    glutDisplayFunc(display);
    glutKeyboardFunc(kbd);
    glutReshapeFunc(reshape);
    glutMainLoop();
}
```

Primer 3/3

Izlaz



Uklanjanje skrivenih linija i površina



- Određivanje vidljivosti (hidden-line/surface algorithms)
  - Za čoveka je uklanjanje skrivenih linija i površina jednostavan problem.
  - Za implementaciju skrivanja linija i površina pri prikazu na ekranu koristi se trigonometrija

Uklanjanje skrivenih linija i površina

- Osnove postupaka uklanjanja skrivenih linija i površina
  - geometrijska izračunavanja – uspostavljaju odnos između poligona, bridova i tačaka ("containment test")
  - geometrijsko uređivanje ("geometric sorting")
  - postupci pretraživanja ("search algorithms")
  - međusobna zavisnost i obeležja ("coherence")
- Određivanje vidljivosti
  - uklanjanje objekata izvan piramide pogleda (delova poligona)
  - uklanjanje zadnjih pogona (back face culling)
  - brzi, jednostavni postupci za rešavanje trivijalnih slučajeva (npr. Min-maks provera)
  - promena složenosti prikaza u zavisnosti od udaljenosti (LOD → level of detail)
- Podela postupaka za uklanjanje skrivenih linija i površina
  - postupci u prostoru objekta 3D
  - postupci u prostoru slike (projekcije) 2D

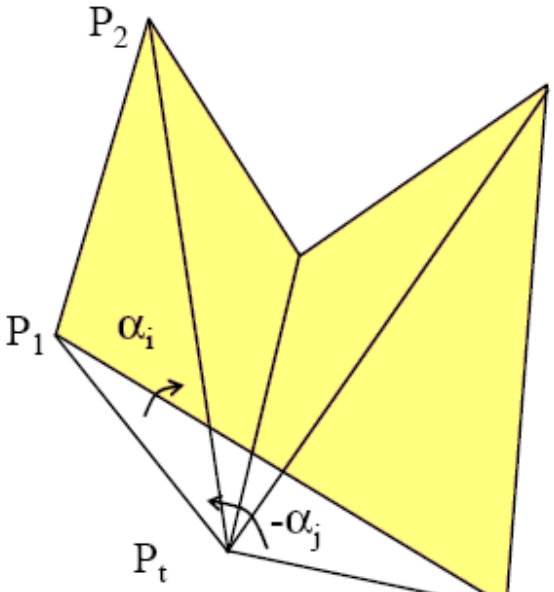
Uklanjanje skrivenih linija i površina

- Slični postupci koriste se kod
  - odsecanja (clipping)
  - detekcije sudara tj. kolizije (collision detection)
  - "bačene" sene (senčenje)
- Geometrijska izračunavanja čine osnovu u postupcima uklanjanja skrivenih linija i poligona, detekcija kolizija, odsecanja. (U prostoru projekcije ili scene).
  - položaj tačke prema
    - pravcu ili ravni
    - poligonu ili telu
  - položaj dužine prema
    - pravcu ili ravni
    - poligonu ili telu
  - Booleove operacije (unija, presek, razlika)
    - dva tela (solid modelling)

Provera odnosa tačke i poligona

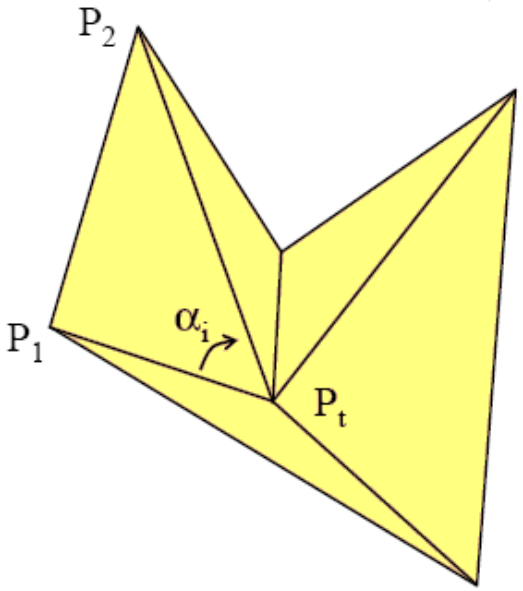
- Korišćenje sume uglova.
  - ako je

P_t je izv $\sum_i \alpha_i = 0^\circ$



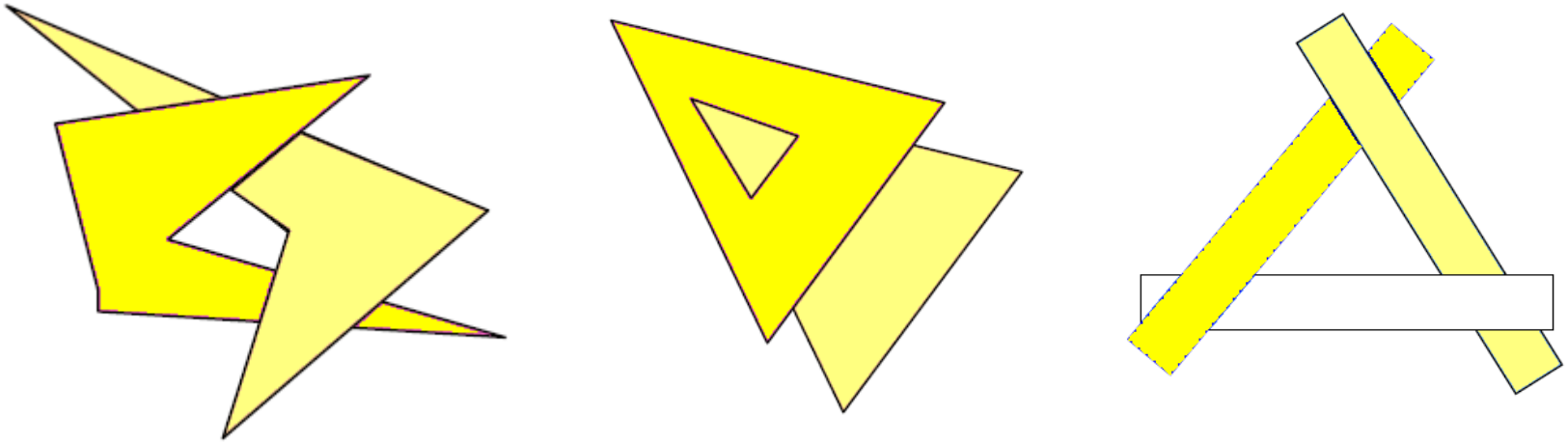
  - ako je

P_t je unu $\sum_i \alpha_i = 2\pi$



Geometrijsko uređivanje

- Na primer algoritam iscrtavanja slikara. BTF (back to front) (Painter algorithm)
  - Nakon iscrtavanja najudaljenijih poligona crtaju se redom sve bliži poligoni koji prekrivaju već iscrtane.
