LABORATOR 4

Controlul evenimentelor de intrare şi funcţii callback

În acest laborator se va studia controlul evenimentelor de intrare ki funcSi callback în OpenGL

Implementările bibliotecii OpenGL diferă de la un sistem grafic la altul.

Aka cum s-a exemplificat ki în conSinutul laboratoarelor anterioare biblioteca graficL OpenGL conSine funcSii de redare a primitivelor geometrice independente de sistemul de operare, de orice sistem Windows sau de platforma hardware. Ea nu conSine nici o funcSie pentru a crea sau administra ferestre de afikare pe display (windows) ki nici funcSii de citire a evenimentelor de intrare (mouse sau tastaturL). Pentru crearea ki administarea ferestrelor de afikare ki a evenimentelor de intrare se pot aborda mai multe soluSii: utilizarea directL a interfeSei Windows (Win32 API), folosirea compilatoarelor sub Windows, care conSin funcSii de acces la ferestre ki evenimente sau folosirea altor instrumente (utilitare) care înglobeazL interfaSa OpenGL în sistemul de operare respectiv.

Un astfel de utilitar este GLUT, care se compileazL ki instaleazL pentru fiecare tip de sistem Windows. Header-ul glut. h trebuie sL fie inclus în fikierele aplicaSiei, iar biblioteca glut.lib trebuie legatL (linkatL) cu programul de aplicaSie. ën lucrarea de faSL s-a folosit versiunea glut pentru VS 2013, care poate fi preluatL din internet (http://phimtk.com/video/how-to-set-up-opengl-and-glut-in-visual-studio-c++-83be8d2121aac5eb15696e.html).

Sub GLUT, orice aplicaSie se structureazL folosind mai multe **funcţii callback**.

O **funcţie callback** este o funcSie care aparSine programului aplicaSiei ki este apelatL de un alt program, în acest caz sistemul de operare, la apariSia anumitor evenimente. În GLUT sunt predefinite câteva tipuri de funcSii callback; acestea sunt scrise în aplicaSie ki pointerii lor sunt transmiki la înregistrare sistemului Windows, care le apeleazL (prin pointerul primit) în momentele necesare ale execuSiei.

În** LABORATOR 3 **la paragraful 2.3. „Controlul evenimentelor de intrare” s-au prezentat pe scurt evenimentele provocate de apLsarea unei taste sau a unuia dintre butoanele mouse-ului, de deplasarea mouse-ului ki de redimensionarea ferestrei de afikare de cLtre utilizator. Reamintim cL sunt disponibile funcSii pentru controlul ferestrei de afikare a programului.

În aplicaSile de graficL tridimensionalL termenul de fereastrL de vizualizare *(view plane window)* se referL la regiunea rectangularL care reprezintL intersecSia dintre planul de vizualizare ki volumul de vizualizare ki în care sunt proiectate toate obiectele vizibile ale scenei. Fereastra de vizualizare se afikeazL într-o poartL de afikare *(viewport)* care se definekte prin transformarea fereastrL-poartL. Poarta de afikare graficL se amplaseazL într-o fereastrL de afikare *(window)* ki legLtura dintre poarta de afikare ki fereastra de afikare este realizatL diferit, în funcSie de modul de programare folosit (interfaSL Windows sau toolkit).

În programele dezvoltate sub GLUT, corelarea dintre poarta de afikare ki fereastra Window este asiguratL de urmLtoarele funcSii:

```
void glutInit(int* argc,char**argv) ;
```

AceastŁ funcŞie iniŞializeazŁ GLUT folosind argumentele din linia de comandŁ; ea trebuie sŁ fie apelatŁ înaintea oricŁror alte funcŞi GLUT sau OpenGL.

```
void glutInitDisplayMode(unsigned int mode) ;
```

SpecificŁ caracteristicile de afiкare a culorilor кi a bufferului de adâncime кi numŁrul de buffere de imagine. Parametrul mode se obŞine prin SAU logic între valorile fiecŁrei opŞiuni. De exemplu,

```
glutlnitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH)
```

iniŞializeazŁ **afişarea culorilor** în **modul** RGB, cu douŁ buffere de imagine кi buffer de adâncime. Alte **valori ale parametrului mode** sunt: **GLUT S INGLE** (un singur buffer de imagine), **GLUT_RGBA (**modelul RGBA al culorii**), GLUT STENCIL** (validare buffer кablon) **GLUT_ACCUM (**validare buffer de acumulare**).**

```
void glutlnitWindowPosition(int x, int y);
```

specificŁ locaŞa iniŞalŁ pe ecran a colŞului stânga sus al ferestrei de afiкare.

```
void glutlnitWindowSize(int width, int heigth);
```

