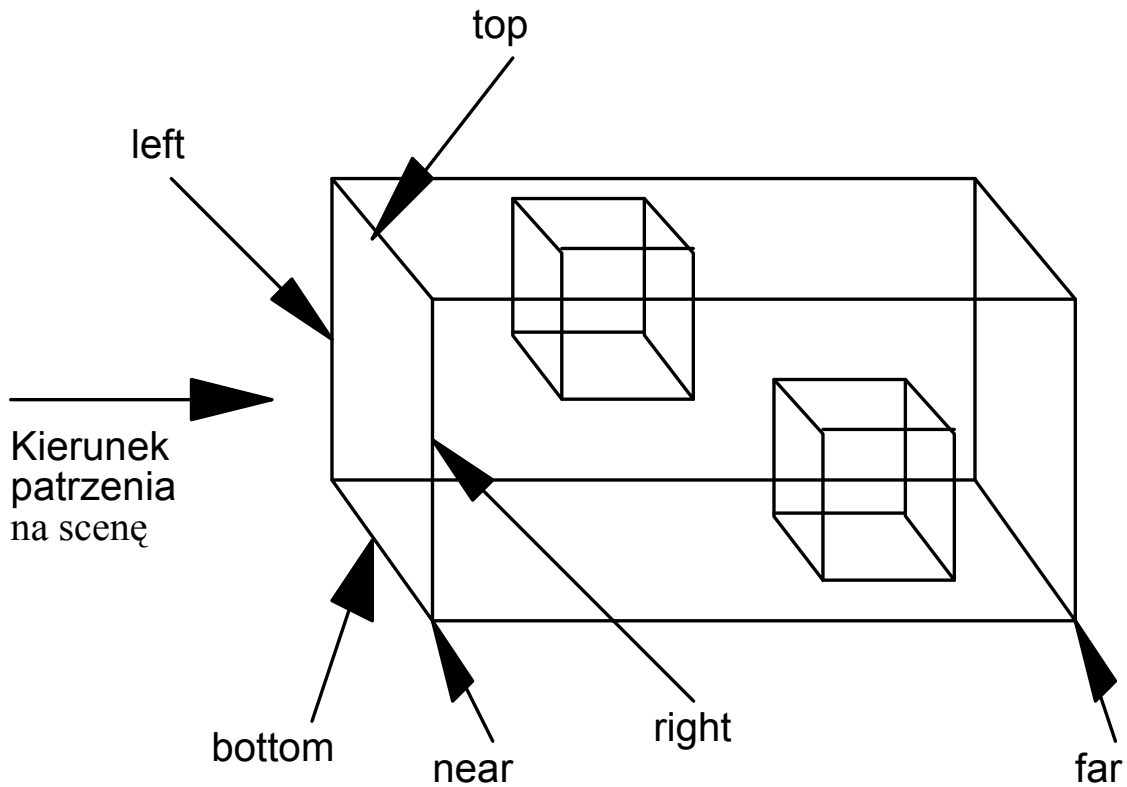
OpenGL – transformacje przestrzenne

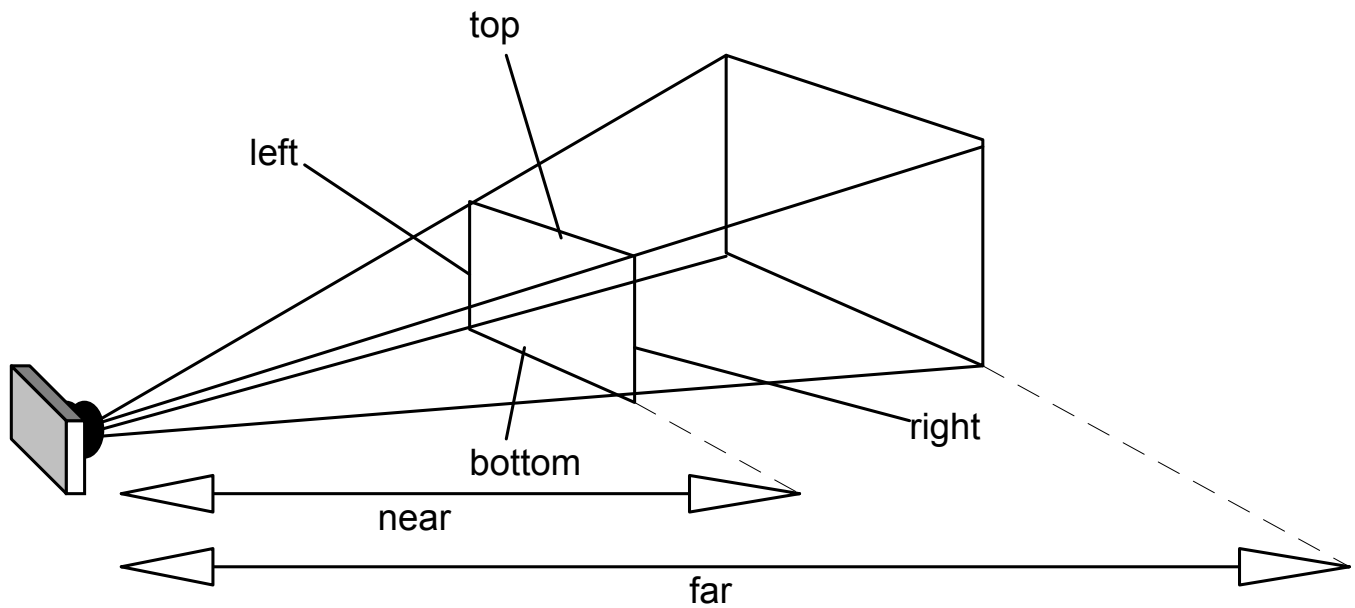
- Każdy zdefiniowany obiekt sceny, zanim pojawi się na ekranie monitora, poddawany jest trzem podstawowym transformacjom:
  - Obserwacji
  - Modelowania
  - Projekcji
- Projekcja określa fragment przestrzeni, który obserwowany jest przez kamerę, oraz sposób odzwierciedlania przestrzeni na ekranie.
- Typy odzwierciedlania przestrzeni w OpenGL:
  - Projekcja prostopadła
  - Projekcja perspektywiczna
- Dostępna jest również transformacja wycinająca określająca sposób przenoszenia sceny bezpośrednio na okno programu.

