2. Cvicenie

February 28, 2022

1. glTrans

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

const int icaskrok = 25; // v milisekundach

const float Lmax = 20.0; // rozmer sceny v smere X

float posunX = 0.0;  // zaciatocna hodnota posuvu (uvazujme len v smere X)

void aktualizuj(const int ihod)
{
        posunX += 0.05;
        glutPostRedisplay();
        glutTimerFunc(icaskrok, aktualizuj, ihod+1);
}

void obsluhaResize(int sirka, int vyska)
{
        glViewport(0, 0, sirka, vyska);
        glMatrixMode(GL_PROJECTION);  // dobre
        //glMatrixMode(GL_MODELVIEW); // zle (tento mod sa tu nema pouzivat)
        //-----------------------------------------------------------------
        // Ak posuvame CELY OBJEKT NARAZ pomocou  glTranslatef
        // (to je OVELA VHODNEJSI SPOSOB nez bod po bode),
        // tak  glLoadIdentity  potrebujeme aj tu a aj v kresliTrojuh2D.
        //
        // Tu (vdaka volbe GL_PROJECTION) nam  glLoadIdentity  resetuje maticu
        // robiacu prepocet suradnic VSETKYCH objektov na scene do intervalov
        // <-1, 1> (to je akoze ta PROJEKCIA, s tym, ze z-ove suradnice v tomto
        // programe ani nepouzivame a teda automaticky su hned od zaciatku
        // z intervalu <-1, 1>, konkretne asi nulove).
        //
        // Po resete treba pravdaze maticu naplnit potrebnymi cislami.
        // O to sa tu postara procedura  gluOrtho2D.
        //-----------------------------------------------------------------
        glLoadIdentity();
        if (sirka == 0) sirka++;
        const float pomstr = ((float)vyska)/sirka;
        gluOrtho2D(-0.5*Lmax, 0.5*Lmax, -0.5*Lmax*pomstr, 0.5*Lmax*pomstr);
```

```
}

void kresliTrojuh2D()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(0.0, 0.0, 1.0);

        glMatrixMode(GL_MODELVIEW);     // dobre, ale nie je nutne to tu volat
        //glMatrixMode(GL_PROJECTION);  // zle (tento mod sa tu nema pouzivat)
        //---------------------------------------------------------------------
        // Tu (vdaka volbe GL_MODELVIEW) nam  glLoadIdentity  resetuje maticu
        // pouzivanu na vypocet suradnic na trojrozmernej scene.
        // (Scena alebo „"svet v OpenGL je vzdy 3D, ale nemusime ten treti
        // rozmer nutne vyuzivat. Vtedy si OpenGL doplni z-ove suradnice samo.)
        // Tato matica a prislusny „matrix "mode teda suvisia s ROZMIESTNENIM
        // OBJEKTOV NA SCENE. Moze ist tak o staticke rozmiestnenie, ako aj
        // o pohyb. Tou maticou sa teda vyjadruju aj zmeny suradnic kvoli
        // pohybu.
        //
        // Po resete treba pravdaze maticu naplnit potrebnymi cislami.
        // O to sa tu postara procedura  glTranslatef.
        //---------------------------------------------------------------------
        glLoadIdentity();
        //---------------------------------------------------------------------
        // Pomocou glTranslatef() posuvame cely trojuholnok jedinym prikazom.
        // Je to tak efektivnejsie. Ale bolo vhodne najprv ukazat aj ten
        // elementarny neefektivny sposob.
        //---------------------------------------------------------------------
        glTranslatef(posunX, 0.0, 0.0);

        glBegin(GL_TRIANGLES);
                glVertex2f(-0.1*Lmax, -0.1*Lmax);
                glVertex2f( 0.1*Lmax, -0.1*Lmax);
                glVertex2f( 0.0*Lmax,  0.1*Lmax);

        glEnd();

        glutSwapBuffers();
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
//      glutInitDisplayMode(GLUT_RGBA);     // nie je nutne pisat
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(1080, 640);
        glutInitWindowPosition(200, 150);
        glutCreateWindow("OpenGL: trojuholnik");
        glutDisplayFunc(kresliTrojuh2D);
        glClearColor(0.8, 0.3, 0.3, 0.3);
        glutReshapeFunc(obsluhaResize);
        glutTimerFunc(icaskrok, aktualizuj, 0);
        glutMainLoop();
        return 0;
```

