Dokument pre vyučujúcich ku 1. cvičeniu z predmetu *Fyzikálne základy počítačových hier*

ÚJFI, FEI STU v Bratislave

posledná aktualizácia: 14. februára 2022

Praktické informácie

- učebňa: **DigLab Samsung**, miestnosť -1.42 (mínus prvé podlažie)
  https://www.fiit.stuba.sk/vyskum/laboratoria/digilab-samsung.html?page_id=4249
- kontaktná osoba: Dominik Gozora (`dominik.gozora@stuba.sk`), tel. 02/210 22 244, kancelária 2.44
- administrátor softvéru: Peter Grell (`peter.grell@stuba.sk`), tá istá kancelária
- Kompilátor, hlavičkové súbory (`.h`) a knižnice (`.dll`) sú niekde pod `C:`
  `mingw`.
- kompilovať a linkovať v príkazovom riadku príkazom ako napr.
  `gcc -Wall mojprogram.c -lopengl32 -lglu32 -lfreeglut`
  alebo (aj so zadaním názvu výsledného exe súboru, napr. `i.x`)
  `gcc -Wall mojprogram.c -lopengl32 -lglu32 -lfreeglut -o i.x`
  (A knižnica `glu32` nie je v skorších programíkoch potrebná.) Ak by to nešlo, prípadne skúsiť iné poradie knižníc a hlavne dať pozor, či kompilátor a linker `gcc` vôbec pozná cesty ku hlavičkovým súborom a knižniciam.
- Ak by skompilovaný program nešlo spustiť, príčinou môže byť, že nie je nastavená cesta ku dynamickým knižniciam (tým s príponami `.dll`). Tá cesta sa nastavuje do premennej `LD_LIBRARY_PATH`; taký názov má na Linuxe a možno aj na Windows, ale nie som si úplne istý (a dúfam, že to ani nebude treba vedieť). Nepopliesť s premennou `LIBRARY_PATH`. Tá sa používa pri linkovaní, zatiaľ čo `LD_LIBRARY_PATH` pri spustení programu. Ak sa nejaká premenná nastaví v konfiguračnom súbore, treba potom otvoriť nový príkazový riadok, aby sa nové nastavenie naozaj prejavilo (a skúsiť kompilovanie a spustenie v ňom). S detailami pomôže správca softvéru, ale dúfam, že to nebude potrebné a všetko bude fungovať.
- Literatúra ku cvičeniam (len narýchlo spísaná):
  - John Kessenich, Graham Sellers, Dave Shreiner: *The OpenGL Programming Guide*,
    (ak by teda niečo chceli o tom čítať; niekde som to videl aj v elektronickej podobe, asi html).
  - Július Cirák a kol.: *Zbierka príkladov a úloh z fyziky* – dostanú z toho nejakú časť ako PDF, ale najprv musím vypýtať súhlas.
  - naše príklady na `kf.elf.stuba.sk`

1 okno_prazdne_okamih.c

```
#include <unistd.h>     // len kvoli funkcii sleep(); inak to vykomentujte
                        // alebo vymazte
#include <GL/glut.h>

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutCreateWindow("OpenGL: Ahoj, ja som okno!");
        sleep(1);  // cas necinnosti v sekundach
        return 0;
}
```

2 okno_prazdne_trvalo.c

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

void nasa_procedura() {}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutCreateWindow("OpenGL: Ahoj, ja som okno!");
        //--------------------------------------------------------------
        // Cielom tohto programiku nie je kreslit nejaky obrazok, ale kedze
        // kompilator na Windows sa stazuje, ze nejaka „"callback procedura
        // na kreslenie mu chyba (ze nie je registrovana), tak mu dame aspon
        // nejaku prazdnu callback proceduru.
        //--------------------------------------------------------------
        glutDisplayFunc(nasa_procedura);
        glutMainLoop();
        return 0;
}
```

3 okno_cierne.c

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

// Je to funkcia typu void, cize nevracia ziadnu hodnotu,
// len cosi robi. Je to teda vlastne procedura.
void nasa_procedura() {
        glClear(GL_COLOR_BUFFER_BIT);
        glutSwapBuffers();
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutCreateWindow("OpenGL: okno v systeme");
        glutDisplayFunc(nasa_procedura);
        glutMainLoop();
```

```
        return 0;
}
```