OpenGL – projekcja prostopadła



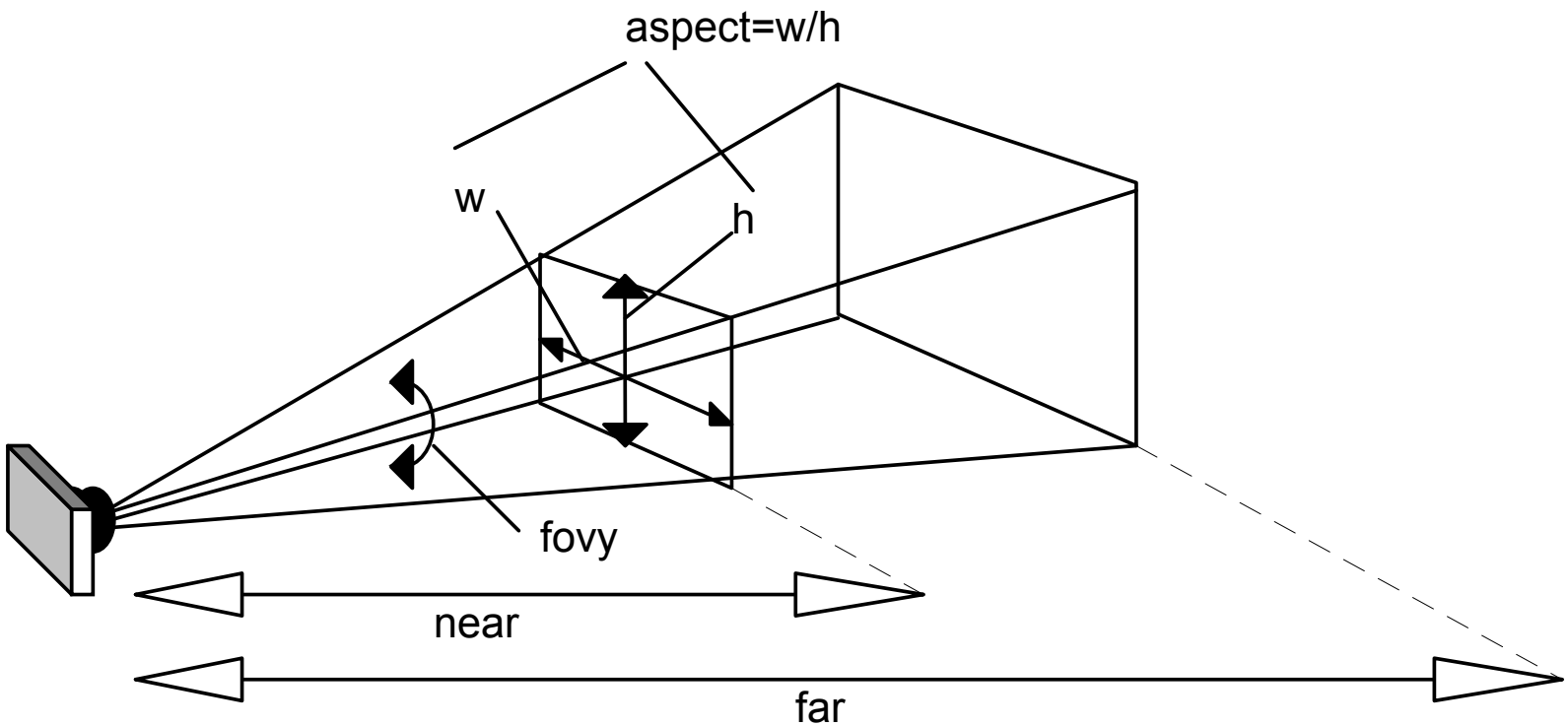

```
void glOrtho( GLdouble left, GLdouble right, GLdouble bottom
              GLdouble top, GLdouble near, GLdouble far );
```

OpenGL – projekcja perspektywiczna (1)



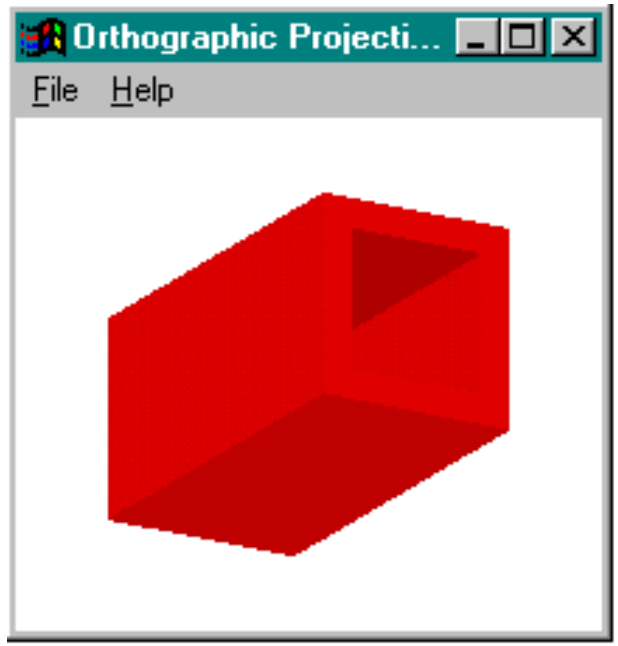

```
void glFrustum(GLdouble left, GLdouble right, GLdouble bottom,
               GLdouble top, GLdouble znear, GLdouble zfar );
```