Defineкte dimensiunea iniŞalŁ a ferestrei de afiкare în numŁr de pixeli pe lŁŞime (width) кi înŁlŞime (heigth).

```
int glutCreateWindow(char* string);
```

CreeazŁ fereastra în care se afiкeazŁ contextul de redare (poarta) OpenGL кi retumeazŁ identificatorul ferestrei. AceastŁ fereastrŁ este afiкatŁ numai dupŁ apelul **funcţiei glutMainLoop ().**

Funcţii callback.

FuncŞile callback se definesc în program кi se înregistreazŁ în sistem prin intermediul unor funcŞi GLUT. Ele sunt apelate de sistemul de operare atunci când este necesar, în funcŞie de evenimentele apŁrute.

```
glutDisplayFunc(void(*Display)(void));
```

AceastŁ funcŞie înregistreazŁ funcŞa callback Display în care se calculeazŁ кi se afiкeazŁ imaginea. Argumentul funcŞiei este un pointer la o funcŞie fŁrŁ argumente care nu retumeazŁ nici o valoare. FuncŞa Display (a aplicaŞiei) este apelatŁ oridecâte ori este necesarŁ desenarea ferestrei: la iniŞializare, la modificarea dimensiunilor ferestrei, sau la apelul explicit al funcŞiei gluPostRedisplay ().

```
glutReshapeFunc(void(*Reshape)(int w, int h) ) ;
```

-înregistreazŁ funcŞa callback Reshape care este apelatŁ ori de câte ori se modificŁ dimensiunea ferestrei de afiкare. Argumentul este un pointer la funcŞa cu numele Reshape cu

douŁ argumente de tip întreg ki care nu retumeazŁ nici o valoare, în aceastŁ funcŞie, programul de aplicaŞie trebuie sŁ refacŁ transformarea fereastrŁ-portŁ, dat fiind cŁ fereastra de afikare ki-a modificat dimensiunile.

```
glutKeyboardFunc(void(*Keyboard)(unsigned int key,
int x, int y);
```

-înregistreazŁ funcŞia callback Keyboard care este apelatŁ atunci când se acŞioneazŁ o tastŁ. Parametrul key este codul tastei, iar x ki y sunt coordonatele (relativ la fereastra de afikare) a mouse-ului în momentul acŞionŁrii tastei.

```
glutMouseFunc(void(*MouseFunc)(unsigned int button,
int state, int x, int y);
```

-înregistreazŁ funcŞia callback MouseFunc care este apelatŁ atunci când este apŁsat sau eliberat un buton al mouse-ului. Parametrul button este codul butonului (poate avea una din constantele GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON sau GLUT_RIGHT_BUTTON) . Parametrul state indicŁ apŁsarea (GLUT_DOWN) sau eliberarea (GLUT_UP) al unui buton al mouse-ului. Parametrii x ki y simt coordonatele relativ la fereastra de afikare a mouse-ului în momentul evenimentului.

Funcții de execuție GLUT

ExecuŞia unui program folosind toolkit-ul GLUT se lanseazŁ prin apelul:
funcţiei glutMainLoop (); dupŁ ce au fost efectuate toate iniŞializŁrile ki înregistrŁrile **funcţiilor callback.** AceastŁ buclŁ de execuŞie poate fi opritŁ prin închiderea ferestrei aplicaŞiei.
În cursul execuŞiei sunt apelate funcŞile callback în momentele apariŞiei diferitelor evenimente. În cursul execuŞiei se mai poate executa un proces atunci cÃnd nu apare nici un eveniment (atunci când coada de evenimente este vidŁ). Acest proces executŁ **funcţia callback IdleFunc** înregistratŁ prin apelul funcŞiei :
glutIdleFunc (void **(*IdleFunc)** (void)); AceastŁ funcŞie este folositŁ în animaŞie.