    - Prednosti: - mogućnost korišćenja prozirnosti
    - Nedostaci: - suvišna iscrtavanja prekrivenih poligona
  - problem kod iscrtavanja površina ili probadanja



Min-maks provera

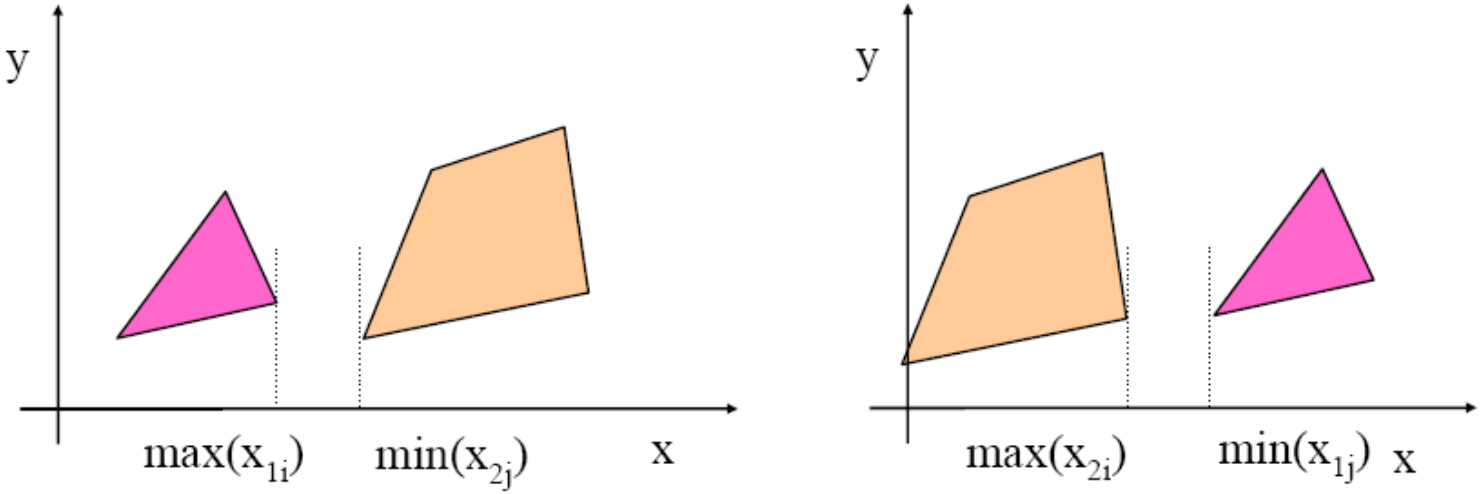
- Brz zahvat kojim se ustanovi da li se dva poligona sigurno ne prekrivaju ili se potencijalno prekrivaju.
- Neka su poligoni zadani vrhovima $P_1(V_{11} .. V_{1n})$ i $P_2(V_{21} .. V_{2m})$.
- Poligoni se ne prekrivaju ako vredi za svaki i, j:

  $\max(x_{1i}) < \min(x_{2j})$ ili

  $\max(x_{2i}) < \min(x_{1j})$ ili

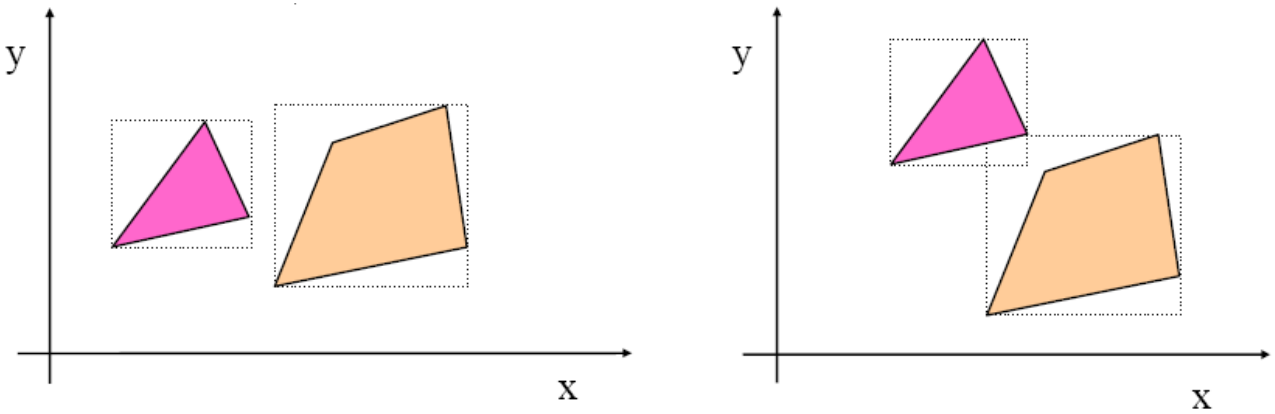
  $\max(y_{1i}) < \min(y_{2j})$ ili

  $\max(y_{2i}) < \min(y_{1j})$

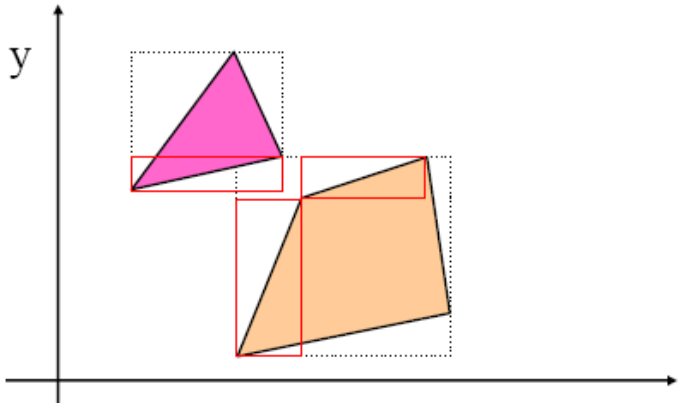


Min-maks provera

- Proveravamo da li se opisani pravougaonici (screen extent) prekrivaju.