OpenGL – projekcja perspektywiczna (2)




```
void gluPerspective( GLdouble fovy, GLdouble aspect,
                     GLdouble zNear, GLdouble zFar );
```

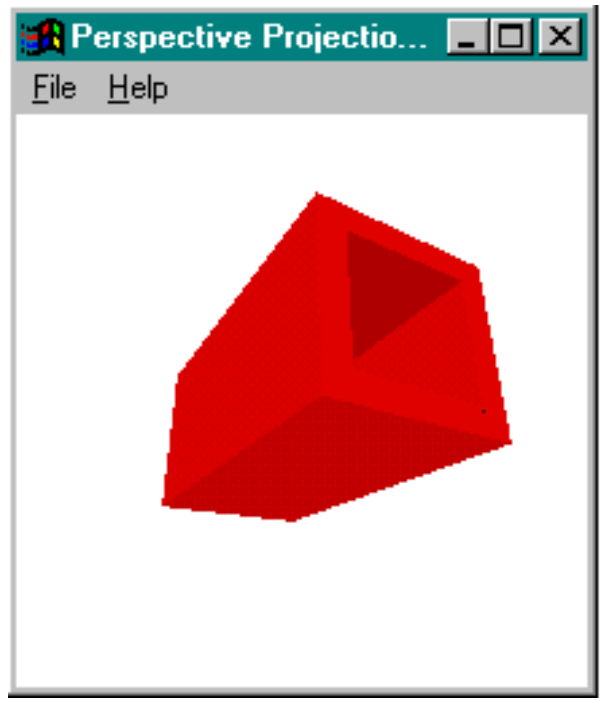
OpenGL – projekcja prostopadła - przykład




```
void ChangeSize(GLsizei w, GLsizei h)
{   GLfloat nRange = 120.0f;
    if(h = = 0) h = 1;

    glViewport(0, 0, w, h);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
       glOrtho (-nRange, nRange, -nRange*h/w,
                nRange*h/w, -nRange*2.0f, nRange*2.0f);
     else
       glOrtho (-nRange*w/h, nRange*w/h, -nRange,
                nRange, -nRange*2.0f, nRange*2.0f);

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

OpenGL – projekcja perspektywiczna - przykład

```
void ChangeSize(GLsizei w, GLsizei h)
    {
    GLfloat fAspect;
    if(h == 0)        h = 1;

    glViewport(0, 0, w, h);

    fAspect = (GLfloat)w/(GLfloat)h;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    gluPerspective(60.0f, fAspect, 1.0, 400.0);

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



OpenGL – modelowanie/obserwacja

- **Notacja jednorodna:**

$$P = [x, y, z, w],$$

$$\text{gdzie}: x_k = \frac{x}{w}, y_k = \frac{y}{w}, z_k = \frac{z}{w}$$

$$\text{zwykle}: P = [x, y, z, 1]$$

OpenGL – modelowanie/obserwacja

- **Macierz jednorodna:**

$$\left[\begin{array}{cccc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ \hline a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right]$$

- **Macierz przekształceń:**

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

- **Macierz translacji:**

$$\begin{bmatrix} a_{14} \\ a_{24} \\ a_{34} \end{bmatrix}$$

- **Macierz zniekształceń optycznych:**

$$\begin{bmatrix} a_{41} & a_{42} & a_{44} \end{bmatrix}$$

OpenGL – modelowanie/obserwacja

- **Zależność pomiędzy współrzędnymi punktów w różnych układach współrzędnych:**

zapis macierzowy:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & r_1 \\ a_{21} & a_{22} & a_{23} & r_2 \\ a_{31} & a_{32} & a_{33} & r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

uklad rownan liniowych:

$$x = a_{11}x' + a_{12}y' + a_{13}z' + r_1 \cdot 1$$

$$y = a_{21}x' + a_{22}y' + a_{23}z' + r_2 \cdot 1$$

$$z = a_{31}x' + a_{32}y' + a_{33}z' + r_3 \cdot 1$$

OpenGL – modelowanie/obserwacja

- **Obrót wokół osi z o kąt α**

$$\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Obrót wokół osi y o kąt α**

$$\begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Obrót wokół osi x o kąt α**