```
}
```

2. bez gluOrtho

```
#include <GL/gl.h>
#include <GL/glut.h>

const int icaskrok = 25; // v milisekundach

const float Lmax = 20.0; // rozmer sceny v smere X

float posunX = 0.0;  // zaciatocna hodnota posuvu (uvazujme len v smere X)

void aktualizuj(const int ihod)
{
        posunX += 0.05;
        glutPostRedisplay();
        glutTimerFunc(icaskrok, aktualizuj, ihod+1);
}

void obsluhaResize(int sirka, int vyska)
{
        glViewport(0, 0, sirka, vyska);
        glMatrixMode(GL_PROJECTION);
        //glLoadIdentity();
        if (sirka == 0) sirka++;
        const float pomstr = ((float)vyska)/sirka;
        //gluOrtho2D(-0.5*Lmax, 0.5*Lmax, -0.5*Lmax*pomstr, 0.5*Lmax*pomstr);

        //============================================================
        // Namiesto zakomentovanych riadov s  glLoadIdentity  a  gluOrtho2D
        // je ekvivalentne pouzit  glLoadMatrixf  so spravne pripravenou
        // maticou. Reset pomocou  glLoadIdentity  nie je potrebny, lebo
        // my tu maticu prikazom  glLoadMatrixf  rovno nastavime na taku,
        // aka je potrebna.
        // Ta matica len preskaluje suradnice sceny tak, aby boli
        // z intervalov <-1, 1>.
        //============================================================
        float mat[16];  // Maticu treba ulozit do 1D pola po stlpcoch,
                        // Tu je symetricka, takze aj po riadkoch by bolo OK.
        //------------------------------------------------------------
        // Naplnime skalovaciu maticu hodnotami.
        // Spravne po slovensky by sa asi mala nazyvat  MERITKOVA MATICA.
        //------------------------------------------------------------
        for (int ii = 0; ii < 16; ii++) mat[ii] = 0.0;
        mat[0] = 2.0/Lmax;           // skalovaci faktor pre x-ove suradnice
        mat[5] = 2.0/(Lmax*pomstr); //                   pre y-ove
        mat[10] = 1.0;               //                   pre z-ove
        mat[15] = 1.0;  // ta pridavna jednotka, pravy dolny prvok matice
        //--------------------------------
        // Posleme maticu OpenGL ,stroju'.
        //--------------------------------
        glLoadMatrixf(mat);
}
```

```
void kresliTrojuh2D()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(0.0, 0.0, 1.0);

        glMatrixMode(GL_MODELVIEW);     // dobre, ale nie je nutne to tu volat
        //glMatrixMode(GL_PROJECTION);  // zle
        //-------------------------------------------------------------------
        // Tu (vdaka volbe GL_MODELVIEW) nam  glLoadIdentity  resetuje maticu
        // robiacu ten PREPOCET SURADNIC objektov na scene, ktory je potrebny
        // LEN KVOLI POHYBU TYCH OBJEKTOV.
        // Tato matica teda nema nic spolocne s projekciou a vyjadruje len zmeny
        // suradnic objektov, pretoze tie objekty sa nejako hybu.
        //
        // Po resete treba pravdaze maticu naplnit potrebnymi cislami.
        // O to sa tu postara procedura  glTranslatef.
        //-------------------------------------------------------------------
        glLoadIdentity();
        //-------------------------------------------------------------------
        // Pomocou glTranslatef() posuvame cely trojuholnok jedinym prikazom.
        // Je to tak efektivnejsie. Ale bolo vhodne najprv ukazat aj ten
        // elementarny neefektivny sposob.
        //-------------------------------------------------------------------
        glTranslatef(posunX, 0.0, 0.0);

        glBegin(GL_TRIANGLES);
                glVertex2f(-0.1*Lmax, -0.1*Lmax);
                glVertex2f( 0.1*Lmax, -0.1*Lmax);
                glVertex2f( 0.0*Lmax,  0.1*Lmax);

        glEnd();

        glutSwapBuffers();
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
//      glutInitDisplayMode(GLUT_RGBA);    // nie je nutne pisat
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(1080, 640);
        glutInitWindowPosition(200, 150);
        glutCreateWindow("OpenGL: trojuholnik");
        glutDisplayFunc(kresliTrojuh2D);
        glClearColor(0.8, 0.3, 0.3, 0.3);
        glutReshapeFunc(obsluhaResize);
        glutTimerFunc(icaskrok, aktualizuj, 0);
        glutMainLoop();
        return 0;
}
```

3. priamo zadana transformacna matica