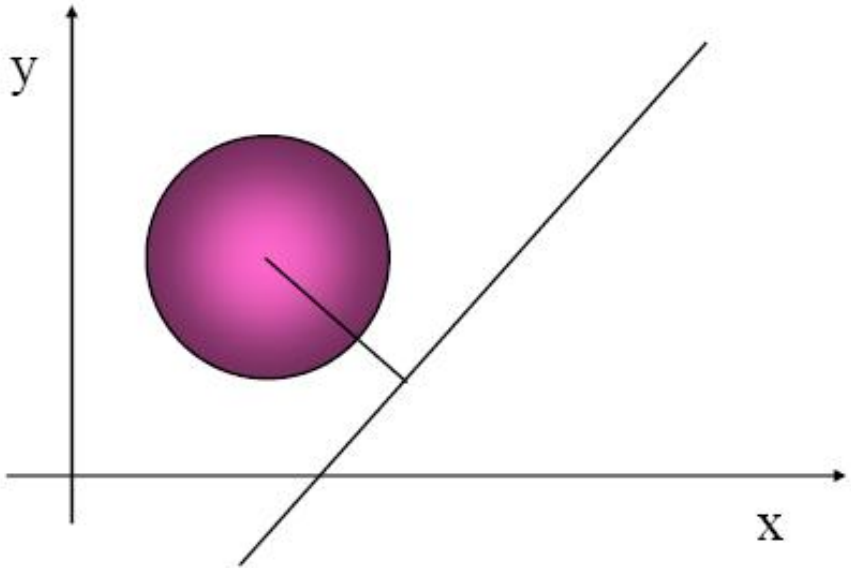


- Ukoliko se poligoni potencijalno prekrivaju potrebne su dodatne provere.
- Algoritam možemo primeniti i na bridove (ivice).



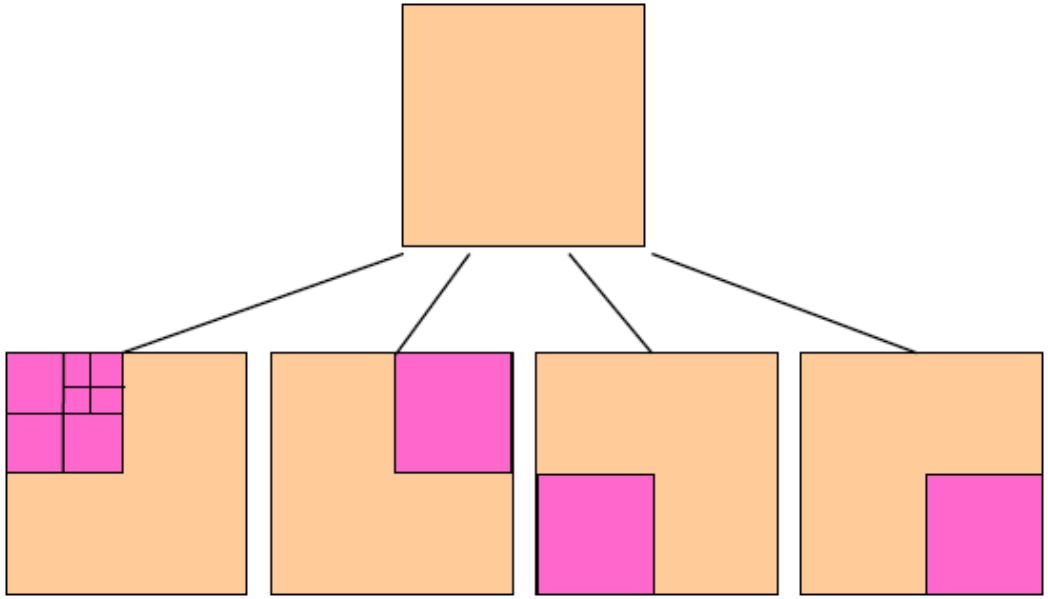
Proširenje algoritma Min-maks na 3D

- Proveravamo da li se kvadri (bounding box), koji obuhvataju tela ili delove tela preklapaju.
- Postupak se obično koristi kod detekcije kolizije.
- Umesto kvadra često se koristi kugla (sfera).
- Ukoliko je udaljenost pravca do središta kugle:
  - veća od poluprečnika, pravac ne seče kuglu.
  - manja od poluprečnika, pravac seče kuglu.



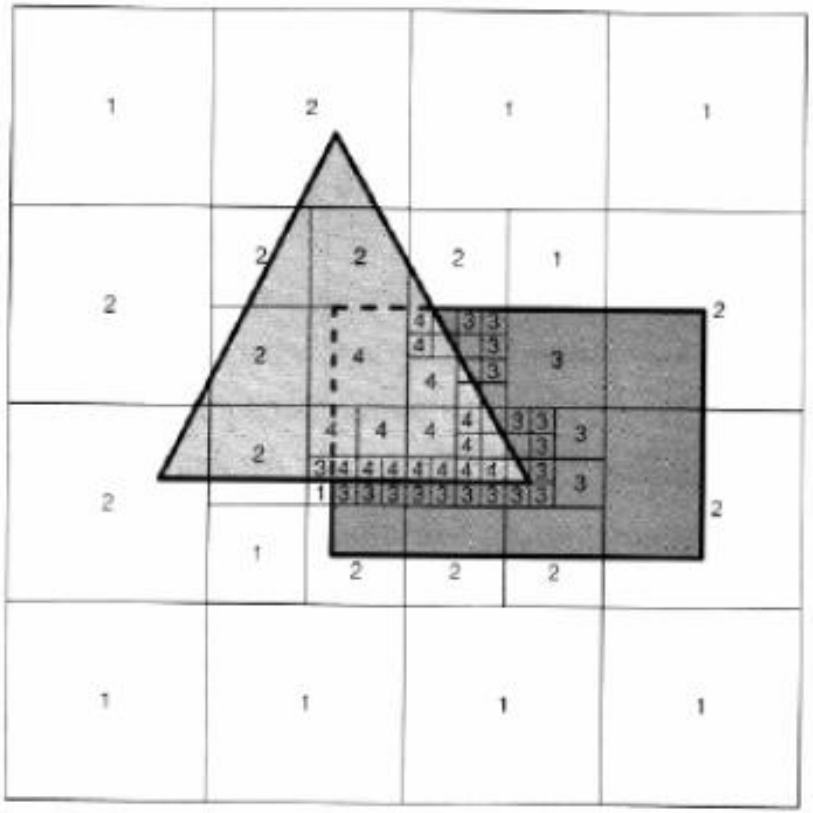
Warnock-ov postupak

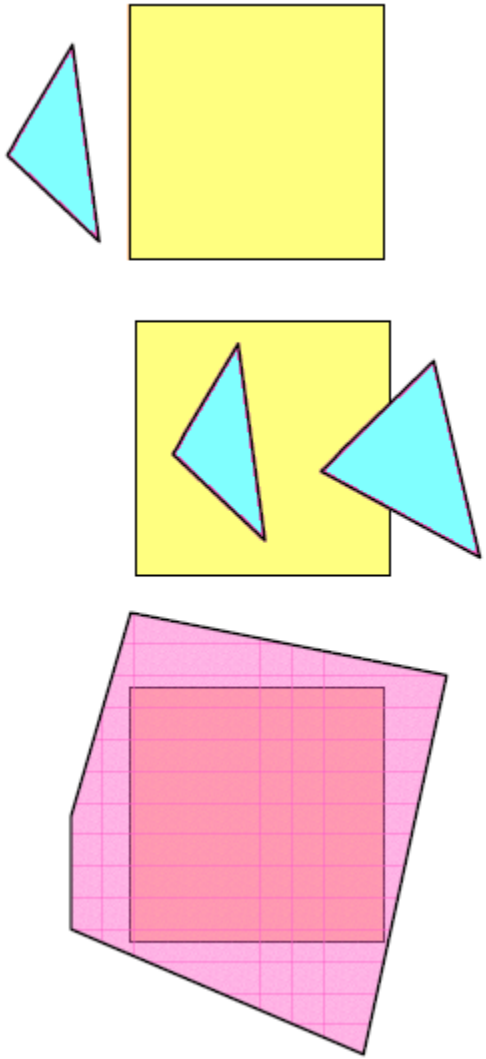
- Radi u prostoru slike (2D).