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Przesunięcie o wektor x_v, y_v, z_v**

$$\begin{bmatrix} 1 & 0 & 0 & x_v \\ 0 & 1 & 0 & y_v \\ 0 & 0 & 1 & z_v \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

OpenGL – modelowanie/obserwacja

- **Skalowanie:**

$$\begin{bmatrix} a & 0 & 0 & 1 \\ 0 & b & 0 & 1 \\ 0 & 0 & c & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Zmniejszenie obiektu:**

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & S \end{bmatrix}$$

OpenGL – modelowanie/obserwacja

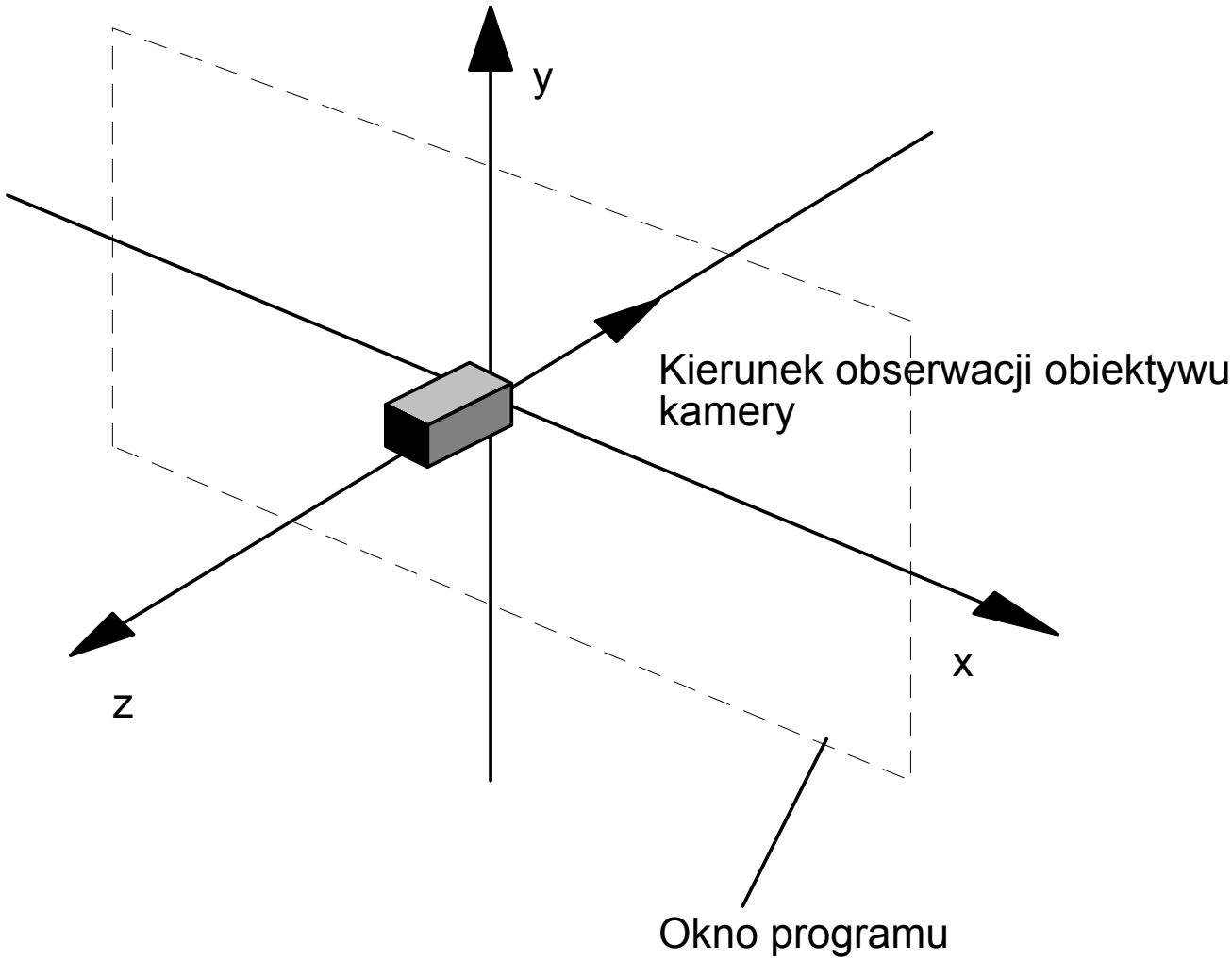
- **Składanie przekształceń w notacji jednorodnej polega na mnożeniu kolejnych macierzy przekształceń.**
- **Z uwagi na nieprzemienność mnożenia macierzy nieprzemienne jest składanie przekształceń.**
- **W wyniku mnożenia uzyskuje się zawsze macierz 4 x 4 odwzorowującą współrzędne danego obiektu na współrzędne po przekształceniu.**
- **Np. złożenie przekształceń Rot(x,30°) Trans(10,30,0) uzyskuje się przez pomnożenie macierzy:**

