Računarska grafika
(2OER7O02)

OpenGL – Pogled u 3D

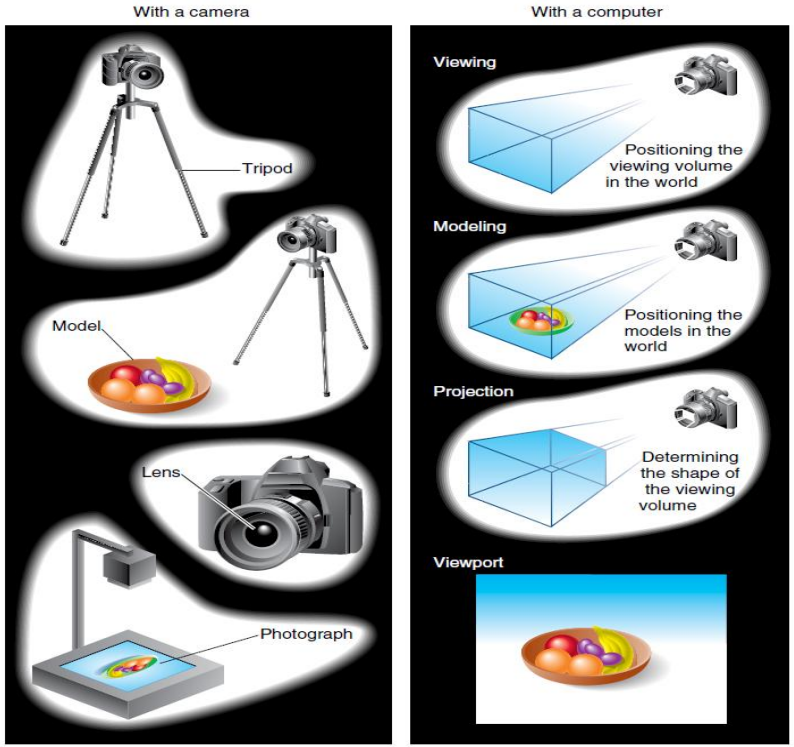
Auditivne vežbe

Koraci u kreiranju slike

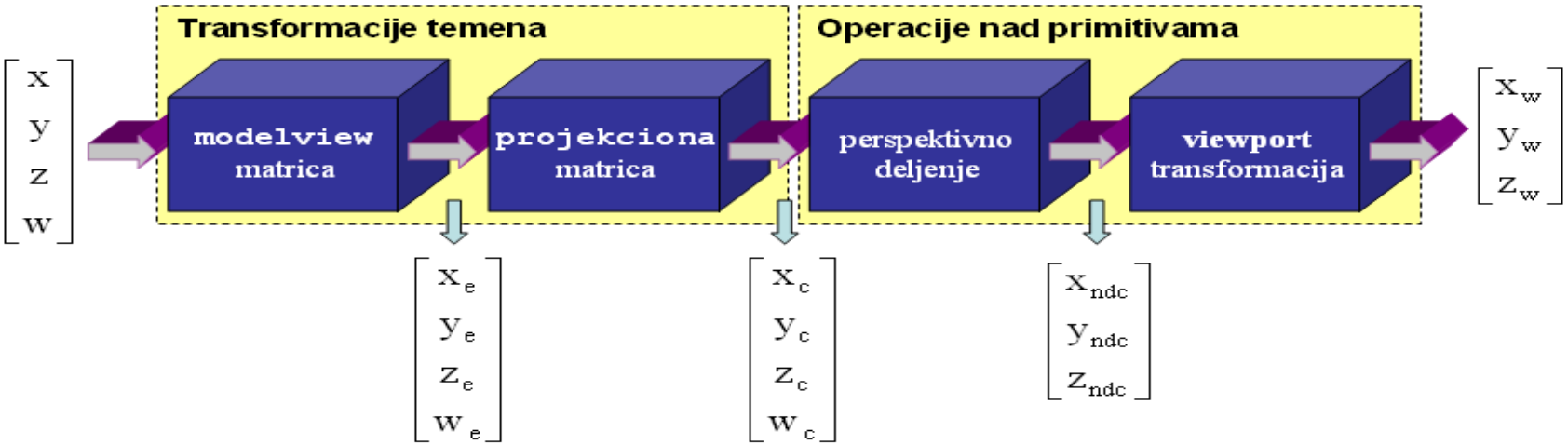
- napraviti ili pribaviti model koji će fotografisati i postaviti ga tako da se dobro uklapa u kompoziciju sa drugim objektima (skup svih objekata koji se vide i njihov uzajamni odnos nazvaćemo „scenom“),
- postaviti foto-aparat na odgovarajuću poziciju, tako da model bude dobro vidljiv i da se postigne željeni efekat cele kompozicije,
- izabrati sočivo na objektivu i podesiti *zoom*,
- nakon formiranja slike, razviti film i prilagoditi sliku određenom formatu

Analogija

- Modeliranje
- pogled (stajna tačka)
- Projekcija
- preslikavanje na ekran



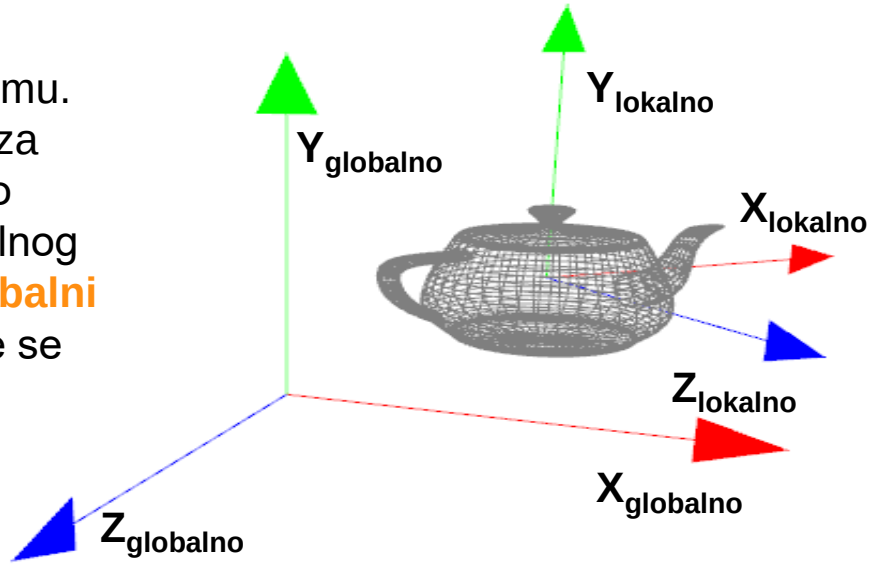
Proces transformacije temena



U digitalnom svetu, objekti su predstavljeni svojim matematičkim modelom, tj. koordinatama (ali i ostalim parametrima) svojih temena, a svaki od prethodnih koraka u procesu formiranja slike predstavlja jednu matematičku transformaciju nad tim temenima.