4 okno_zelene.c

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

void nasa_procedura() {
        //-------------------------------------------------------------
        // glClear nastavi farbu okna bud na default (cierna) alebo na nami
        // definovanu (prikazom glClearColor v main).
        // Treba teda najprv zavolat glClearColor a az potom glClear.
        // Inak by prvy frame bol cierny (co by sme si vsak ani nemuseli
        // stihnut vsimnut.
        //
        // Ak glClear umiestnime sem (a nie do main), tak nastavena farba
        // automaticky vyplni cele okno aj pri jeho zvacseni.
        //-------------------------------------------------------------
        glClear(GL_COLOR_BUFFER_BIT);
        glutSwapBuffers();
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE); // na Windows MOZNO treba,
                                          // aby farba bola zelena hned
        glutCreateWindow("OpenGL: okno v systeme");
        glutDisplayFunc(nasa_procedura);
        glClearColor(0.0, 1.0, 0.0, 0.0); // definuje farbu vyplne okna - zelena
        //-------------------------------------------------------------
        // Ak by sme v programe mali glClear len tu, teda v main, tak pri
        // zvacseni okna by zelena farba vyplnala iba povodne definovanu cast
        // okna (povodny defaultovy viewport).
        //-------------------------------------------------------------
        //glClear(GL_COLOR_BUFFER_BIT);  // nevhodne umiestnenie glClear
        glutMainLoop();
        return 0;
}
```

5 okno_geometria.c

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

void nasa_procedura() {
        glClear(GL_COLOR_BUFFER_BIT);
        glutSwapBuffers();
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
```

```
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(640, 480);
        glutInitWindowPosition(200, 150);
        glutCreateWindow("OpenGL: okno v systeme");
        glutDisplayFunc(nasa_procedura);
        glClearColor(0.0, 0.0, 1.0, 0.0);
        glutMainLoop();
        return 0;
}
```

6 trojuh2D_nehyb.c

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

void kresliTrojuh2D()
{
        glClear(GL_COLOR_BUFFER_BIT); // nastavi definovanu farbu pozadia okna
        glColor3f(0.0, 0.0, 1.0);     // definuje a nastavi farbu trojuholnika

        //-------------------------------------------------------------------
        // Pokial to nezariadime inak (a tu sme to naozaj nezariadili), tak na
        // OpenGL scene sa budu dat zobrazit len suradnice z rozsahov  <-1, 1>.
        //-------------------------------------------------------------------
        glBegin(GL_TRIANGLES);
                glVertex2f(-0.8, -0.8);
                glVertex2f( 0.8, -0.8);
                glVertex2f( 0,    0.8);

                // Tento vacsi by sa nezmestil na scenu:
                // glVertex2f(-2.0, -2.0);
                // glVertex2f( 2.0, -2.0);
                // glVertex2f( 0,    2.0);
        glEnd();

        glutSwapBuffers();  // Vykresli pripravenu scenu.
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
//      glutInitDisplayMode(GLUT_RGBA);    // nie je nutne pisat
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(640, 640);  // zvolme nateraz stvorcove
        glutInitWindowPosition(200, 150);
        glutCreateWindow("OpenGL: trojuholnik");
        glutDisplayFunc(kresliTrojuh2D);
        glClearColor(0.8, 0.3, 0.3, 0.3);  // definuje farbu vyplne okna
                                           // (teda farbu pozadia)
        glutMainLoop();
        return 0;
}
```

7 trojuh2D_divne.c

```
#include <GL/gl.h>
#include <GL/glu.h>  // GL Utilities; toto potrebujeme az teraz.
#include <GL/glut.h>

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

        // Nastavme nasej 2-rozmernej OpenGL scene suradnice z intervalov
        // -2 < x < 2,   -1 < y < 1 .
        gluOrtho2D(-2.0, 2.0, -1.0, 1.0);  // funkcia z GL utilities
        // Dala by sa nahradit zakladnou OpenGL funkciou
        // glOrtho(-2.0, 2.0, -1.0, 1.0, -1.0, 1.0);

        glBegin(GL_TRIANGLES);
                glVertex2f(-0.8, -0.8);
                glVertex2f( 0.8, -0.8);
                glVertex2f( 0,    0.8);
        glEnd();

        glutSwapBuffers();  // Vykresli pripravenu scenu.
}

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

8 trojuh2D_nehyb_resizeOK.c

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

// Ako elegantne tie nazvy hlavickovych suborov na seba nadvazuju! :-)