Poarta de afişare OpenGL

Poarta de afikare mai este numitŁ context de redare *(rendering context),* ki este asociatŁ unei ferestre din sistemul Windws. DacŁ se programeazŁ folosind biblioteca GLUT, corelarea dintre fereastra Windows ki portul OpenGL este asiguratŁ de funcŞii ale acestei biblioteci. DacŁ nu se folosekte GLUT, atunci funcŞile bibliotecilor de extensie XGL, WGL sau PGL permit atribuirea unui context de afikare Windows pentru contextul de redare OpenGL ki accesul la contextul de afikare Windows *{device context).*

FuncŞia OpenGL care definekte transformarea fereastrŁ-portŁ este:

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

unde x ki y specificŁ poziŞia colŞului stânga-jos al dreptunghiului porŞi în fereastra de afikare

(window) ки au valorile implicite 0, 0. Parametrii weidth ки heigth specificЋ lЋŞimea, respectiv înЋlŞimea, porŞi de afiкare, datЋ ca numЋr de pixeli. Se reaminteкte cЋ transformarea ferestrЋ-poartЋ este componentЋ a transformЋrii din sistemul de referinŞЋ normalizat în sistemul de referinŞЋ ecran 3D .

Un pixel este reprezentat în OpenGL printr-un descriptor care defineкte mai mulŞi parametri:

Å numЋrul de biŞi/pixel pentru memorarea culorii
Å numЋrul de biŞi/pixel pentru memorarea adâncimii
Å numЋrul de buffere de imagine

Bufferul de cadru

Bufferul de cadru *(frame buffer)* conŞine toate datele care definesc o imagine ки constЋ din mai multe secŞiuni logice: bufferul de imagine (sau bufferul de culoare), bufferul de adâncime (*Z-buffer*), bufferul кablon *(stencil),* bufferul de acumulare *(acumulation).*

Bufferul de imagine *(color buffer)* în OpenGL poate conŞine una sau mai multe secŞiuni, în fiecare fiind memoratЋ culoarea pixelilor din poarta de afiкare. Redarea imaginilor folosind un singur buffer de imagine este folositЋ pentru imagini statice, cel mai frecvent în **proiectarea grafică (CAD).** În generarea interactivЋ a imaginilor dinamice, un singur buffer de imagine produce efecte nedorite, care diminueazЋ mult calitatea imagii generate.

Orice cadru de imagine începe cu ştergerea (de fapt, umplerea cu o culoare de fond) a bufferului de imagine. DupЋ aceasta sunt generaŞi pe rând pixelii care aparŞin tuturor elementelor imaginii (linii, puncte, suprafeŞe) ки intensitЋŞile de culoare ale acestora sunt înscrise în bufferul de imagine. Pentru trecerea la cadrul urmЋtor, trebuie din nou кters bufferul de imagine ки reluatЋ generarea elementelor componente, pentru noua imagine. Chiar dacЋ ar fi posibilЋ generarea ки înscrierea în buffer a elementelor imaginii cu o vitezЋ foarte mare (ceea ce este greu de realizat), tot ar exista un interval de timp în care bufferul este кters ки acest lucru este perceput ca o pâlpâire a imaginii, în grafica interactivЋ timpul necesar pentru înscrierea datelor în buffer este (în cazul cel mai fericit) foarte apropiat de intervalul de schimbare a unui cadru a imaginii (update rate) ки, dacЋ acest proces are loc simultan cu extragerea datelor din buffer ки afiкarea lor pe display, atunci ecranul va prezenta un timp foarte scurt imaginea completЋ a fiecЋrui cadru, iar cea mai mare parte din timp ecranul va fi кters sau va conŞine imagini parŞiale ale cadrului. Tehnica universal folositЋ pentru redarea imaginilor dinamice (care se schimbЋ de la un cadru la altul) este tehnica *dublului buffer de imagine.*

ën aceastЋ tehnicЋ existЋ douЋ buffere de imagine: bufferul din faŞЋ *(front*), din care este afiкatЋ imaginea pe ecran ки bufferul din spate *(back),* în care se înscriu elementele de imagine generate. Când imaginea unui cadru a fost complet generatЋ, ea poate fi afiкatЋ pe ecran printr-o simplЋ operaŞie de comutare între buffere: bufferul spate devine buffer faŞЋ, ки din el urmeazЋ sЋ fie afiкatЋ imagine cadrului curent, iar bufferul faŞЋ devine buffer spate ки în el urmeazЋ sЋ fie generatЋ imaginea noului cadru. Comutarea între bufferele de imagine se poate sincroniza cu cursa de revenire pe verticalЋ a monitorului ки atunci imaginile sunt prezentate continuu, fЋrЋ sЋ se observe imagini fragmentate sau pâlpâiri.

În OpenGL comutarea bufferelor este efectuatЋ de funcŞia SwapBuffers (). AceastЋ funcŞie trebuie sЋ fie apelatЋ la terminarea generЋrii imaginii tuturor obiectelor vizibile în fiecare

cadru.