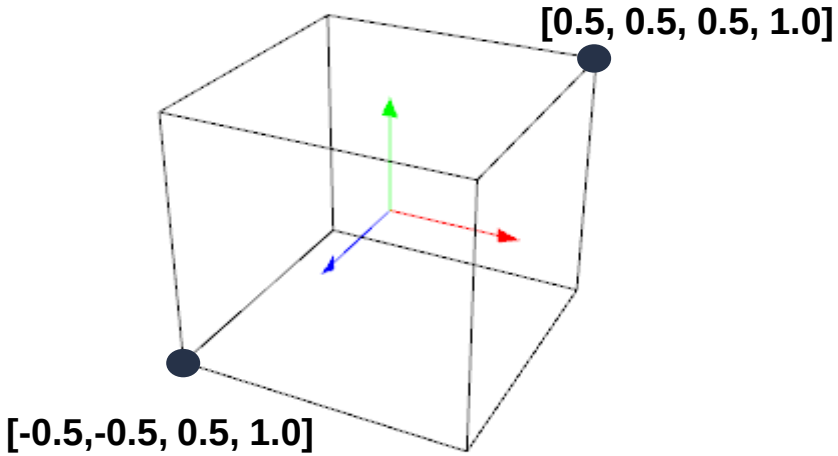
Globalni i lokalni koordinatni sistemi

Objekti se zadaju koordinatama svojih temena u lokalnom koordinatnom sistemu. **Lokalni koordinatni sistem** vezan je za objekat i najčešće se smešta u njegovo središte. Pomeranjem i rotiranjem lokalnog koordinatnog sistema u odnosu na **globalni (svetski) koordinatni sistem**, definiše se položaj objekta u sceni.



Primer jedinične kocke

Za slučaj jedinične kocke (kocka čije su stranice dužine 1), donje levo teme prednje stranice imaće koordinate $[-0.5, -0.5, 0.5, 1.0]^T$ u lokalnom koordinatnom sistemu. Gornje desno teme prednje stranice biće na koordinatama $[0.5, 0.5, 0.5, 1.0]^T$.



Modelview transformacija

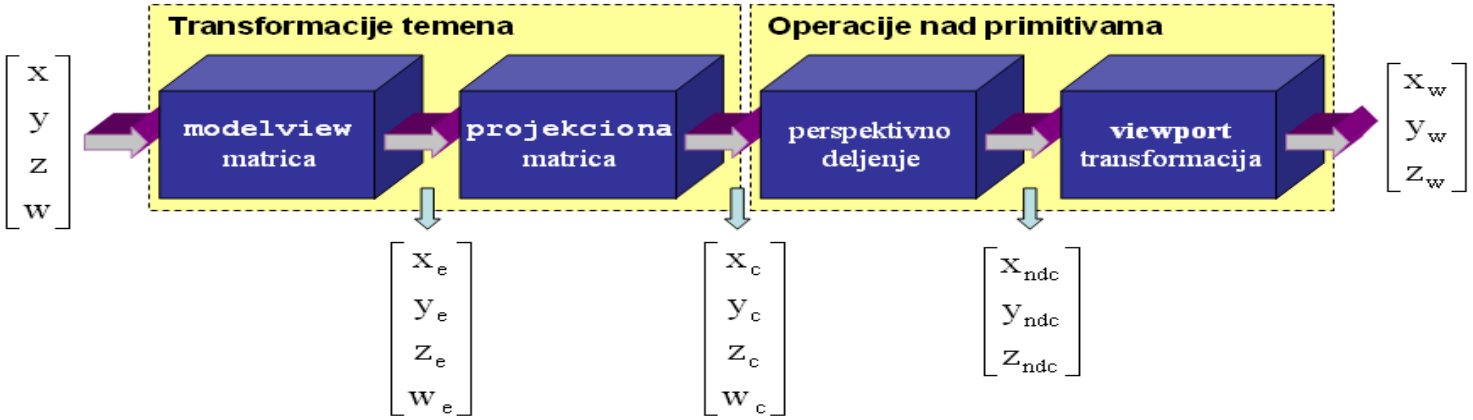
Da bi se jedinična kocka postavila na proizvoljno mesto u prostoru, potrebno je primeniti odgovarajuću transformaciju nad temenima kocke. Ta transformacija naziva se ***modelview*** transformacija, tj. transformacija modeliranja i pogleda.

Pod pojmom modeliranja podrazumeva se razmeštanje objekata po sceni.

Pogled predstavlja postavljanje i orijentaciju posmatrača, odnosno, kamere.

Zašto su pomešane ove dve naizgled potpuno različite stvari?

Modelview transformacija



$V_e = M_{mv} \cdot V$

$$\begin{bmatrix} x_e \\ y_e \\ z_e \\ w_e \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Definisanje matrica transformacija

- Sve transformacione matrice dimenzija 4x4
- U OpenGL-u matrice su memorisane kao vektori
- Elementi su smešteni po kolonama
- Primer deklaracije: *double M[16]*

$$M = \begin{bmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{bmatrix}$$

Transformacione matrice

Postoje tri matrice koje definišu način iscrtavanja scene:

- **Modelview matrica** transformiše koordinate iz svetskog koordinatnog sistema u sistem vezan za oko posmatrača.

- **Projekciona matrica** definiše „vidljivi“ prostor i transformiše 3D koordinate iz sistema vezanog za oko posmatrača u *clip* koordinate. U *clip* koordinatnom sistemu čitav vidljivi prostor ograničen je kockom koja se prostire od –w do +w po sve tri koordinate.

- **Teksturna matrica** omogućuje definisanje složenih transformacija nad teksturnim koordinatama.

Selekcija i učitavanje matrice

Da bismo mogli da modifikujemo neku od datih matrica, najpre moramo da je selektujemo. Selekcija se obavlja pozivom funkcije **glMatrixMode()** i prosleđivanjem jednog od 3 parametra:

- **GL_MODELVIEW**,
- **GL_PROJECTION** ili
- **GL_TEXTURE**.