//#include <stdio.h>  // odkomentovat pri pouziti printf


const float Lmax = 4.0; // rozmer sceny v smere X

void obsluhaResize(int sirka, int vyska)
```

```
{
        //-------------------------------------------------------------
        // Nazorna predstava je, ze OpenGL „"ceruzka kresli na sklo
        // (to je ten Viewport) umiestnene tesne za nepriehladnou doskou.
        // V doske je vyrezane okno so sirkou a vyskou v pixeloch.
        // Ak chceme, aby bolo „"sklo (a na nom kresleny obraz) vidiet,
        // treba sklo (viewport) umiestnit presne tam, kde je okno v doske
        // (inak by sklo, alebo jeho cast, bolo doskou zaclonene).
        // Zvycajne najvhodnejsie je, ked ma to sklo presne taku velkost,
        // aku ma okno v doske.
        // 0, 0, sirka, vyska  su v pixeloch.
        // Tie  0, 0  su posun „"skla voci oknu v doske.
        //
        // Farba vyplne okna je, ak sa nemylim, viazana na okno,
        // nie na ,sklo'.
        //-------------------------------------------------------------
        glViewport(0, 0, sirka, vyska);
        //printf("sirka = %d\n", sirka);

        //-------------------------------------------------------------
        // glMatrixMode s volbou GL_PROJECTION nastavi, ze operacie
        // glLoadIdentity a gluOrtho2D robene nizsie sa budu tykat
        // (nam neviditelnej) matice zabezpecujucej projekciu sceny
        // (prepocet suradnic vsetkych objektov na scene do zobrazitelnych
        // rozsahov, t. j. do rozsahov <-1, 1>).
        //-------------------------------------------------------------
        glMatrixMode(GL_PROJECTION);

        glLoadIdentity(); // Prave TATO FUNKCIA zabezpeci, ze sa velkost troj-
                          // uholnika nebude pri prekresleni sceny scvrkavat.
        //-------------------------------------------------------------------
        // Pomocou gluOrtho2D() definujeme, KTORA CAST PRIESTORU „("fyzickeho
        // sveta, tu vlastne len „"fyzickej plochy) SA BUDE ZOBRAZOVAT (kolmou
        // projekciou). Ak chceme, aby mal obraz prirodzeny pomer vysky ku
        // sirke, musime pomer stran zobrazovanej plochy nastavit zhodny
        // s pomerom stran „"skla.
        // Funkcii gluOrtho2D() treba rozumiet tak, ze definuje ten vysek
        // z „"fyzickeho sveta, ktory chceme pomocou OpenGL zobrazit na
        // viewport (t. j. na „"sklo) a nasledne aj do okna.
        //
        // gluOrtho2D robi svoju pracu tak, ze preskaluje suradnice sceny z nami
        // definovanych rozsahov do rozsahov <-1, 1>. Robi to pomocou maticoveho
        // nasobenia. gluOrtho2D by sa teda dala nahradit priamym vytvorenim
        // vhodnej matice napr. pomocou glLoadMatrixf.
        //-------------------------------------------------------------------
        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);
```

```
        // V tejto procedure sa teda gluOrtho2D() NEMA POUZIVAT.
        // Ma byt v obsluhaResize(), kam sme to presunuli.

        glBegin(GL_TRIANGLES);
                glVertex2f(-0.8, -0.8);
                glVertex2f( 0.8, -0.8);
                glVertex2f( 0,    0.8);