- Analizira se sadržaj ispitnog prostora koji je u početku jednak zaslonu.
- Mogući slučajevi:
    - prozor je prazan
    - scena u prozoru je jednostavna i moguće je prikazati istu
    - scena u prozoru je složena, rekurzivno se deli dalje



Podela prostora (quad tree)

- Poligoni se raspodele obzirom na relaciju s prozorom:
  - poligon je izvan prozora (uklonimo ih iz liste) (1)
  - poligon seče prozor ili je u prozoru (2)
  - poligon (jedan) prekriva prozor (3)
  - više poligona prekriva prozor (4)



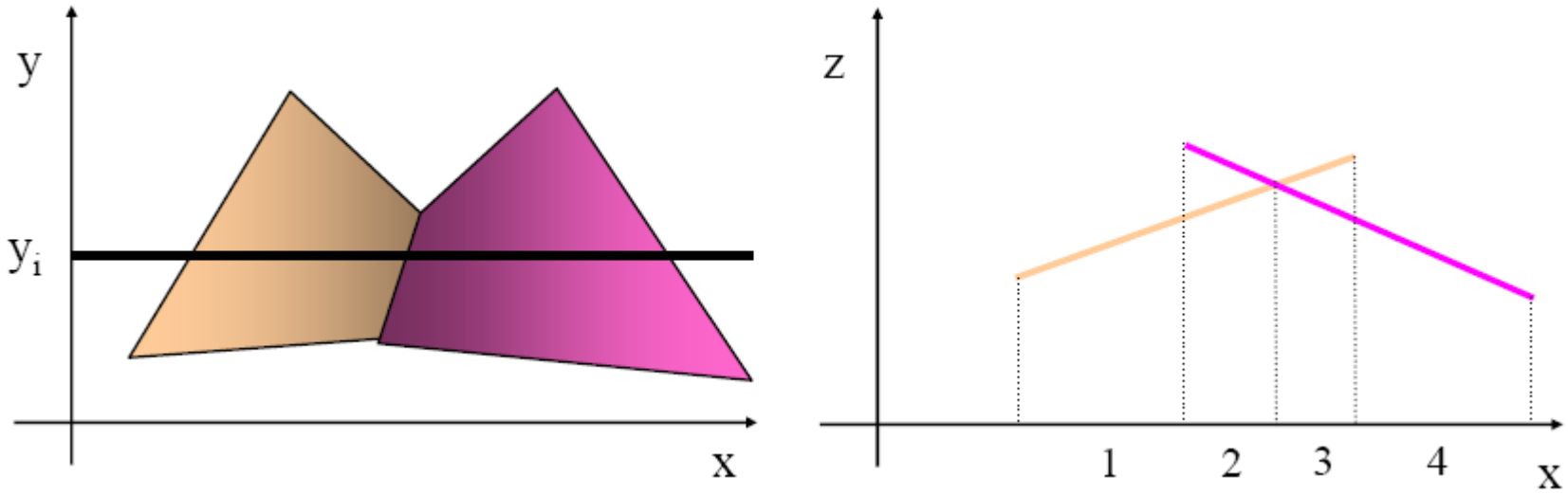


Watkins-ov postupak (scan line method)

- Radi u prostoru slike. Projektuju se poligoni na ravan xy.
- U ravni projekcije postavi se ispitna linija.
  1. Određivanje raspona uzorka (deo linije na kojoj se ne može dogoditi promena vidljivosti)

     Raspon uzorka je deo linije koji zadovoljava uslove:

     - broj segmenata u rasponu uzorka *konstantan*
     - projekcije preseka za $y = y_i$ unutar raspona uzorka u ravnini *xz ne seku* se unutar raspona uzorka (svaki presek u projekciji označava novi raspon uzorka)

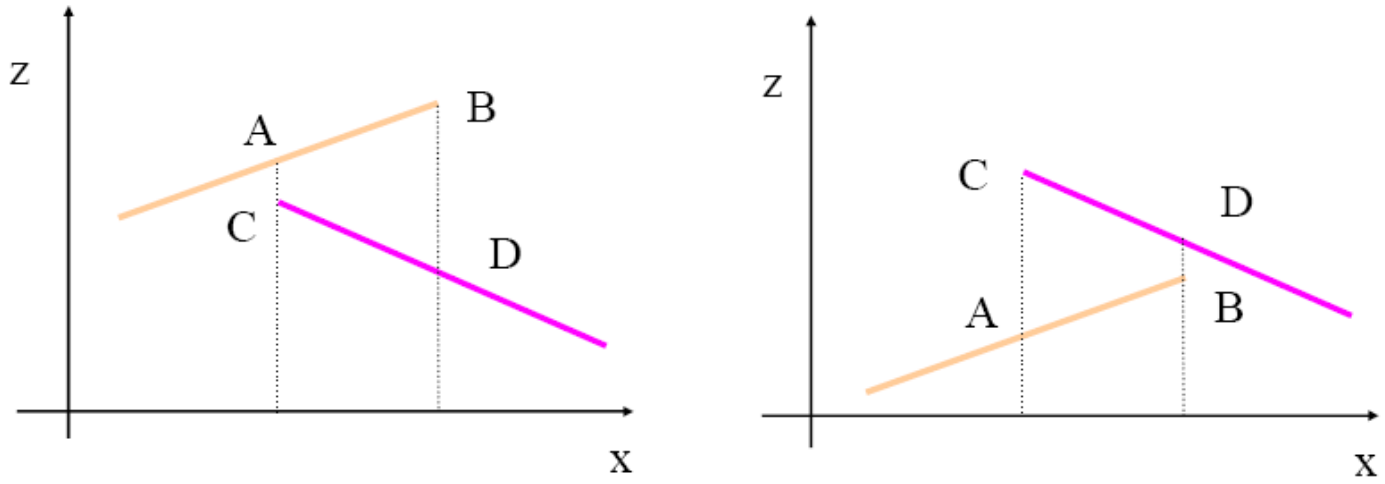