Učitavanje se ostvaruje funkcijom **glLoadMatrix{fd}()**

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

```
float mat[16] = { 1.0, 0.0, 0.0, 0.0,
                  0.0, 1.0, 0.0, 0.0,
                  0.0, 0.0, 1.0, 0.0,
                  0.0, 0.0, 0.0, 1.0};
glMatrixMode(GL_MODELVIEW);
glLoadMatrixf(mat);
```

Osnovne modelview transformacije

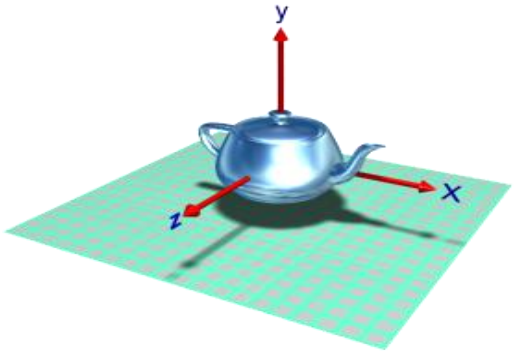
Postoje tri osnovne transformacije koje definišu položaj, orijentaciju i veličinu geometrijskih objekata u sceni:

- translacija, void **glTranslate**{fd}(**TYPE***x*, **TYPE** *y*, **TYPE** *z*);
- rotacija i void **glRotate**{fd}(**TYPE** *angle*, **TYPE** *x*, **TYPE** *y*, **TYPE** *z*);
- skaliranje. void **glScale**{fd}(**TYPE***x*, **TYPE** *y*, **TYPE** *z*);

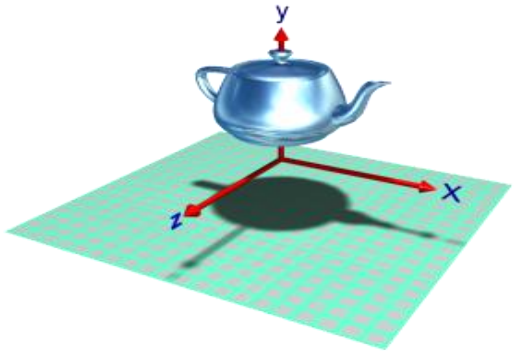
Translacija

$$x' = x + a$$
$$y' = y + b$$
$$z' = z + c$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \\ z+c \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



glTranslate*(*a*, *b*, *c*)



Translacija duž **Y** ose

Rotacija

$$\mathbf{R}_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

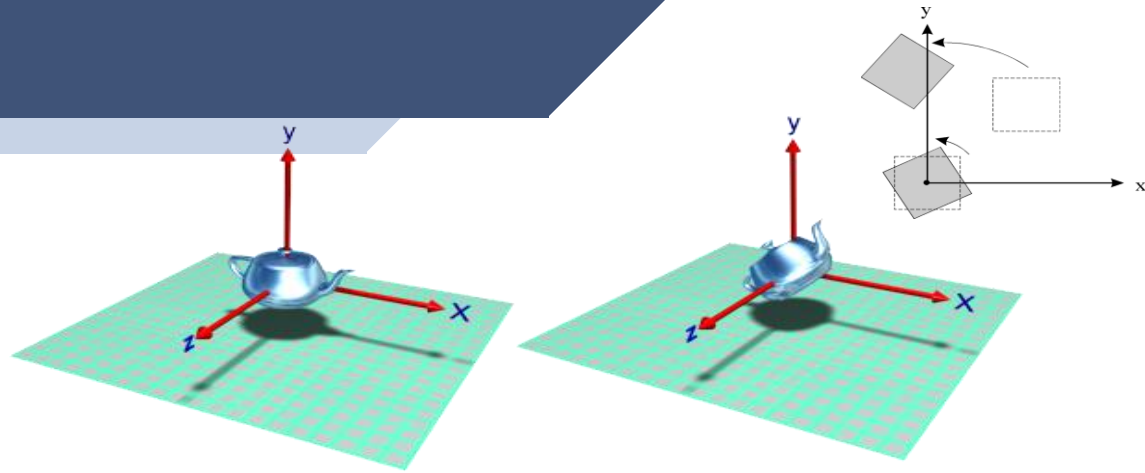
$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

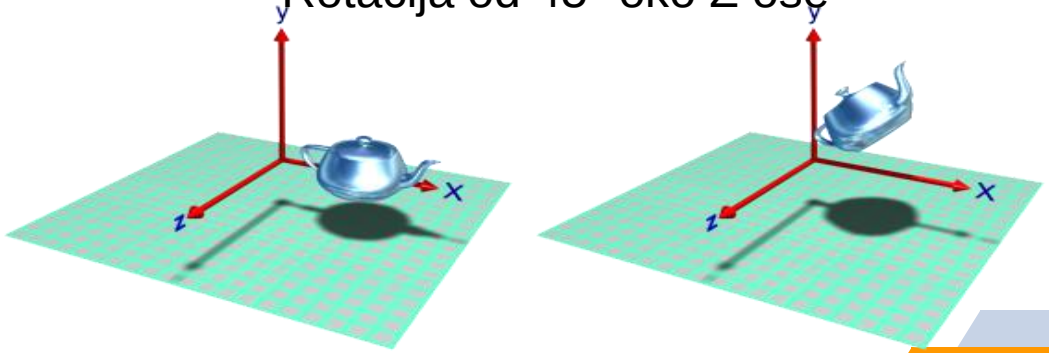
$$\begin{matrix} \cos(-\theta) = \cos\theta \\ \sin(-\theta) = -\sin\theta \end{matrix} \Rightarrow \mathbf{R}^{-1}(\theta) = \mathbf{R}(-\theta) = \mathbf{R}^T(\theta)$$

Ako je **M** $^{-1}$ = **M**^{T} tada je **M** ortogonalna. Sve ortogonalne matrice su rotacije oko koordinatnog početka.

glRotate*(θ, x, y, z)