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(30°) & -\sin(30°) & 0 \\ 0 & \sin(30°) & \cos(30°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 30 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & \cos(30°) & -\sin(30°) & 10\cos(30°) \\ 0 & \sin(30°) & \cos(30°) & 30\sin(30°) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

OpenGL – modelowanie/obserwacja

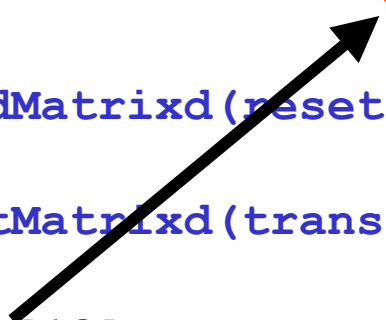
- **Początkowa orientacja i kierunek obserwacji sceny w OpenGL:**



OpenGL – modelowanie/obserwacja

- **Definiowanie własnych macierzy przekształceń w OpenGL:**

```
{   double x = 1.0;
    static double reset[]=
                    { 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
                    };

    static double trans1[]=
                    { 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

    glLoadMatrixd(reset); //glLoadIdentity();

    glMultMatrixd(trans1);

    trans1[12]+=x;
    glRectd(-20.0,-20.0,20.0,20.0); }
```



OpenGL – modelowanie/obserwacja

- **Podstawowe składnie przekształceń:**

```
{   double x = 1.0;  double a = 1.0;
    static double angle =0.0;
    static double trans1[]=        { 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
                                   };
    static double rot1[]=          { 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
                                   };
    glLoadIdentity();
    glMultMatrixd(trans1);
    glMultMatrixd(rot1);
    trans1[12]+=x;   angle+=a;
    rot1[0]=cos(angle*3.14/180); rot1[1]=-sin(angle*3.14/180);
    rot1[4]=sin(angle*3.14/180); rot1[5]=cos(angle*3.14/180);
    glRectd(-20.0,-20.0,20.0,20.0);
```

OpenGL – modelowanie/obserwacja

- **Predefiniowane funkcje OpenGL do transformacji układów współrzędnych:**

```
{
double b = 1.0;
double a = 1.0;
static double angle =0.0;
static double x =0.0;

glLoadIdentity();
glTranslated(x,0,0);
//void glTranslatef(GLfloat x, GLfloat y, GLfloat z);
glRotated(angle,0,0,-1);
//void glRotatef(GLfloat angle,GLfloat x,GLfloat y, GLfloat z);
angle+=a;
x+=b;
glRectd(-20.0,-20.0,20.0,20.0);}
```

- **Trzecia funkcja umożliwia zdefiniowanie skalowania:**

```
void glScalef(GLfloat z,GLfloat y,GLfloat z);
```