```
#include <GL/gl.h>
#include <GL/glut.h>

const int icaskrok = 25; // v milisekundach

const float Lmax = 20.0; // rozmer sceny v smere X

float posunX = 0.0;  // zaciatocna hodnota posuvu (uvazujme len v smere X)

void aktualizuj(const int ihod)
{
        posunX += 0.05;
        glutPostRedisplay();
        glutTimerFunc(icaskrok, aktualizuj, ihod+1);
}

void obsluhaResize(int sirka, int vyska)
{
        glViewport(0, 0, sirka, vyska);
        glMatrixMode(GL_PROJECTION);
        if (sirka == 0) sirka++;
        const float pomstr = ((float)vyska)/sirka;
        float mat[16];  // Maticu treba ulozit do 1D pola po stlpcoch,
                        // Tu je symetricka, takze aj po riadkoch by bolo OK.
        //-------------------------------------------------------------
        // Naplnime skalovaciu maticu hodnotami.
        // Spravne po slovensky by sa asi mala nazyvat  MERITKOVA MATICA.
        //-------------------------------------------------------------
        for (int ii = 0; ii < 16; ii++) mat[ii] = 0.0;
        mat[0] = 2.0/Lmax;           // skalovaci faktor pre x-ove suradnice
        mat[5] = 2.0/(Lmax*pomstr);  //                  pre y-ove
        mat[10] = 1.0;               //                  pre z-ove
        mat[15] = 1.0;  // ta pridavna jednotka, pravy dolny prvok matice
        //-------------------------------
        // Posleme maticu OpenGL ,stroju'.
        //-------------------------------
        glLoadMatrixf(mat);
}

void kresliTrojuh2D()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(0.0, 0.0, 1.0);

        glMatrixMode(GL_MODELVIEW);
        //glLoadIdentity();
        //glTranslatef(posunX, 0.0, 0.0);
        float mat[16];  // Maticu treba ulozit do 1D pola po stlpcoch,
                        // Tu je symetricka, takze aj po riadkoch by bolo OK.
        //-------------------------------------
        // Naplnime translacnu maticu hodnotami.
        //-------------------------------------
        for (int ii = 0; ii < 16; ii++) mat[ii] = 0.0;
        mat[0]  = 1.0; // prva jednicka na diagonale
```

```
        mat[5]  = 1.0; // druha
        mat[10] = 1.0; // tretia
        mat[15] = 1.0; // stvrta
        mat[12] = posunX;  // tx, prvok s INDEXAMI [1][4]
        //--------------------------------
        // Posleme maticu OpenGL ,stroju'.
        //--------------------------------
        glLoadMatrixf(mat);

        glBegin(GL_TRIANGLES);
                glVertex2f(-0.1*Lmax, -0.1*Lmax);
                glVertex2f( 0.1*Lmax, -0.1*Lmax);
                glVertex2f( 0.0*Lmax,  0.1*Lmax);

        glEnd();

        glutSwapBuffers();
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
//      glutInitDisplayMode(GLUT_RGBA);    // nie je nutne pisat
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(1080, 640);
        glutInitWindowPosition(200, 150);
        glutCreateWindow("OpenGL: trojuholnik");
        glutDisplayFunc(kresliTrojuh2D);
        glClearColor(0.8, 0.3, 0.3, 0.3);
        glutReshapeFunc(obsluhaResize);
        glutTimerFunc(icaskrok, aktualizuj, 0);
        glutMainLoop();
        return 0;
}
```

4. ronovbezky v Ortho

```
#include <GL/gl.h>
#include <GL/glu.h> // tu potrebne len kvoli gluOrtho2D
#include <GL/glut.h>

const float Lmax = 40.0;

void obsluhaResize(int sirka, int vyska)
{
        glViewport(0, 0, sirka, vyska);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if (sirka == 0) sirka++;
        const float pomstr = ((float)vyska)/sirka;
        //gluOrtho2D(-0.5*Lmax, 0.5*Lmax, -0.5*Lmax*pomstr, 0.5*Lmax*pomstr);
        glOrtho(-0.5*Lmax, 0.5*Lmax, -0.5*Lmax*pomstr, 0.5*Lmax*pomstr, -1, +1);
        //-------------------------------------------------------------------
        // Pouzitie vseobecnejsej procedury  glOrtho  tak, ako je napisane
```