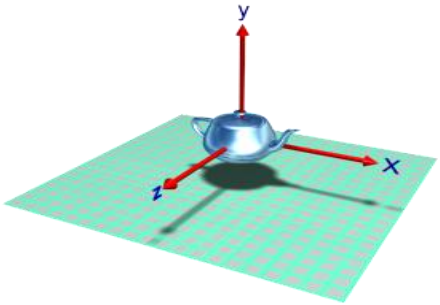


Rotacija od 45^o oko Z ose

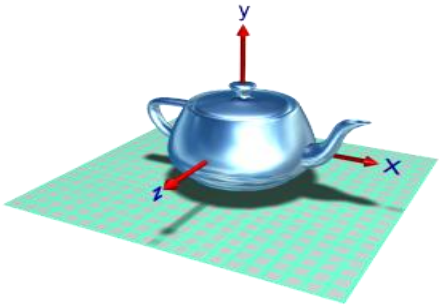


Rotacija nakon translacija

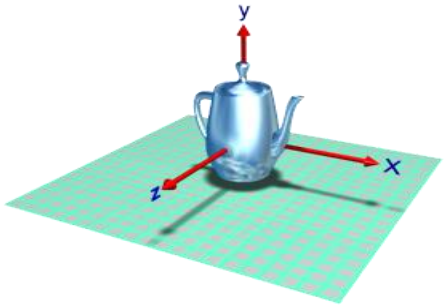
Skaliranje



Original

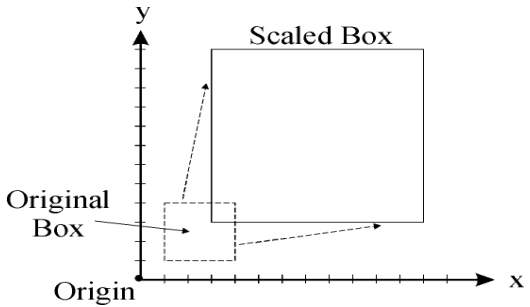


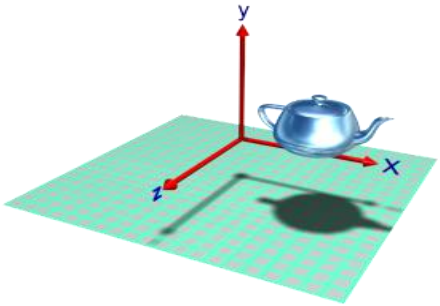
Skaliranje po svim osama



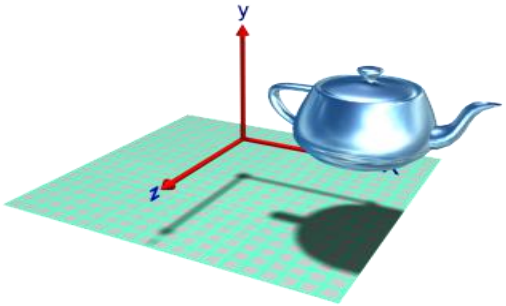
Skaliranje po Y osi

glScale*(s_x, s_y, s_z**)**





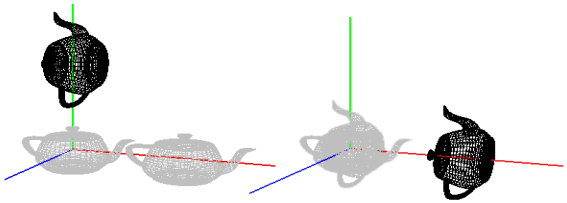
Translacija



Skaliranje nakon translacije

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ s_z z \\ w \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Kombinovanje transformacija



```
glTranslatef(1.5, 0.0, 0.0);
glRotatef(90.0,0.0,0.0,1.0);
glutWireTeapot(0.5);
```

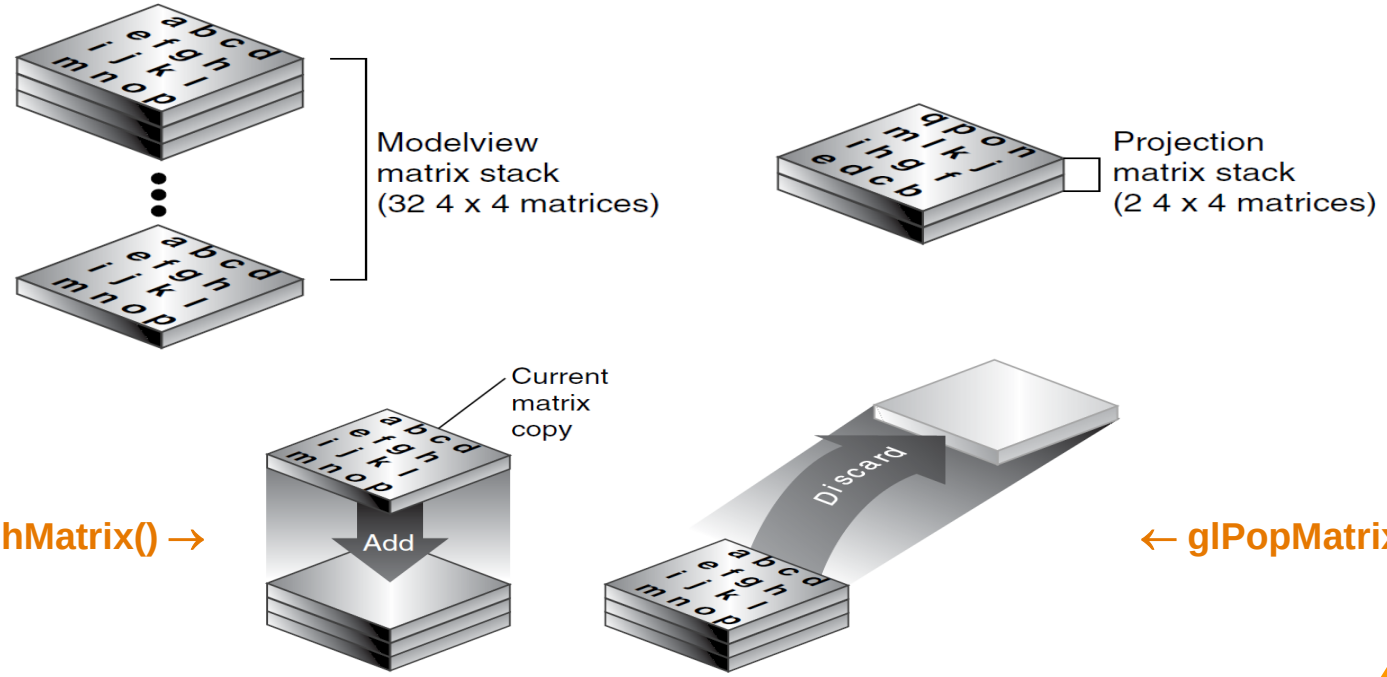
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glMultMatrixf(N);                /* primeni transformaciju N */
glMultMatrixf(M);                /* primeni transformaciju M */
glMultMatrixf(L);                /* primeni transformaciju L */
glBegin(GL_POINTS);
   glVertex3f(v);                /* crtaj teme v */
glEnd();
```

I → N → NM → NML → NMLv → N(M(Lv))

Lokalni i globalni sistem

- Da bi olakšali kombinovanje transformacija, uvešćemo dva načina razmišljanja:
  - razmišljanje u globalnom (svetskom) koordinatnom sistemu i
  - razmišljanje u lokalnom koordinatnom sistemu.
- Ako razmišljamo u **globalnom koordinatnom sistemu**, položaj objekta i transformacije uvek ćemo vršiti u odnosu na koordinatni početak tog sistema. U globalnom koordinatnom sistemu transformacije se u programskom kodu navode u redosledu **suprotnom** od onoga kojim se primenjuju.
- U **lokalnom koordinatnom sistemu** transformacijama zapravo pomeramo lokalni koordinatni sistem. Redosled izvršavanja transformacija u tom slučaju je **isti** kao i redosled navođenja u programskom kodu.

Stekovi matrica



glPushMatrix()/glPopMatrix()