Watkins-ov postupak (scan line method)

2. Određivanje vidljivosti

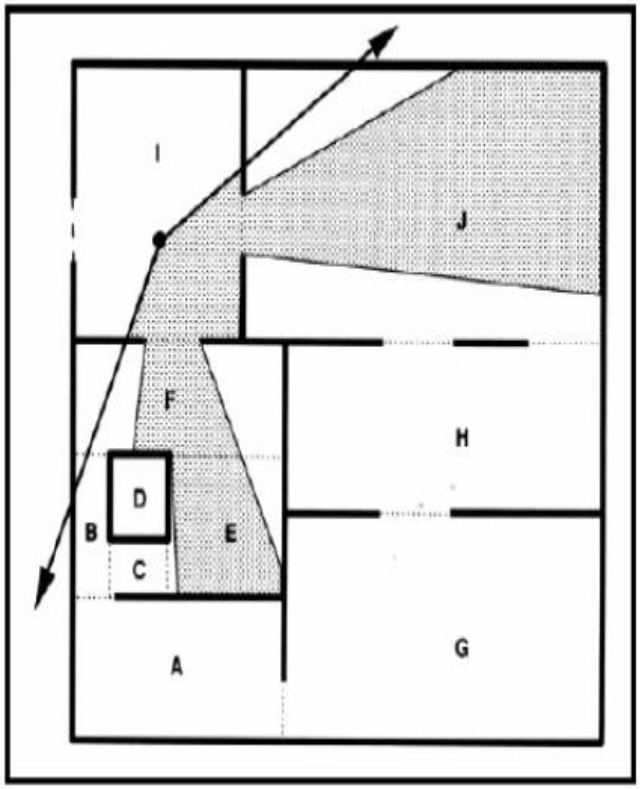
3. Raspon uzorka testiramo obzirom na vidljivost:
   - ako je broj segmenata na tekućoj liniji pretrage različit od broja segmenata na prethodnoj liniji pretrage ili
   - ako se krajnje tačke dva segmenta zamene po veličini koordinate z (na primer: tačke A i C te B i D) kad prelazimo iz tekuće u susednu liniju pretrage (tada se obe površine seku u prostoru između dve ravni pretrage).

– Ispitivanjem z-koordinate određuje se koji segment u rasponu uzorka prekriva druge segmente.

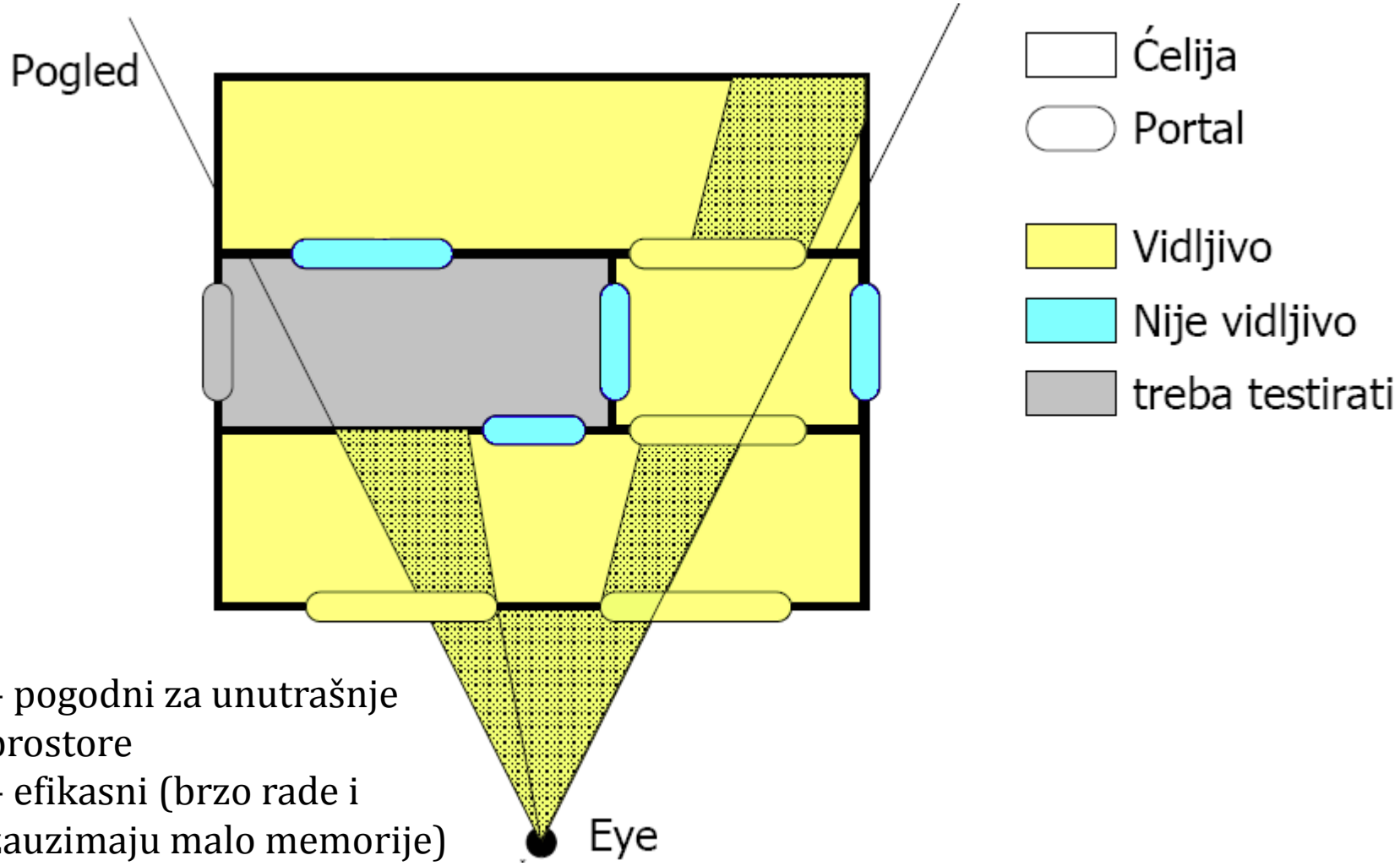


Uklanjanje poligona (culling)

- Uklanjanje poligona (culling)
  - uklanjanje zadnjih poligona
  - uklanjanje poligona (objekata) izvan piramide pogleda (view frustum)
  - uklanjanje zaklonjenih poligona (objekata) (occlusion culling)
- Ukoliko objekti nisu prozirni možemo ukloniti zadnje poligone.
- Ukoliko nema ogledanja (mirroring) mogu se ukloniti poligoni (objekti) koji nisu vidljivi.
- Potrebno je načiniti efikasne hijerarhijske grafove složenih scena (ćelije i portali).