                //------------------------------------------------------------
                // Aj takyto vacsi by sa tentoraz uz zmestil na scenu, aspon
                // ak prilis neznizime vysku okna (a "skla).
                // Vhodnejsie by bolo vyjadrit suradnice tych vrcholov
                // ako nasobky Lmax.
                //------------------------------------------------------------
                // glVertex2f(-1.3, -1.3);
                // glVertex2f( 1.3, -1.3);
                // glVertex2f( 0,    1.0);
        glEnd();

        glutSwapBuffers();
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
//      glutInitDisplayMode(GLUT_RGBA);    // nie je nutne pisat
        glutInitDisplayMode(GLUT_DOUBLE);
        //--------------------------------------------------------------------
        // Prikazom  glutInitWindowSize()  akoby vytvorime v nejakej
        // NEPRIEHLADNEJ nekonecne rozlahlej doske obdlznikovy otvor - okno,
        // cez ktore budeme vidiet na svet za doskou. Na obrazovke budeme
        // vidiet len to, co nam umozni ten "vypileny otvor.
        // Vyssie je tato predstava rozvita dalej.
        //--------------------------------------------------------------------
        glutInitWindowSize(640, 640);  // zvolme nateraz stvorcove okno
        glutInitWindowPosition(200, 150);
        glutCreateWindow("OpenGL: trojuholnik");
        glutDisplayFunc(kresliTrojuh2D);
        glClearColor(0.8, 0.3, 0.3, 0.3);
        // Zavedieme "callback funkciu obsluhaResize().
        glutReshapeFunc(obsluhaResize);
        glutMainLoop();
        return 0;
}
```

9 trojuh2D_posuvany_elem.c

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdio.h>  // odkomentovat, ak chceme pouzivat printf()

//============================================================
```

```
// icaskrok je v milisekundach. Kazdych 25 ms sa teda bude volat toto:
//      kresliTrojuh2D()
//      aktualizuj()
// Je to potrebne, lebo objekty na scene sa mohli pohnut a my chceme
// tieto zmeny co najplynulejsie zobrazovat.
//=============================================================
const int icaskrok = 25;

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)
{
        printf(" aktualizuj(): ihod = %d\n", ihod);
        posunX += 0.05;
        glutPostRedisplay();  // Tymto podavame ziadost o prekreslenie sceny.

        // Je v zasade jedno, co je poslednym argumentom nasledujucej funkcie.
        glutTimerFunc(icaskrok, aktualizuj, ihod+1);
}

void obsluhaResize(int sirka, int vyska)
{
        printf(" obsluhaResize(): sirka = %d px,  vyska = %d px\n",sirka,vyska);
        glViewport(0, 0, sirka, vyska);
        glMatrixMode(GL_PROJECTION);
        //-------------------------------------------------------------
        // Ak posuvame bod po bode (menej vhodny sposob),
        // tak  glLoadIdentity()  musi byt tu, a nie v kresliTrojuh2D().
        //-------------------------------------------------------------
        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()
{
        printf(" kresliTrojuh2D():  Pripravujem kresbu.\n");
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(0.0, 0.0, 1.0);
        glMatrixMode(GL_MODELVIEW); // nie je nutne to volat,
                                    // lebo GL_MODELVIEW je default.
        glBegin(GL_TRIANGLES);
                glVertex2f(-0.8+posunX, -0.8);
                glVertex2f( 0.8+posunX, -0.8);
                glVertex2f( 0.0+posunX,  0.8);
        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); // "callback" procedura na kreslenie
        glClearColor(0.8, 0.3, 0.3, 0.3); // definuje farbu vyplne okna
        glutReshapeFunc(obsluhaResize);
        // aktualizuj() je "callback" funkcia kvoli zmenam v case.
        // 0 je nami zvolena zaciatocna hodnota ihod.
        glutTimerFunc(icaskrok, aktualizuj, 0);
        glutMainLoop();
        return 0;
}
```

10 trojuh2D_posuvany_glTrans.c

```
#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;
}
```

11 obdlzniky_2D_posuv.c

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

const int icaskrok = 25; // v milisekundach

const float Lmax = 50.0; // rozmer sceny v smere X

float posunX = 0.0;  // zaciatocna hodnota premennej pre posuvanie

void aktualizuj(const int ihod)
{
        posunX += 0.05;
        glutPostRedisplay();
        glutTimerFunc(icaskrok, aktualizuj, ihod-7);
}

void obsluhaResize(int sirka, int vyska)
{
        glViewport(0, 0, sirka, vyska);
        glMatrixMode(GL_PROJECTION);    // tu nutne s parametrom 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);
}