```
draw_wheel_and_bolts()
{
    long i;
    draw_wheel();
    for(i=0;i<5;i++){
      glPushMatrix();
        glRotatef(72.0*i,0.0,0.0,1.0);
        glTranslatef(3.0,0.0,0.0);
        draw_bolt();
      glPopMatrix();
    }
}
```

```
draw_body_and_wheel_and_bolts() {
    draw_car_body();
    glPushMatrix();
      /*move to first wheel position*/
      glTranslatef(40,0,30);
      draw_wheel_and_bolts();
    glPopMatrix();
    glPushMatrix();
      /*move to 2nd wheel position*/
      glTranslatef(40,0,-30);
      draw_wheel_and_bolts();
    glPopMatrix();
      /*draw last two wheels similarly*/
      ...
}
```

Usmeravanje pogleda

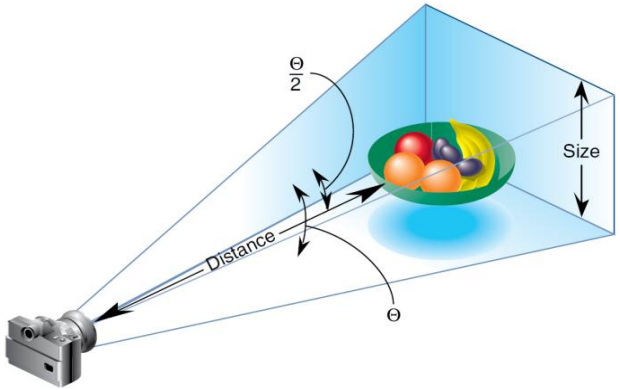
```
void gluLookAt( GLdouble eyex,    GLdouble eyey,    GLdouble eyez,
                GLdouble centerx, GLdouble centery, GLdouble centerz,
                GLdouble upx,     GLdouble upy,     GLdouble upz );
```

Prva tri parametra određuju položaj posmatrača (tj. oka), sledeća tri koordinate tačke u koju se gleda, a poslednja tri vektor upravan na pogled (*up* vektor).

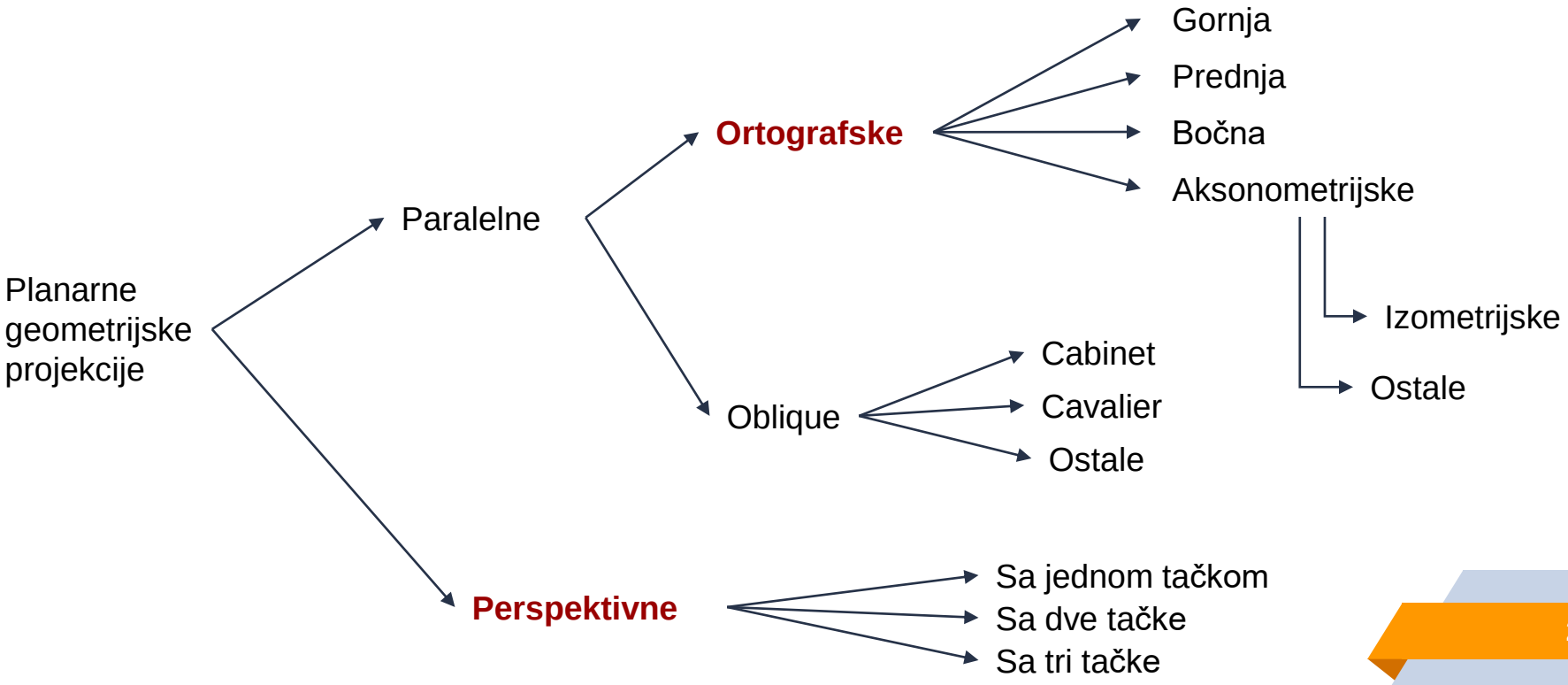
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(10.0, 5.0, 1.0, -5.0, 0.0, 5.0, 0.0, 1.0, 0.0);
```