Modul în care se execută comutarea bufferelor depinde de platforma hardware. În lipsa unui accelerator grafic, toate operaţiile OpenGL sunt executate software, bufferul spate este o zonă din memorie principală, iar bufferul faţă este memoria video din adaptorul grafic. La comutarea bufferelor care se produce la apelul funcţiei SwapBuffers (), are loc copierea bufferului back (care este implementat ca un bitmap independent de dispozitiv - DIB - device independent bitmap), în bufferul front, care este memoria de ecran a adaptorului grafic.

Definirea şi utilizarea bufferelor componente ale unui buffer de cadru OpenGL este puţin mai complexă decât modul general de execuţie descris mai sus, deoarece se oferă posibilitatea creerii atât a imaginilor cu simplu sau dublu buffer, monografice şi stereografice şi selectarea bufferului în care se scrie sau se citeşte la un moment dat.

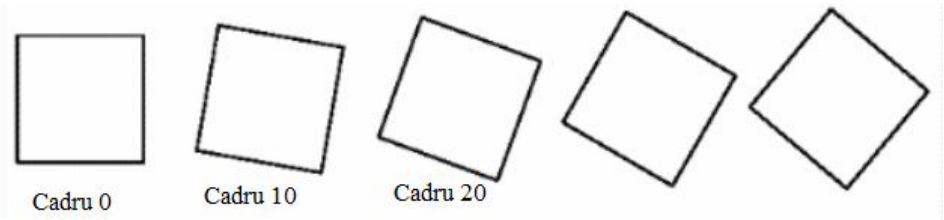

Figura 4-1

Exrcitiul 4-1 ilustrează utilizarea glutSwapBuffers (), într-un exemplu care desenează un pătrat care se roteşte, aşa cum se arată în Figura 4-1. Exemplul următor arată, de asemenea modul de utilizare a GLUT pentru a controla un **eveniment de intrare** în de a activa în dezactiva o funcţie de mers în gol. In acest exemplu, butoanele mouse-ului comută rotirea pe on în off.

Exrcitiul 4-1

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

static GLfloat spin = 0.0;

void init(void)
{
       glClearColor(0.0, 0.0, 0.0, 0.0);
       glShadeModel(GL_FLAT);
}

void display(void)
{      glClear(GL_COLOR_BUFFER_BIT);
       glPushMatrix();
       glRotatef(spin, 0.0, 0.0, 1.0);
       glColor3f(1.0, 1.0, 1.0);
       glRectf(-25.0, -25.0, 25.0, 25.0);
       glPopMatrix();
       glutSwapBuffers();
}

void spinDisplay(void)
```

```
{
	spin = spin + 2.0;
	if (spin > 360.0)
		spin = spin - 360.0;
	glutPostRedisplay();
}

void reshape(int w, int h)
{
	glViewport(0, 0, (GLsizei)w, (GLsizei)h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

void mouse(int button, int state, int x, int y)
{
	switch (button) {
	case GLUT_LEFT_BUTTON:
		if (state == GLUT_DOWN)
			glutIdleFunc(spinDisplay);
		break;
	case GLUT_MIDDLE_BUTTON:
		if (state == GLUT_DOWN)
			glutIdleFunc(NULL);
		break;
	default:
		break;
	}
}

/*
*  Cerere de afisare in modul dublubuffer.
*  Inregistreaza functia callback MouseFunc
*/
int main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(100, 100);
	glutCreateWindow(argv[0]);
	init();
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
	glutMouseFunc(mouse);
	glutMainLoop();
	return 0;
}
```

FuncŞia:

```
void glDrawBuffer(GLenum mode);
```

stabilekte bufferul în care se deseneazŁ primitivele geometrice.

Parametrul mode poate lua una din valorile: GL_NONE, GL_FRONT_LEFT, GL_FRONT_RIGHT, GL_BACK_LEFT, GL_BACK_RIGHT, GL_FRONT, GL_BACK, GL_LEFT, GL_RIGHT, GL_FRONT_AND_BACK. Valoarea implicitŁ este GL_FRONT

pentru imagini cu un singur buffer ki GL_BACK pentru imagini cu dublu buffer. Celelate opSuni se referL la imagini stereoscopice.

FuncSa `glReadBuffer (GLenum mode)` stabilekte bufferul curent din care se citesc pixelii sursL în operaSi de combinare sau acumulare. Parametrul mode poate lua una din valorile descrise mai sus. Valoarea implicitL este GL_FRONT pentru imagini monografice cu un singur buffer de culoare sau GL_BACK pentru imagini monografice cu douL buffere de culoare.

Bufferul de adâncime *(depth buffer)* memoreazL adâncimea fiecLrui pixel ki, prin aceasta, permite eliminarea suprafeSelor ascunse. Bufferul de adâncime conSine acelaki numLr de locaSi ca ki un buffer de imagine, fiecare locaSie corespunzând unui pixel, de o anumitL adresL. Valoarea memoratL în locaSia corespunzLtoare unui pixel este distanSa faSL de punctul de observare (adâncimea pixelului). La generarea unui nou pixel cu aceeaki adresL, se comparL adâncimea noului pixel cu adâncimea memoratL în bufferul de adâncime, ki noul pixel înlocuiekte vechiul pixel (îl ñascundeò) dacL este mai apropiat de punctul de observare. Bufferul de adâncime se mai numekte ki Z-buffer, de la coordonata z, care reprezintL adâncimea în sistemul de referinSL ecran 3D.

Bufferul kablon *{stencil buffer)* este un buffer auxiliar care este utilizat pentru restricSionarea desenLrii în anumite zone ale porSi de afikare. Se pot obSine astfel imagini încadrate în anumite kabloane, ca de exemplu, instrumente de afikare folosite în simulatoare de antrenament. Bufferul kablon poate fi folosit ki pentru redarea suprafeSelor coplanare.

Bufferul de acumulare *{accumulation buffer)* este folosit pentru crearea imaginilor antialising prin acumularea intensitLSilor de culoare a pixelilor rezultaS prin ekantionarea imaginii.

Exrcitiul 4-2

Program OpenGL pentru **interacţiunea cu mouse-ul**:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include <GL/glut.h>

void gasket();
void myinit()
{
	glClearColor(1.0, 1.0, 1.0, 1.0); /* fundal alb */
	glColor3f(1.0, 0.0, 0.0); /* desenare in rosu */
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-10.0, 10.0, -10.0, 10.0);
	glMatrixMode(GL_MODELVIEW);
}



void draw(void)
{
	int r = 10;
	GLfloat theta;

	glClear(GL_COLOR_BUFFER_BIT);
```

```
	glBegin(GL_POLYGON);
	for (theta = 0; theta <= 360; theta += 0.01)
	{
		glVertex2f(5 * sin(theta*3.142 / 180), 2 * cos(theta*3.142 / 180));
	}
	glEnd();
	glFlush();
}


void myMouse(int button, int state, int x, int y)
{
	if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
	{
		draw();

	}
	else if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
	{
		gasket();

	}



}

void gasket()
{
	GLfloat v[3][2] = { { 1.0, 1.0 }, { 6.0, 1.0 }, { 3.5, 5.5 } };    // Alocare
coordonate pentru a desena 3 puncte ale unui triunghi
	GLfloat p[2] = { 2.5, 2.5 };                                       // Introdu punctul
initial
	int j, i;

	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POINTS);
	for (i = 0; i<50000; i++)
	{
		j = rand() % 3;                                            // Selecteaza un
vertex aleatoriu in fiecare iteratie
		p[0] = (p[0] + v[j][0]) / 2.0;                             // Calculeaza
mijlocul distantei dintre P si vertexul selectat
		p[1] = (p[1] + v[j][1]) / 2.0;                             // Calculeaza
mijlocul distantei dintre P si vertexul selectat

		glVertex2fv(p);                                          // Afiseaza acest punct
pe ecran
	}
	glEnd();
	glFlush();
}




void display(void)
{

	glClear(GL_COLOR_BUFFER_BIT);
	glFlush();
}
```

```
int main(int argc, char** argv)
{



	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); /* implicit, nu este necesar */
	glutInitWindowSize(500, 500); /* 500 x 500 marimea ferestrei */
	glutInitWindowPosition(0, 0); /* plaseaza fereastra initiala stanga sus */
	glutCreateWindow("Control butoane mouse"); /* Titlu ferestrei */
	glutDisplayFunc(display);
	glutMouseFunc(myMouse);  /* afiseaza functia de control cand fereastra este
deschisa */
	myinit(); /* setare attribute */
	glutMainLoop(); /* intervine bucla eveniment */
	return 0;
}
```