```
        // vyssie, by malo identicky efekt ako  gluOrtho2D.
        // Vysek priestoru zobrazovany pomocou gluOrtho2D ma teda napevno
        // nastaveny rozsah z-ovych suradnic <-1, 1>.
        // Pritom os Z smeruje za rovinu nakresne, lebo mame pravotocivu
        // suradnicovu sustavu a osi X, Y  tak, ako sme zvyknuti.
        // Pri kolmej projekcii sa scena kolmo premieta na nejaku myslenu
        // plochu (nakresnu). Fiktivny pozorovatel alebo kamera pozoruje uz len
        // ten premietnuty dvojrozmerny obraz.
        //
        // Nakresna je umiestnena v rovine z=0.
        // Ak si zvolime zobrazovany vysek priestoru tak, ze ho nakresna
        // pretina (ze sa v nom nachadza), tak si musime predstavit, ze
        // to kolme premietanie sa robi z oboch stran nakresne.
        // Tak to je pri gluOrtho2D, kedze ma z-ovy rozsah sceny <-1,1>.
        //                 Ale pozor!!!
        // POSLEDNE DVA PARAMETRE v glOrtho nie su suradnice pravotocivej
        // sustavy; su nazvane (zovseobecnenymi) vzdialenostami od premietacej
        // plochy. Zovseobecnenymi preto, lebo mozu byt aj zaporne; ak je
        // nieco za premietacou plochou, ma to od nej kladnu vzdialenost.
        // Vsetko pred nou (blizsie k divakovi) ma zapornu akoze vzdialenost.
        // Tie posledne dva parametre FAKTICKY SU Z-OVE SURADNICE z hladiska
        // LAVOTOCIVEJ SURADNICOVEJ SUSTAVY, teda akoby os z smerovala ku
        // divakovi. Ona vsak smeruje od neho prec, lebo OpenGL POUZIVA
        // PRAVOTOCIVU SURADNICOVU SUSTAVU. Len procedura glOrtho je spravena
        // tak, ze to vlastne zodpoveda lavotocivej sustave, len to v dokumen-
        // tacii nechcu priznat, tak musia tie posledne dva parametre nazyvat
        // vzdialenosti.
        // Strucne:
        // Zobrazovany vysek priestoru moze vo vseobecnosti mat z-ovu
        // suradnicu z lubovolneho rozsahu <zn, zf>.
        //      zn = near-clip plane
        //      zf = near-clip plane
        // a musi byt  zn > zf,  lebo os z (pravotocivej sustavy) ide za rovinu
        // nakresne. Posledne dva parametre v glOrtho su
        //      dn = -zn
        //      df = -zf
        // gluOrtho2D napevno predpoklada zn = +1, zf = -1, teda nastavuje
        // dn = -1, zn = +1.
        // Pritom typicke, a pri perspektivnej projekcii asi povinne, byva, ze
        // obe tie roviny, zn aj zf, su za rovinou nakresne, cize obe sa zvyknu
        // nastavovat zaporne. Procedura gluOrtho2D je vsak spravena netypicky:
        // ma zn rovinu pred nakresnou, cize  zn = +1 (kladna hodnota, co je
        // v OpenGL grafike dost netypicke umiestnenie, takmer by sme povedali,
        // ze za chrbtom pozorovatela), ale pri kolmej projekcii na tom
        // nezalezi. Ale dolezite potom je, aby sa na nakresnu premietala
        // aj ta cast sceny, ktora je aj pred nou, teda blizsie ku divakovi.
        // To sa aj robi, ako si mozeme overit. Premieta sa vzdy ta cast
        // sceny, ktora ma hlbkovu (z-ovu) suradnicu z rozsahu <zn, zf>.
        //--------------------------------------------------------------------
}