Projekcije

Projekciona matrica definiše „vidljivi“ prostor i transformiše 3D koordinate iz sistema vezanog za oko posmatrača u *clip* koordinate. U *clip* koordinatnom sistemu čitav vidljivi prostor ograničen je kockom koja se prostire od –w do +w po sve tri koordinate. Sve primitive van ovog prostora se odbacuju, a one koje presecaju kocku se „odsecaju“ na mestima preseka. Zato se ovaj koordinatni sistem i naziva *clip* (eng. odsecanje).

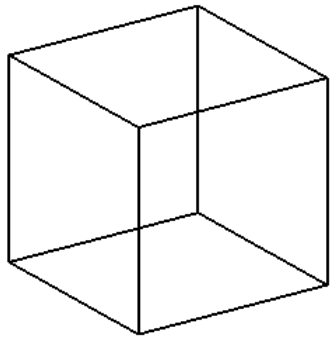


Projekcije


Planarne geometrijske projekcije
Paralelne
Perspektivne
Ortografske
Oblique
Gornja
Prednja
Bočna
Aksonometrijske
Izometrijske
Ostale
Cabinet
Cavalier
Ostale
Sa jednom tačkom
Sa dve tačke
Sa tri tačke

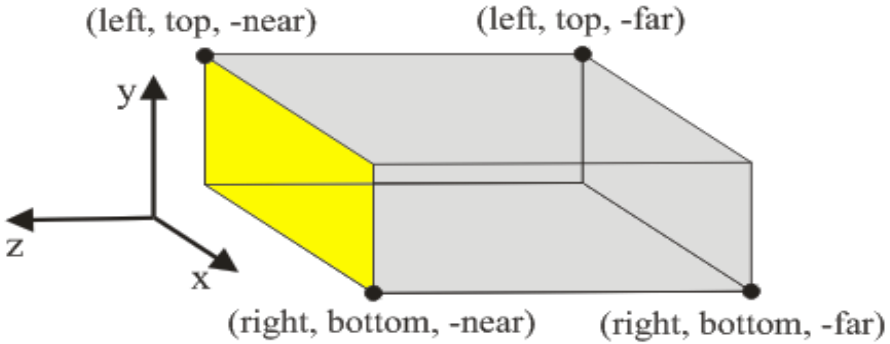
Ortografska projekcija

Ortografska projekcija je projekcija koja čuva dužine i paralelnost linija. Udaljeni objekti se ne smanjuju, a bliski objekti se ne deformišu. Površine koje su paralelne projekcionoj ravni zadržavaju odnos visine i širine, pa je moguće meriti delove tih površina na osnovu ortografske projekcije.



Ortografska projekcija

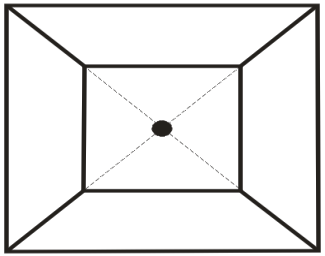
void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);