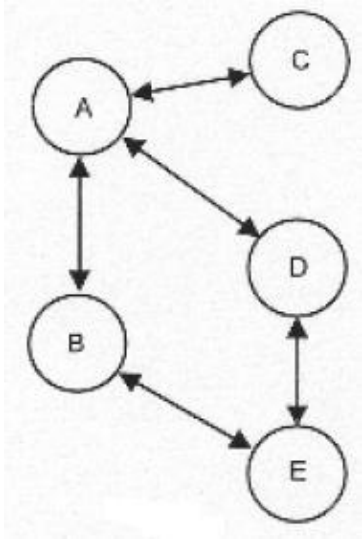
Ćelije i portali

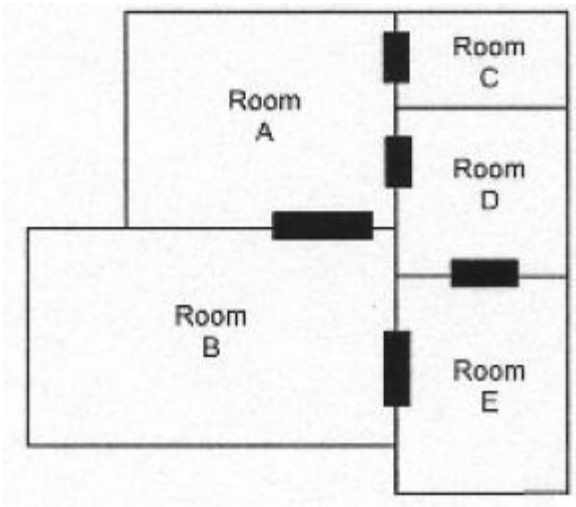


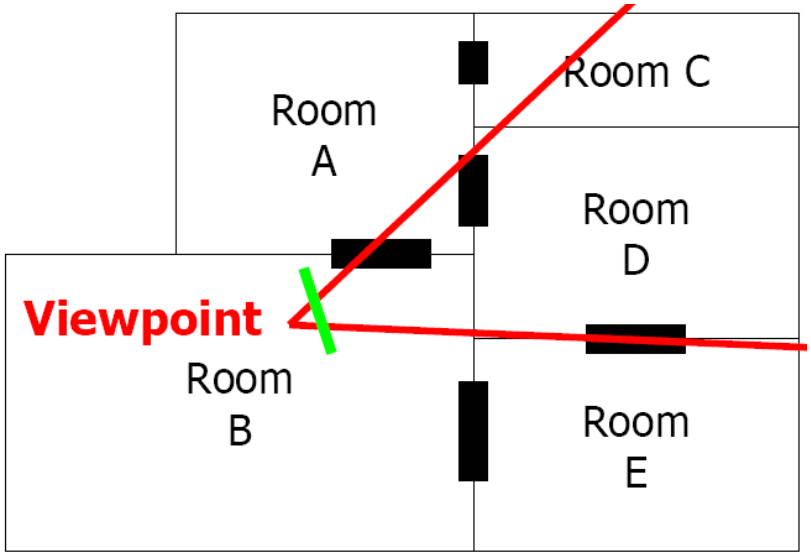
– pogodni za unutrašnje prostore
– efikasni (brzo rade i zauzimaju malo memorije)

PVS potencijalno vidljivi skup (Potentially visible set)

- Graf (scene) vidljivosti, samo se jedna vrata otvaraju.





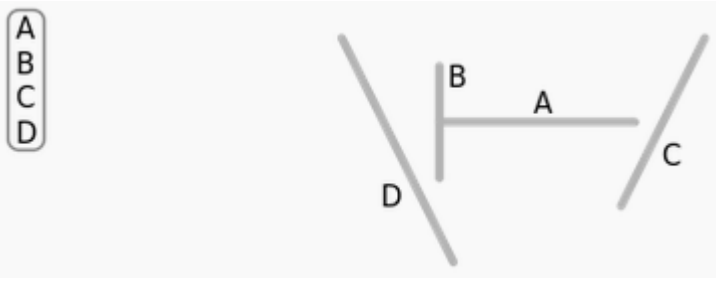


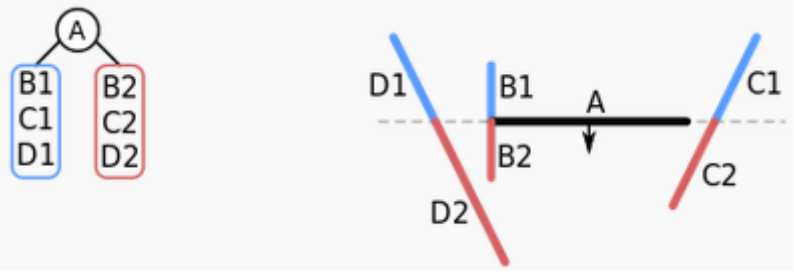
LOD, morfing, Grafički protočni sistem

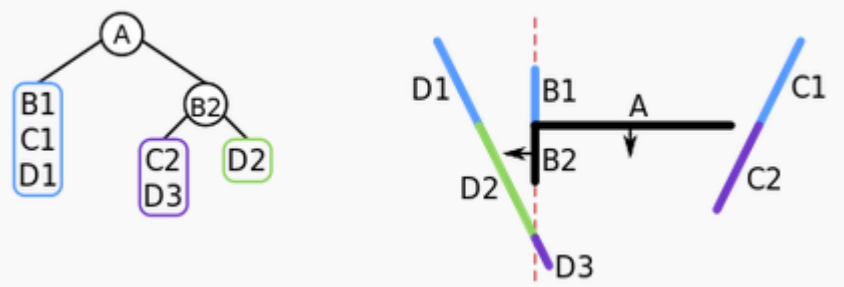
- Promena složenosti prikaza (LOD level of detail)
  - Unapred se pohrane objekti s različitim brojem poligona.
  - Zavisno od udaljenosti prikazuje se različit nivo detalja složenosti, a između se vrši interpolacija.
- Morfing: geometrijski preobražaj iz jednog oblika u drugi (morphing).
- Grafički protočni sistem – uklanjanje skrivenih linija i površina
  - geometrijski sistem (3D koordinate FP)
    - pretraga grafa scene
    - transformacije
    - proračun osvetljenja u vrhovima
    - uklanjanje zadnjih poligona (back face culling)
    - odsecanje (view volume clipping)
  - rasterski sistem (diskretne koordinate)
    - Z-buffer
    - senčenje
- U svim algoritmima prisutni su problemi zaokruživanja i numeričkih grešaka koji izazivaju vidljive promene na rezultatu.