OpenGL – zaawansowane składanie przekształceń

- **Składnie kilku przekształceń:**

```
{
    static double rot1=0.0, rot2=0.0;
    glLoadIdentity();
    glRectd(-10.0,-10.0,10.0,10.0);

    glRotated(rot1,0,0,1);
    glTranslated(30,0,0);
    glRotated(rot2,0,0,1);

    glRectd(-10.0,-10.0,10.0,10.0);
    rot1+=1.0;
    rot2-=2.0;
}
```

OpenGL – zaawansowane składanie przekształceń

- **Funkcje glPushMatrix(); i glPopMatrix();**

```
{   static double rot1=0.0, rot2=0.0;
    glLoadIdentity();
    glRectd(-10.0,-10.0,10.0,10.0);

    glPushMatrix();
    glRotated(rot1,0,0,1);
    glTranslated(30,0,0);
    glRotated(rot2,0,0,1);
    glRectd(-10.0,-10.0,10.0,10.0);
    glPopMatrix();

    glPushMatrix();
    glRotated(-rot1,0,0,1);
    glTranslated(60,0,0);
    glRotated(-rot2,0,0,1);
    glRectd(-10.0,-10.0,10.0,10.0);
    glPopMatrix();

    rot1+=1.0;
    rot2-=2.0;
}
```

OpenGL – przykład – uproszczony model robota

```
// Ustalenie odwzorowania przestrzeni:
void ChangeSize(GLsizei w, GLsizei h)
{   GLfloat fAspect;
    if(h == 0)    h = 1;

    glViewport(0, 0, w, h);

    fAspect = (GLfloat)w/(GLfloat)h;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(65.0f, fAspect, 1.0, 50.0);

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

OpenGL – przykład – uproszczony model robota

```
// Obsługa klawiatury:
case WM_KEYDOWN:
        switch ((int)wParam)
        {
                case VK_ESCAPE:
                        DestroyWindow(hWnd);
                        break;
                case '1':
                        elbowAdd();
                        InvalidateRect(hWnd, NULL, FALSE);
                        break;
                case '2':
                        elbowSubtract();
                        InvalidateRect(hWnd, NULL, FALSE);
                        break;
                ...
                default:
                        break;
        }
break;
```

OpenGL – przykład – uproszczony model robota

```
// Rysowanie sceny:
void RenderScene(void)
    {   glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        //glTranslatef(0.0f,0.0f,-5.0f);
        //glRotated(20.0,1.0,0.0,0.0);

    gluLookAt(          0.0,1.0,5.0,
                        0.0,0.0,0.0,
                        0.0,1.0,0.0
                );
/*
    gluLookAt(          0.0,5.0,0.0,
                        0.0,0.0,0.0,
                        0.0,0.0,-1.0
                );

*/
```

OpenGL – przykład – uproszczony model robota

```
// Rysowanie sceny (cd):
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor3f (1.0, 0.0, 0.0);

    glPushMatrix();
        glTranslatef(2.0f,0.0f,1.0f);
        auxWireBox(1.0, 1.0, 1.0);
    glPopMatrix();

    glPushMatrix();
        glTranslatef (0.0, -0.6, 0.0);
        auxWireBox(1.0, 0.2, 1.0);

        glTranslatef (0.0, 0.6, 0.0);
        glRotatef ((GLfloat) base, 0.0, 1.0, 0.0);
        auxWireBox(0.4, 1.0, 0.4);

        glTranslatef (0.0, 0.5, 0.0);
        glRotatef ((GLfloat) elbow, 0.0, 0.0, 1.0);
        glTranslatef (0.0, 0.5, 0.0);
        auxWireBox(0.4, 1.0, 0.4);
```

OpenGL – przykład – uproszczony model robota

```
// Rysowanie sceny (cd):
        glTranslatef (0.0, 0.5, 0.0);
        glRotatef ((GLfloat) wrist, 0.0, 0.0, 1.0);
        glTranslatef (0.0, 0.5, 0.0);
        auxWireBox(0.4, 1.0, 0.4);

    glPopMatrix();

glFlush();

    }
```