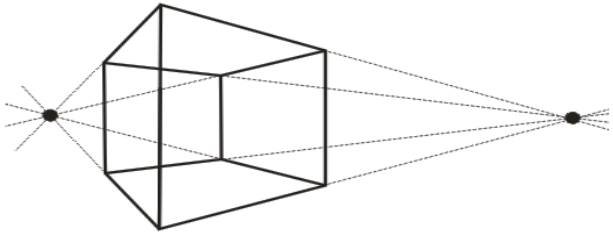


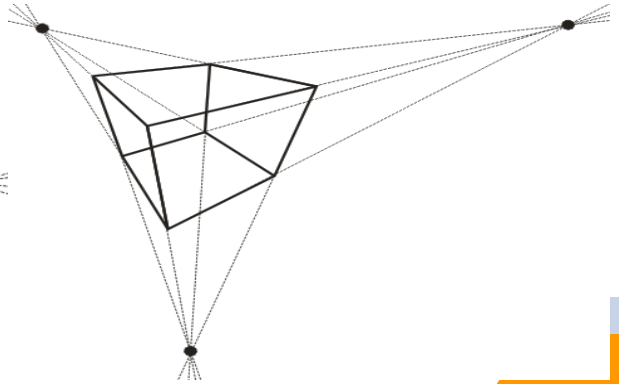

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-2.0, 2.0, -2.0, 2.0, 1.0, 100.0);
glMatrixMode(GL_MODELVIEW);
```

Perspektivna projekcija

Perspektivna projekcija je mnogo bliža načinu na koje oko ili kamera sagledavaju svet oko sebe. Sa uvođenjem perspektive predmeti postaju sve manji kako se udaljavaju od posmatrača, a paralelne linije prestaju biti paralelne (osim ako se ne nalaze u ravni paralelnoj sa projekcionom ravni).

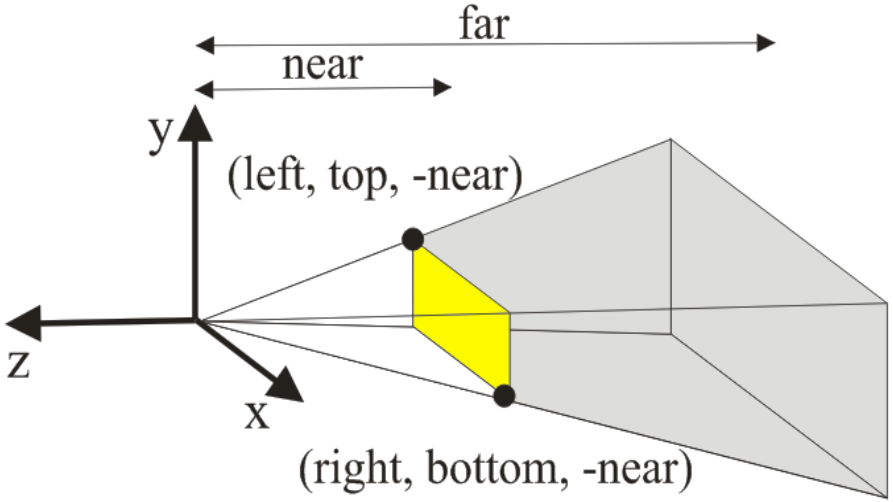






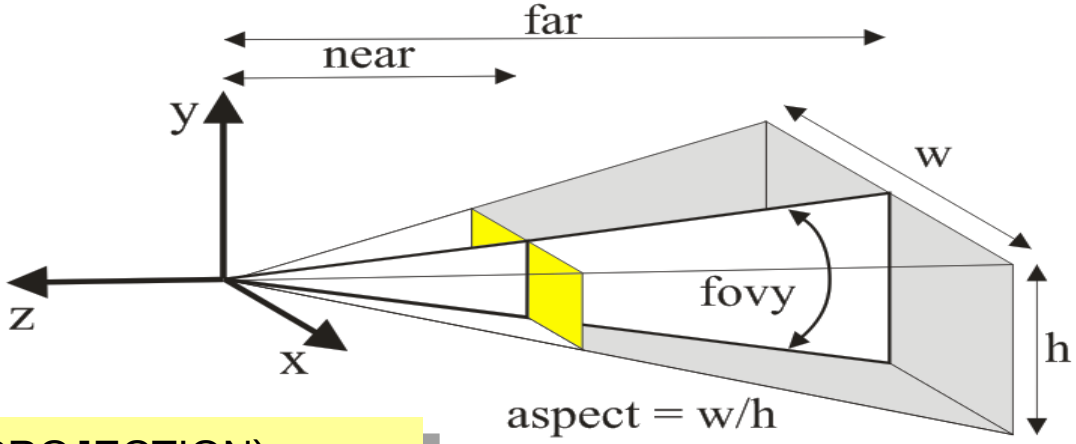
Perspektivna projekcija

void glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);



Perspektivna projekcija

void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble near, GLdouble far);




```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(45.0, aspect, 1.0, 100.0);
glMatrixMode(GL_MODELVIEW);
```

Standardna sočiva

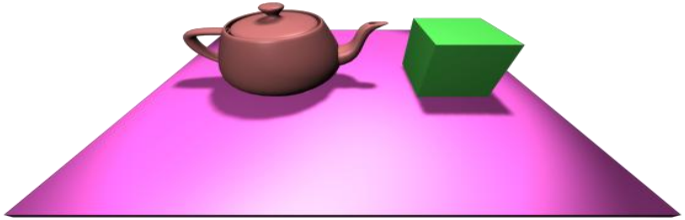
- **telefoto sočiva** (uvećavaja prikaz, približavaju udaljene objekte) >50mm
- **48.24 mm** – ekvivalent ljudskog vida
- **širokougaona sočiva** (obuhvataju veću scenu – umanjuju prikaz) <50mm
  - 35 mm i 28 mm – standardna širokougaona sočiva
  - 10-15 mm – “riblje oko” (sverno sočivo), velika distorzija

Sočivo	FOV
15 mm	100.389
20 mm	83.974
24 mm	73.74
28 mm	65.47
35 mm	54.432
50 mm	39.598
85 mm	23.913
135 mm	15.189
200 mm	10.286

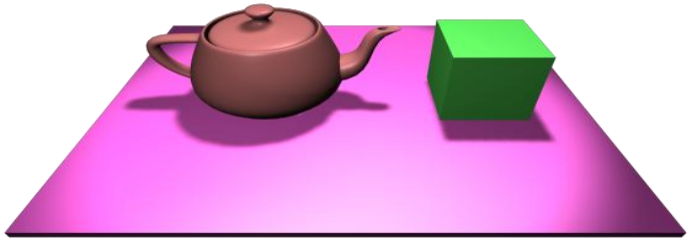
Koristiti FOV u opsegu: **30-60^{o},**

za realne scene, da ne bi došlo do distorzije.

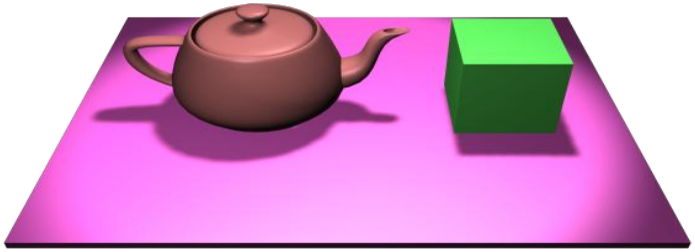
Primer korišćenja različitih sočiva



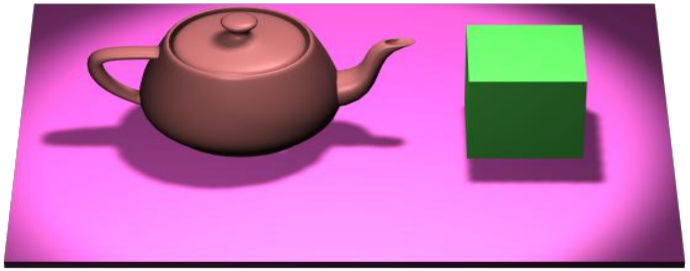
10mm Lens (fov = 122°)



20mm Lens (fov = 84°)



35mm Lens (fov = 54°)

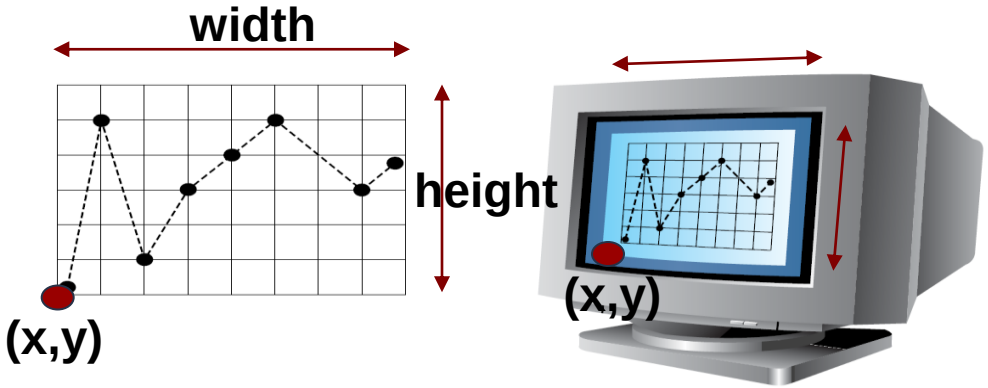


200mm Lens (fov = 10°)

Viewport transformacija

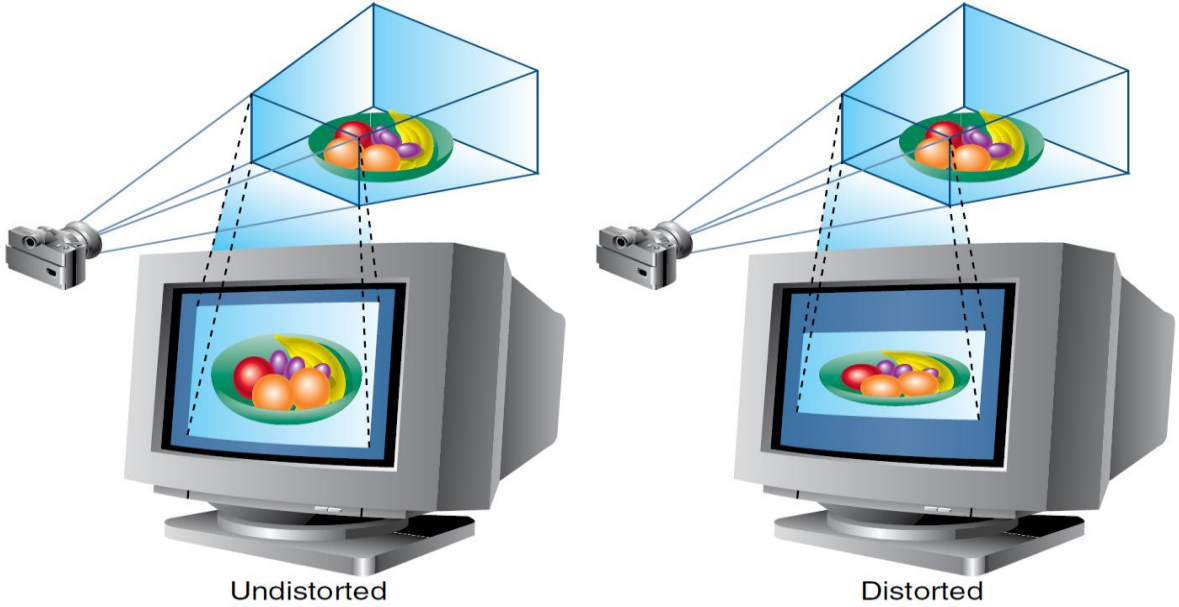
void glViewport(GLint x, GLint y, GLsizei width, GLsizei height)

Parametri x i y određuju donji levi ugao, a width i height širinu i visinu slike, respektivno. Svi parametri ove funkcije zadate su u pikselima.



Aspekt

Aspekt definiše odnos širine i visine slike. **Viewport aspekt** mora biti jednak **gluPerspective aspektu**, inače dolazi do distorzije.



Kompletan primer Reshape funkcije