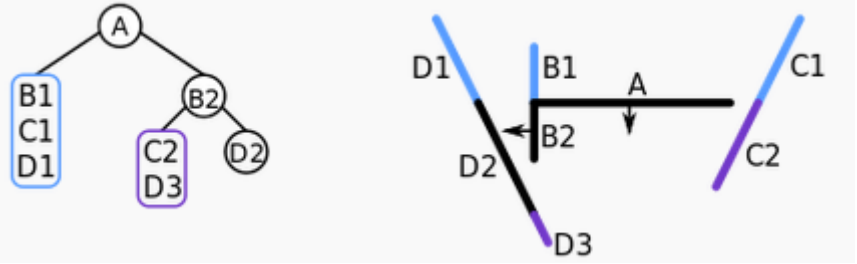
BSP

- BSP – stabla (Binary Space Partitioning)
- Binarna podela prostora.
  Koristi se npr. za detekciju kolizija.
- Algoritam je prikazan u dve dimenzije i lako se proširi na tri dimenzije.
- Prvo je potrebno sagraditi binarno stablo.
- Kreće se sa podelom prostora npr. pravcem koji je određen sa duži A.
  - dalje se posmatraju ostale duži ili njihovi delovi koji se nalaze
    - "ispred" pravca "A"
    - "iza" pravca "A"
  - kreira se binarno stablo i postupak se nastavlja u svakoj grani stabla za elemente (duži) u toj grani stabla



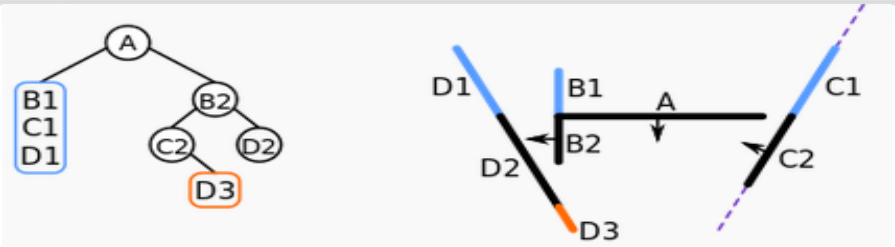


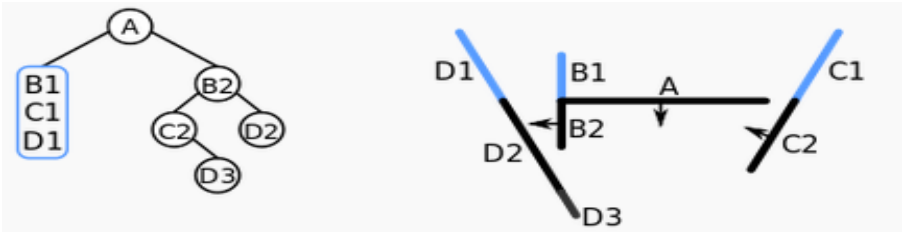


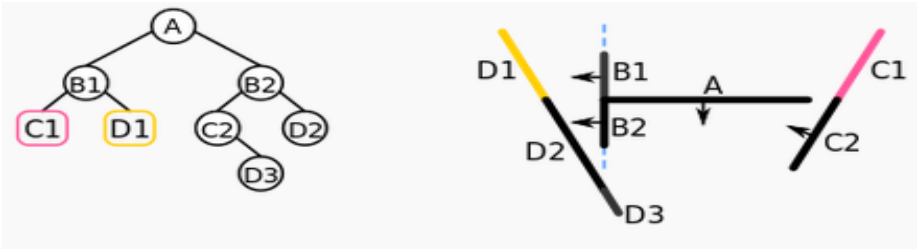


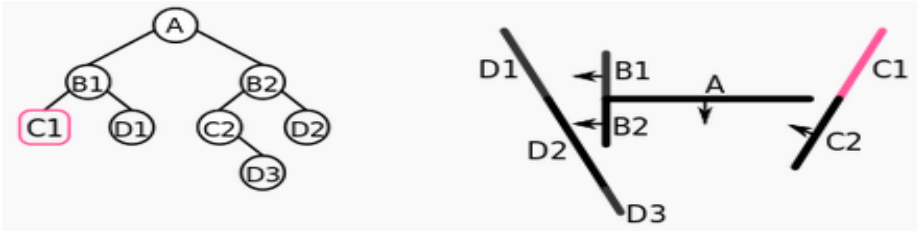
BSP

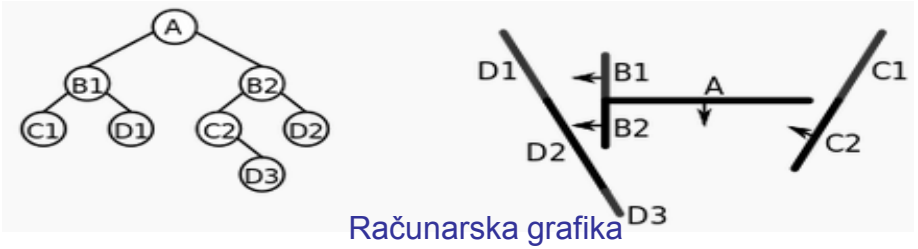
- Nastavak:





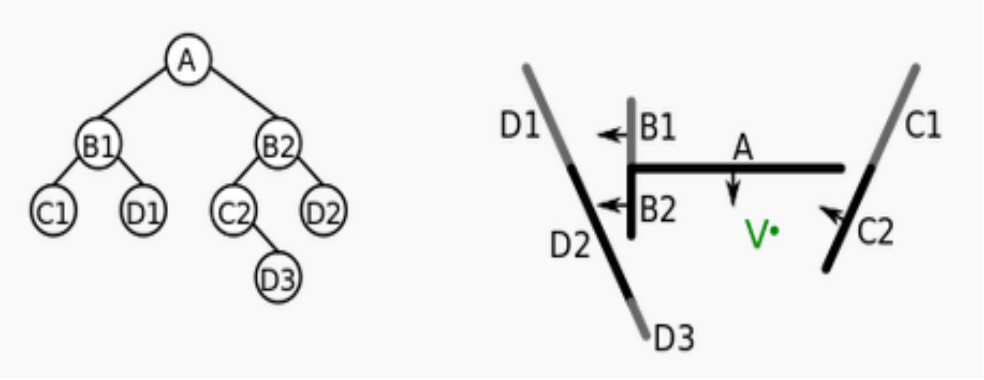






BSP – stabla

- BSP primer postavljanja kamere (položaja i pogleda):



  - Ako je tekući čvor list, renderuje se.
