Curs 3/ Tema laborator:

Sa se foloseasca toate argumentele functie glBegin…..

Aplicatia 1

for (int dist = 0, i = 1; i <= 3; i++)

{

dist += 40;

glVertex2i(40\*i+dist, 40);

}

A picture containing graphical user interface

Description automatically generated

#include <iostream>

#include <gl/freeglut.h>

#include<math.h>

int width = 400;

int height = 400;

int psize = 40;

int distx = 0;

int disty = 0;

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(psize);

glBegin(GL\_POINTS);

disty = 20;

for (int k = 0; k < 3; k++)

{

distx = 20;

for (int j = 0; j < 3; j++)

{

double r = ((double)rand() / (RAND\_MAX));//0 pana 1.

double g = ((double)rand() / (RAND\_MAX));

double b = ((double)rand() / (RAND\_MAX));

glColor3d(r, g, b);

glVertex2i(40 \* j + distx, 40 \* k + disty);

}

disty += 0;

}

glEnd();

glFlush();

}

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h); //stabilirea viewportului la dimensiunea ferestrei

glMatrixMode(GL\_PROJECTION); //specificare matrice modificabila la valoare argumentului de modificare

glLoadIdentity(); //initializarea sistemului de coordonate

gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h); //stabileste volumul de vedere folosind o proiectie ortografica

}//end reshape()

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(400, 300);

glutCreateWindow("Puncte");

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}

//sa se realizeze in oglida fata de cele 3 varfuri ale ferestrei imaginea din coltul stanga joi

Aplicatia 2

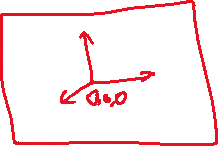
glBegin(GL\_LINES);

**for(int i=0;i<=4;++i)**



**{**

**glVertex3f(0,0,0); //x,y,z//**



**glVertex3f(1-i/4.0, i/4.0, 0); //1,0,0**

}

**glEnd();**



Chart

Description automatically generated

#include <iostream>

#include <gl/freeglut.h>

void Display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0, 1, 0);

glBegin(GL\_LINES);

// Cadran 1

for (int i = 0; i < 20; i++)

{

glVertex3f(0, 0, 0);

glVertex3f(1 - i / 20.0, i / 20.0, 0);

} // Cadran 2

glColor3f(1, 0.4, 0);

for (int i = 0; i < 20; i++)

{

glVertex3f(0, 0, 0);

glVertex3f(-1 + i / 20.0, i / 20.0, 0);

} // Cadran 3

glColor3f(1, 0.4, 1);

for (int i = 0; i < 20; i++)

{

glVertex3f(0, 0, 0);

glVertex3f(-1 + i / 20.0, -i / 20.0, 0);

}

// Cadran 4

glColor3f(0.8, 0.4, 0.2);

for (int i = 0; i < 20; i++)

{

glVertex3f(0, 0, 0);

glVertex3f(1 - i / 20.0, -i / 20.0, 0);

}

glEnd(); glFlush(); glColor3f(0, 0, 1);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(600, 600);

//se specifica modelul de culoare al ferestrei: un singur buffer si culoare RGB

glutCreateWindow("Curs 3-2021");

glutDisplayFunc(Display);

glutMainLoop();

return 0;

}

Aplicatia 3

Se se roteasca imaginea

Background pattern

Description automatically generated with low confidence

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

Înregistrează funcţia callback Keyboard care este apelată atunci când se acţionează o tastă. Parametrul key este codul tastei, iar x şi y sunt coordonatele (relativ la fereastra de afişare) a mouse-ului în momentul acţionării tastei.

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

Înregistrează funcţia callback MouseFunc care este apelată atunci când este apăsat sau eliberat un buton al mouse-ului.

Parametrul button este codul butonului si poate avea una din constantele GLUT\_LEFT\_BUTTON, GLUT\_MIDDLE\_BUTTON sau GLUT\_RIGHT\_BUTTON) . Parametrul state indică apăsarea (GLUT\_DOWN) sau eliberarea (GLUT\_UP) al unui buton al mouse-ului. Parametrii x şi y sunt coordonatele relativ la fereastra de afişare a mouse-ului în momentul evenimentului.

<https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glRotate.xml>

glRotatef

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**glutPostRedisplay**

***glutPostRedisplay*** marchează fereastra curentă ca fiind nevoie să fie reafișată

***Usage***

***void glutPostRedisplay(void);***

***Description***

***Mark the normal plane of current window as needing to be redisplayed. The next iteration through glutMainLoop, the window's display callback will be called to redisplay the window's normal plane. Multiple calls to glutPostRedisplay before the next display callback opportunity generates only a single redisplay callback. glutPostRedisplay may be called within a window's display or overlay display callback to re-mark that window for redisplay.***

***Logically, normal plane damage notification for a window is treated as a glutPostRedisplay on the damaged window. Unlike damage reported by the window system, glutPostRedisplay will not set to true the normal plane's damaged status (returned by glutLayerGet(GLUT\_NORMAL\_DAMAGED).***

#include <iostream>

#include <gl/freeglut.h>

void roteste\_Y(int p\_grade)

{

glRotatef(p\_grade, 0.0, 1.0, .0);

glutPostRedisplay();

}

void roteste\_X(int p\_grade)

{

glRotatef(p\_grade, 0., 1., .0);

glutPostRedisplay();

}

void OnKeyPress(unsigned char key, int x, int y)

{

if (key == 27)

exit(0);

switch (key)

{

case 'q':

case 'Q':

roteste\_Y(3);

break;

case 'w':

case 'W':

roteste\_Y(-3);

break;

case 'a':

case 'A':

roteste\_X(3);

break;

case 's':

case 'S':

roteste\_X(-3);

break;

}

}

void OnMouseClick(int button, int state, int x, int y)

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

roteste\_Y(20);

}

if (button == GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN)

{

roteste\_Y(-20);

}

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

int l = 10;

for (double i = 0; i <= l; i++) {

glBegin(GL\_LINE\_LOOP);

glColor3f(1 - i / l, i / l, 1);

glVertex3f(1 - i / l, 0, 0);//l=10; i=0;1,0,0

glVertex3f(0, 1 - i / l, 0);

glVertex3f(-(1 - i / l), 0, 0);

glVertex3f(0, -(1 - i / l), 0);

glEnd();

}

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);//se specifica modelul de culoare al ferestrei: un singur buffer si culoare RGB

glutCreateWindow("Curs 3 2021");

glutKeyboardFunc(OnKeyPress);

glutMouseFunc(OnMouseClick);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Tema propusa

Chart

Description automatically generated