Exrcitiul 4-3

Program OpenGL pentru **interacţiunea cu mouse şi tastatura**:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include <GL/glut.h>

void myinit()
{
	glClearColor(1.0, 1.0, 1.0, 1.0);
	glColor3f(1.0, 0.0, 0.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-10.0, 10.0, -10.0, 10.0);
	glMatrixMode(GL_MODELVIEW);
}

void draw(void)
{
	int r = 10;
	GLfloat theta;

	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POLYGON);
	for (theta = 0; theta <= 360; theta += 0.01)
	{
		glVertex2f(5 * sin(theta*3.142 / 180), 2 * cos(theta*3.142 / 180));
	}
	glEnd();
	glFlush();
}

void gasket()
{
```

```
	GLfloat v[3][2] = { { 1.0, 1.0 }, { 6.0, 1.0 }, { 3.5, 5.5 } };	// Alocare coordonate triunghi
	GLfloat p[2] = { 2.5, 2.5 };					// Da pozitia punctului initial
	int j, i;

	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POINTS);
	for (i = 0; i<50000; i++)
	{
		j = rand() % 3;					// Selecteaza vertex
		p[0] = (p[0] + v[j][0]) / 2.0;			// Calculeaza mijlocul distantei
		p[1] = (p[1] + v[j][1]) / 2.0;			// Calculeaza mijlocul dist

		glVertex2fv(p);				// Afiseaza acest punct
	}
	glEnd();
	glFlush();
}

void myMouse(int button, int state, int x, int y)
{
	if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
	{
		draw();
	}
	else if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
	{
		gasket();
	}

}

void display(void)
{

	glClear(GL_COLOR_BUFFER_BIT);
	glFlush();
}

void myKey(unsigned char key, int x, int y)
{
	if (key == 'g' || key == 'G')
	{
		gasket();
	}
	else if (key == 'c' || key == 'C')
	{
		draw();
	}
	else exit(0);
}




int main(int argc, char** argv)
{
```

```
	/* Initializare GLUT */

	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500, 500);
	glutInitWindowPosition(0, 0);
	glutCreateWindow("Control tastatura");
	/* Titlu ferestrei */
	glutDisplayFunc(display);
	glutMouseFunc(myMouse);
	/* Afiseaza functia atunci cand fereastra este deschisa */
	glutKeyboardFunc(myKey);


	myinit();

	glutMainLoop();
	return 0;
}
```

Exrcitiul 4-4

Program OpenGL care permite control prin meniuri simple

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include <GL/glut.h>

void myinit()
{
	glClearColor(1.0, 1.0, 1.0, 1.0);
	glColor3f(1.0, 0.0, 0.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-10.0, 10.0, -10.0, 10.0);
	glMatrixMode(GL_MODELVIEW);
}



void draw(void)
{
	int r = 10;
	GLfloat theta;

	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POLYGON);
	for (theta = 0; theta <= 360; theta += 0.01)
	{
		glVertex2f(5 * sin(theta*3.142 / 180), 2 * cos(theta*3.142 / 180));
	}
	glEnd();
	glFlush();
}



void gasket()
```

```
{


	GLfloat v[3][2] = { { 1.0, 1.0 }, { 6.0, 1.0 }, { 3.5, 5.5 } };	// Alocare
coord
	GLfloat p[2] = { 2.5, 2.5 };					// Punct initial
oint
	int j, i;

	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POINTS);
	for (i = 0; i<50000; i++)
	{
		j = rand() % 3;					// Selecteaza
vertex
		p[0] = (p[0] + v[j][0]) / 2.0;			// Calculeaza
mijlocul dist
		p[1] = (p[1] + v[j][1]) / 2.0;			// Calculeaza
mijlocul

		glVertex2fv(p);				// Afiseaza acest punct
pe ecran
	}
	glEnd();
	glFlush();
}

void display(void)
{

	glClear(GL_COLOR_BUFFER_BIT);
	glFlush();
}



void demo_menu(int id)
{
	switch (id)
	{
	case 1: draw();
		break;
	case 2: gasket();
		break;
	case 3: exit(0);
		break;
	}

}
int main(int argc, char** argv)
{


	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500, 500);
	glutInitWindowPosition(0, 0);
	glutCreateWindow("Fereastra creare meniu");
	glutCreateMenu(demo_menu);
	glutAddMenuEntry("Cerc", 1);
	glutAddMenuEntry("Gasket", 2);
	glutAddMenuEntry("Iesire", 3);
```

```
        glutAttachMenu(GLUT_RIGHT_BUTTON);
        glutDisplayFunc(display);



        myinit();

        glutMainLoop();
        return 0;
}
```