void kresliObdlzniky2D()
{
        //======================================
        // nastavenia pre vsetky objekty na scene
        //======================================
        glClear(GL_COLOR_BUFFER_BIT);  // vyskusajte, co program robi bez tohto
        //glClear(GL_DEPTH_BUFFER_BIT);// v tomto programe nie je nutne nastavit
        //-------------------------------------------------------------
        // Kvoli tomu, aby aj prvy „"frame bol vykresleny spravnou farbou,
        // treba funkciu glClearColor volat pred glClear. Ale rozdiel si
        // ani nevsimneme, lebo jeden frame trva velmi kratko.
        // glClearColor volame v main(). Tam sa naozaj vola pred zavolanim
        // kresliObdlzniky2D, teda aj pred zavolanim glClear.
        // glClear nam v kazdom novom frame vykresli farbu pozadia taku,
        // aka je nastavena v najnovsom volani funkcie glClearColor.
        //-------------------------------------------------------------
        glMatrixMode(GL_MODELVIEW);     // tu nutne s parametrom GL_MODELVIEW
        //==============================================
```

```
        // nastav transformacnu maticu na jednotkovu:  M = I
        //================================================
        glLoadIdentity();
        //========================================
        // maly ruzovy obdlznik, pohyb po diagonale
        //========================================
        glColor3f(1.0, 0.0, 1.0);
        //----------------------------------------------------------------
        // vynasob aktualnu maticu prvou translacnou maticou:  M = I*T1 = T1
        //----------------------------------------------------------------
        glTranslatef(posunX, posunX/2, 0);
        //----------------------------------------------------------------
        // Ruzovy obdlznik nakreslime pouzitim glVertex2f.
        // Nech ma „("fyzicky aj pixelovy) pomer stran sx:sy = 2:1.
        // sx = 0.04*Lmax,  sy = 0.02*Lmax   ==>  sx:sy = 2:1
        //
        // AKYKOLVEK GRAFICKY OBJEKT DEFINUJEME, AUTOMATICKY SA NAN APLIKUJE
        // AKTUALNA TRANSFORMACNA MATICA. Ta je sice ulozena pre nas
        // neviditelne, ale to nevadi. Momentalne je to matica vyssie oznacena
        // ako T1.
        //----------------------------------------------------------------
        glBegin(GL_QUADS);
                glVertex2f(-0.45*Lmax, -0.15*Lmax);  //  lavy horny roh
                glVertex2f(-0.41*Lmax, -0.15*Lmax);  // pravy horny
                glVertex2f(-0.41*Lmax, -0.17*Lmax);  // pravy dolny
                glVertex2f(-0.45*Lmax, -0.17*Lmax);  //  lavy dolny
        glEnd();
        //========================================
        // velky modry obdlznik, pohyb pozdlz osi x
        //========================================
        glColor3f(0.0, 0.0, 1.0);
        //------------------------------------------------------
        // zresetuj transformacnu maticu (nastav na jednotkovu)
        //------------------------------------------------------
        glLoadIdentity();
        glTranslatef(posunX, 0, 0);
        //----------------------------------------------------------------
        // Modry obdlznik nakreslime pouzitim specializovanej funkcie glRectf.
        // glRectf(x1, y1, x2, y2) obsahuje suradnice protilahlych vrcholov.
        //----------------------------------------------------------------
        glRectf(-0.50*Lmax, -0.20*Lmax, -0.45*Lmax, -0.30*Lmax);

        glutSwapBuffers();   // Vykresli pripravenu scenu.
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
//      glutInitDisplayMode(GLUT_RGBA);    // nie je nutne pisat
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(1600, 1000);
        glutInitWindowPosition(50, 20);
        glutCreateWindow("OpenGL: Obdlzniky");
        glutDisplayFunc(kresliObdlzniky2D);
```

```
        glutReshapeFunc(obsluhaResize);
        //-------------------------------------------------------------
        // Ak nechceme menit farbu pozadia okna „za "jazdy, tak ju staci
        // nastavit len raz, povedzme tu niekde v main().
        //-------------------------------------------------------------
        glClearColor(0.0, 0.6, 0.3, 0); // farba pozadia zelena
        glutTimerFunc(icaskrok, aktualizuj, 0);
        glutMainLoop();
        return 0;
}
```