```
void CGLRenderer::Reshape(CDC *pDC, int w, int h)
{
        wglMakeCurrent(pDC->m_hDC, m_hrc);
        //------------------------------------
        glViewport(0, 0, (GLsizei) w, (GLsizei) h);
        double aspect = (double)w / double)h;
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(45.0, aspect, 0.1, 100);
        glMatrixMode(GL_MODELVIEW);
        //------------------------------------
        wglMakeCurrent(NULL, NULL);
}
```

Uključivanje Z-bafera

- **glEnable**(**GL_DEPTH_TEST**);

Brisanje Z-bafera

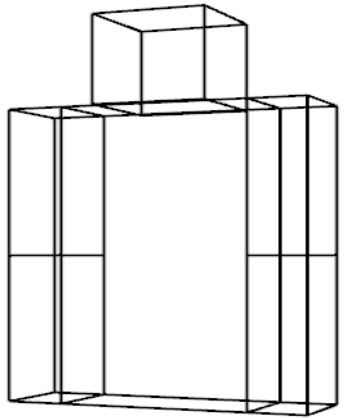
- **glClear**(GL_COLOR_BUFFER_BIT | **GL_DEPTH_BUFFER_BIT**);

Zadatak

Napisati funkciju **CGLRenderer::DrawRobot()**, koja crta robota. Za crtanje svih delova robota koristiti samo jedinične kocke (pozivom funkcije **DrawCube(1.0)**, koja je definisana u prethodnom poglavlju). Pojedini delovi tela dobijaju se primenom odgovarajućih transformacija (skaliranje/translacija/rotacija) na jedinične kocke. Robota nacrtati kao žičani model sa sledećim dimenzijama delova:

- trup – 1.0 x 2.0 x 0.5
- nadlaktice i podlaktice – 0.3 x 1.0 x 0.5
- glava – 0.6 x 0.6 x 0.6

Inicijalni izgled robota, sa međusobnim položajem svih delova, prikazan je na slici.



Zadatak - nastavak

U klasi **CGLView** dodati funkciju koja će reagovati na pritisak tastera na tastaturi, odnosno rukovati porukom WM_KEYDOWN. Ova funkcija treba da pozove funkciju **CGLRenderer::OnKeyDown(UINT nChar)** i prosledi joj pritisnuti taster.
Funkcija **CGLRenderer::OnKeyDown()** treba da izvrši sledeće operacije:
-ako je pritisnut taster Q da rotira telo robota ulevo,
-ako je pritisnut taster W da rotira telo robota udesno,
-ako je pritisnut taster E da rotira glavu robota ulevo,
-ako je pritisnut taster R da rotira glavu robota udesno,
-ako je pritisnut taster A da rotira levu ruku robota u ramenu naviše,
-ako je pritisnut taster Z da rotira levu ruku robota u ramenu naniže,
-ako je pritisnut taster S da rotira desnu ruku robota u ramenu naviše,
-ako je pritisnut taster X da rotira desnu ruku robota u ramenu naniže,
-ako je pritisnut taster D da rotira levu ruku robota u laktu naviše,
-ako je pritisnut taster C da rotira levu ruku robota u laktu naniže,
-ako je pritisnut taster F da rotira desnu ruku robota u laktu naviše,
-ako je pritisnut taster V da rotira desnu ruku robota u laktu naniže.
Korak rotacije neka bude 5^o. Pri definisanju rotacije nije neophodno voditi računa o ograničavanju uglova do kojih mogu da se rotiraju ruke robota.