  - Inače, ako je lokacija pogleda V ispred tekućeg čvora:
    - Render dece BSP stabla iza tekućeg čvora, `BSP_display(tree->backChild);`
    - Render tekućeg čvora, `displayPolygon(tree->root);`
    - Render dece BSP stabla ispred tekućeg čvora, `BSP_display(tree->frontChild);`
  - U suprotnom, ako je lokacija pogleda V iza tekućeg čvora:
    - Render dece BSP stabla ispred tekućeg čvora, `BSP_display(tree->frontChild);`
    - Render tekućeg čvora, `displayPolygon(tree->root);`
    - Render dece BSP stabla iza trenutnog čvora `BSP_display(tree->backChild);`

`Rezultat bi bio:` (*D1*, *B1*, *C1*, *A*, *D2*, *B2*, *C2*, *D3*)

Zadatak

- Napisati program koji pretvara četvorostranu piramidu datu koordinatama:

  A(50,50,30)

  B(150,50,30)

  C(150,150,30)

  D(50,150,30)

  E(100,100,100)

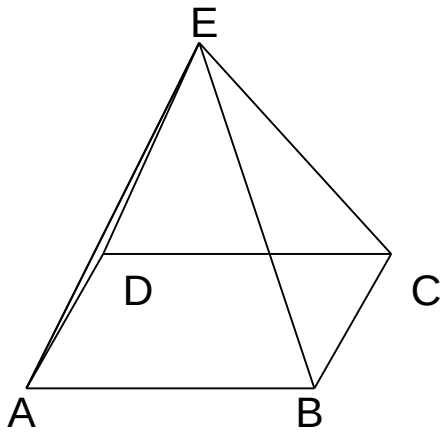
  u kvadrat čija je baza ista kao i date piramide:

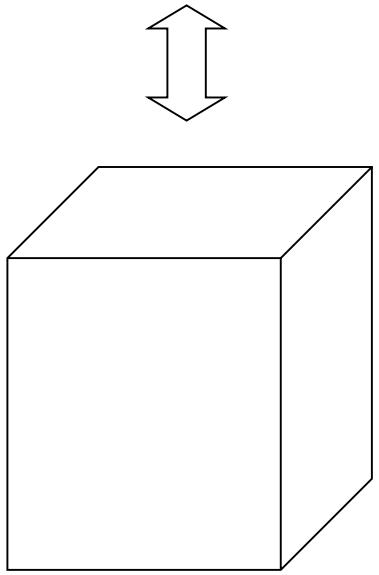
  Koraci pretvaranja su trigerovani pritiskom na taster:

  'k' za smer piramida→kvadrat

  'p' za smer kvadrat →piramida

  Morfing realizovati u 100 faza prelaza.





Primer 1/4

```
#pragma comment( lib, "opengl32.lib")
#pragma comment( lib, "glu32.lib")
#pragma comment( lib, "glut32.lib")
#include <GL/glut.h>
bool flag = false;
void line3d(float x1, float y1, float z1, float x2, float y2, float z2){
  glLineWidth(2.0);
  glBegin(GL_LINES);
    glVertex3f(x1,y1,z1);  glVertex3f(x2,y2,z2);
  glEnd();
}
struct point3d { float x; float y; float z; };
point3d mpoint[10]={
         50.0, 50.0, 30.0,        150.0, 50.0, 30.0,
        150.0,150.0, 30.0,         50.0,150.0, 30.0,
         50.0, 50.0, 30.0,        100.0,100.0,100.0,
        100.0,100.0,100.0,        100.0,100.0,100.0,
        100.0,100.0,100.0,        100.0,100.0,100.0 };
point3d mpoint2[10]={
         50.0, 50.0, 30.0,        150.0, 50.0, 30.0,
        150.0,150.0, 30.0,         50.0,150.0, 30.0,
         50.0, 50.0, 30.0,         50.0, 50.0,100.0,
        150.0, 50.0,100.0,        150.0,150.0,100.0,
         50.0,150.0,100.0,         50.0, 50.0,100.0 };

float t=0;
static float fRotAngle = -60.0f;
```

Primer 2/4

```
void Redraw(){
  glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );  glLoadIdentity();
  glTranslatef( 0.0f, 0.0f, 0.0f );  glRotatef( fRotAngle, 1.0f, 0.0f, 1.0f );
  fRotAngle+= 0.5f; if(fRotAngle>360.0) fRotAngle = 0;
  int i;
    for( i=0; i<4; i++ )
      line3d( mpoint  [i].x,mpoint  [i].y,mpoint  [i].z,
              mpoint[i+1].x,mpoint[i+1].y,mpoint[i+1].z);
    for( i=5; i<9; i++ )
      line3d( mpoint[i].x   + (mpoint2[i].x   - mpoint[i].x)*t,
              mpoint[i].y   + (mpoint2[i].y   - mpoint[i].y)*t,
              mpoint[i].z   + (mpoint2[i].z   - mpoint[i].z)*t,
              mpoint[i+1].x + (mpoint2[i+1].x - mpoint[i+1].x)*t,
              mpoint[i+1].y + (mpoint2[i+1].y - mpoint[i+1].y)*t,
              mpoint[i+1].z + (mpoint2[i+1].z - mpoint[i+1].z)*t
              );
    for( i=0; i<4; i++ )
      line3d( mpoint[i].x,mpoint[i].y,mpoint[i].z,
              mpoint[i+5].x + (mpoint2[i+5].x-mpoint[i+5].x)*t,
              mpoint[i+5].y + (mpoint2[i+5].y-mpoint[i+5].y)*t,
              mpoint[i+5].z + (mpoint2[i+5].z-mpoint[i+5].z)*t
            );
   glutSwapBuffers();  }
void Keyboard( unsigned char uKey, int x, int y ){
switch ( uKey ){   case 0x1B: exit( 0 ); break;
                   case 'q' : if(t<1.0) t+=0.01; break;
                   case 'p' : if(t>0.0) t-=0.01; break;}}
```

Primer 3/4

```
void Resize( int iWidth, int iHeight ){
    glViewport( 0, 0, (GLsizei)iWidth, (GLsizei)iHeight );
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    glFrustum( -10.0f, 10.0f, -10.0f, 10.0f, 10.0f, -10.0f );
    glMatrixMode( GL_MODELVIEW );
    gluLookAt(0,0,30.0,0,0,0,0,1.0,0);
}
void Timer( int iValue ){glutPostRedisplay(); glutTimerFunc( 10, Timer, iValue );
}
int main( int argc, char **argv ){
    for (int i=0; i<sizeof(mpoint)/sizeof(point3d); i++)
    {
      float umanji = 300.0;
      mpoint [i].x/=umanji;     mpoint [i].y/=umanji;     mpoint [i].z/=umanji;
      mpoint2[i].x/=umanji;     mpoint2[i].y/=umanji;     mpoint2[i].z/=umanji;
    }
    glutInitDisplayMode( GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE );
    glColor3f(1.0,1.0,1.0);
    glutInitWindowPosition( 0, 0 );    glutInitWindowSize( 600, 600 );
    glutInit( &argc, argv );
    glutCreateWindow( "Morfing" );
    Resize(600,600);
    glutKeyboardFunc( Keyboard ); glutReshapeFunc( Resize );
    glutDisplayFunc( Redraw );
    glutTimerFunc( 1, Timer, 100 );
    glutMainLoop();    return 0;}
```