void kresliRovnobezky3D()
{
        glClear(GL_COLOR_BUFFER_BIT);
```

```
//      glClear(GL_DEPTH_BUFFER_BIT);   // nie je nutne nastavit
        glMatrixMode(GL_MODELVIEW);
        //glLoadIdentity(); // Netreba, lebo umiestenenie definujeme
                            // pomocou  glVertex3f.
        glColor3f(0.0, 0.0, 0.0);
        //-------------------------------------------------------------------
        // Pomocou vyberu z nasledujucich piatich rozne dlhych ciar v smere
        // suradnice  z  sa mozeme presvedcit, ze pri kolmej projekcii pomocou
        // gluOrtho2D sa zobrazi len cast priestoru  -1  <= z <= 1 .
        // Tak sa aj dlha palicka javi opticky kratkou.
        //
        // A ak je palicka v smere  z  KRATSIA nez tento interval, tak
        // ... odskusajte si sami a popremyslajte o tom. Ku takemu pripadu som
        // sa tu najprv rozpisal, ale potom to sem nedal, lebo som si to
        // nestihol dostatocne odskusat. Neskor sa ku tomuto este vratim.
        // Asi bude vhodne nakrelit ku tomu nejaky jednoduchy obrazok.
        //-------------------------------------------------------------------
        glBegin(GL_LINES);
                glVertex3f(-0.1*Lmax, 0.2*Lmax,  0.0);
                glVertex3f(-0.1*Lmax, 0.3*Lmax, -1.0);
                glVertex3f(+0.1*Lmax, 0.2*Lmax,  0.0);
                glVertex3f(+0.1*Lmax, 0.3*Lmax, -4.0);
                //glVertex3f(+0.1*Lmax, 0.3*Lmax, -0.01);
        glEnd();
        glutSwapBuffers();
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
//      glutInitDisplayMode(GLUT_RGBA);    // nie je nutne pisat
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(640, 640);
        glutInitWindowPosition(200, 150);
        glutCreateWindow("OpenGL: rovnobezky");
        glutDisplayFunc(kresliRovnobezky3D);
        glutReshapeFunc(obsluhaResize);
        glClearColor(1.0, 1.0, 1.0, 0); // farba pozadia biela
        glutMainLoop();
        return 0;
}
```

4. ronovbezky v Perspective

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

const float Lmax = 40.0;

const double zuY = 85.0; // zorny uhol pozdlz Y
const double dnear = 10.0;
const double dfar  = 20.0;
```

```
void obsluhaResize(int sirka, int vyska)
{
        glViewport(0, 0, sirka, vyska);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if (vyska == 0) vyska++;
        const float sir_ku_vys = ((float)sirka)/vyska;
        //-----------------------------------------------------------------
        // gluPerspective(fovY, sir_ku_vys, dnear, dfar);
        //
        // fovY  =  field of view along Y  =  sirka uhla pohladu pozdlz Y
        //                                    (zorny uhol pozdlz Y)
        // dnear = vzdialenost pozorovatela od blizsej orezavacej roviny
        // dfar  = vzdialenost pozorovatela od vzdialenejsej orezavacej roviny
        // Musi teda byt  0 < dnear < dfar.
        // Pozorovatel vidi len svet zo ZAPORNYMI Z-ovymi suradnicami.
        // Preto  zn = -dnear,  zf = -dfar  (suradnice tych rovin).
        //
        // gluPerspective nasobi doterajsiu transformacnu maticu, cize
        // transformacna matica sa zmeni. Aby sme tym nedostali nejaku
        // neziaducu transformaciu, ale len perspektivnu projekciu, bolo vyssie
        // treba spravit reset pomocou glLoadIdentity.
        //
        // Nasobenim matice a vektora sa vsak neda spravit pozadovana
        // transformacia (zobrazenie, ,mapping') uplne. (Neda sa spravit delenie
        // hodnotou  Z.) Ale OpenGL stroj sa kdesi pre nas neviditelne postara
        // o to, aby za to delenie spravilo.
        //-----------------------------------------------------------------
        gluPerspective(zuY, sir_ku_vys, dnear, dfar);
}

void kresliRovnobezky3D()
{
        glClear(GL_COLOR_BUFFER_BIT);
//      glClear(GL_DEPTH_BUFFER_BIT);   // nie je nutne nastavit
        glMatrixMode(GL_MODELVIEW);
        glColor3f(0.0, 0.0, 0.0);
        //-----------------------------------------------------------------
        // Ked stojime na kolajniciach a pozerame ponad ne do dialky, tak sa
        // zda, ze sa zbiehaju. Moze za to perspektivna projekcia.
        // Ciary vytvorene v tomto programe su presne take: su rovnobezne,
        // ale zda sa, ze sa zbiehaju.
        //
        // Ak by sme tie ciary pre zmenu nakreslili vyssie (y = +0.2*Lmax),
        // tak by sa zasa zdalo, ze su to trolejove vedenia nad nami
        // a ze sa tiez zbiehaju (poznamenal Peter Bokes).
        //-----------------------------------------------------------------
        glBegin(GL_LINES);
                glVertex3f(-0.1*Lmax, -0.2*Lmax, -10.0);
                glVertex3f(-0.1*Lmax, -0.2*Lmax, -12.0);
                glVertex3f(+0.1*Lmax, -0.2*Lmax, -10.0);
                glVertex3f(+0.1*Lmax, -0.2*Lmax, -12.0);
        glEnd();
```

```
        glutSwapBuffers();
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
//      glutInitDisplayMode(GLUT_RGBA);    // nie je nutne pisat
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(640, 640);
        glutInitWindowPosition(200, 150);
        glutCreateWindow("OpenGL: rovnobezky");
        glutDisplayFunc(kresliRovnobezky3D);
        glutReshapeFunc(obsluhaResize);
        glClearColor(1.0, 1.0, 1.0, 0); // farba pozadia biela
        glutMainLoop();
        return 0;
}
```