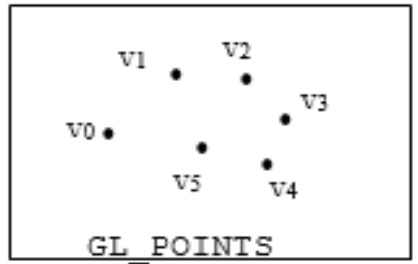
Grafica pe calculator
23.11.2022

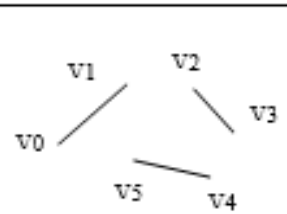
OpenGL

Tipurile de primitive geometrice

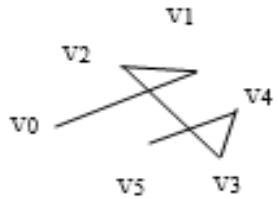
Proprietate	Primitivă
GL_POINTS	Desenează n puncte
GL_LINES	Desenează segmentele de dreaptă izolate (v_0,v_1), (v_2,v_3),… ş.a.m.d. Dacă n este impar ultimul vârf este ignorat
GL_LINE_STRIP	Desenează linia poligonală formată din segmentele (v_0,v_1), (v_1,v_2), .. . …(v_{n-2}, v_{n-1})
GL_LINE_LOOP	La fel ca primitiva GL_LINE_STRIP, dar se mai desenează segmentul (v_n, v_0) care închide o buclă.
GL_TRIANGLES	Desenează o serie de triunghiuri folosind vârfurile (v_0,v_1,v_2), (v_3,v_4,v_5), ş.a.m.d. Dacă n nu este multiplu de 3, atunci ultimele 1 sau 2 vârfuri sunt ignorate.
GL_TRIANGLE_STRIP	Desenează o serie de triunghiuri folosind vârfurile (v_0,v_1,v_2), (v_2,v_1,v_3),..ş.a.m.d. Ordinea este aleasă astfel ca triunghiurile să aibă aceeaşi orientare, deci să poată forma o suprafaţă închisă.
GL_TRIANGLE_FAN	Desenează triunghiurile (v_0,v_1,v_2), (v_0,v_2,v_3), ş.a.m.d.
GL_QUADS	Desenează o serie de patrulatere (v_0,v_1,v_2,v_3), (v_4,v_5,v_6,v_7), ş.a.m.d. Dacă n nu este multiplu de 4, ultimele 1, 2 sau 3 vârfuri sunt ignorate.
GL_QUADS_STRIP	Desenează o serie de patrulatere (v_0,v_1,v_3,v_2), (v_3,v_2,v_5,v_4), ş.a.m.d Dacă $n < 4$, nu se desenază nimic. Dacă n este impar, ultimul vârf este ignorat.
GL_POLYGON	Desenează un poligon cu n vârfuri, $(v_0,v_1,…v_{n-1})$. Dacă poligonul nu este convex, rezultatul este impredictibil.



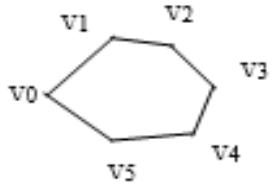
GL_POINTS



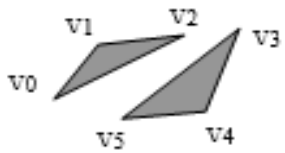
GL_LINES



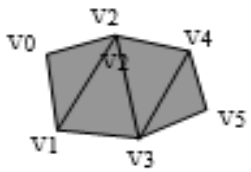
GL_LINE_STRIP



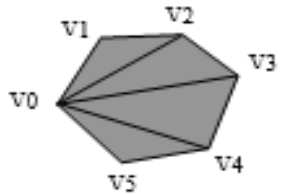
GL_LINE_LOOP



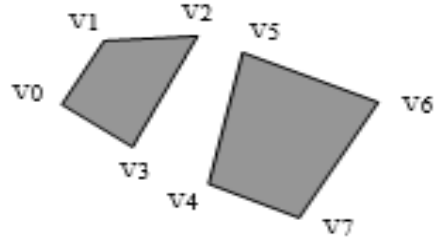
GL_TRIANGLES



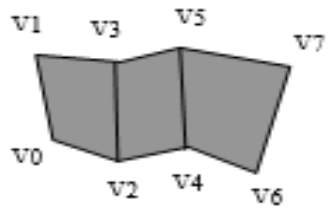
GL_TRIANGLE_STRIP



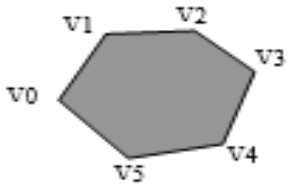
GL_TRIANGLE_FAN



GL_QUADS



GL_QUAD_STRIP



GL_POLYGON

Primitivele de tip suprafaţă (triunghiuri, patrulatere, poligoane) pot fi desenate în modul “cadru de sârmă” (*wireframe*), sau în modul plin (*fill*), prin setarea variabilei de stare GL_POLYGON_MODE folosind funcţia:

- void glPolygonMode(GLenum face, GLenum mode);
- unde argumentul mode poate lua una din valorile:
- GL_POINT : se desenează numai vârfurile primitivei, ca puncte în spaţiu, indiferent de tipul acesteia.
- GL_LINE: muchiile poligoanelor se desenează ca segmente de dreaptă.
- GL_FILL: se desenează poligonul plin.

Argumentul face se referă la tipul primitivei geometrice din punct de vedere al orientării căreia i se aplică modul de redare mode.

OpenGL admite primitive orientate direct (frontface) şi primitive orientate invers (backface). Argumentul face poate lua una din valorile: GL_FRONT, GL_BACK sau GL_FRONT_AND_BACK, pentru a se specifica primitive orientate direct, primitive orientate invers şi, respectiv ambele tipuri de primitive.

Tipuri de date OpenGL

Sufix	Tipul de date	Corespondentul în C	Tipul definit în OpenGL
b	8-bit integer	signed char	GLbyte
s	16-bit integer	Short	GLshort
i	32-bit integer	int sau long	GLint, GLsizei
f	32-bit floating-point	Float	GLfloat, GLclampf
d	64-bit floating-point	Double	GLdouble, GLclampd
ub	8-bitunsigned integer	unsigned char	GLubyte, GLboolean
us	16-bit unsigned integer	unsigned short	GLushort
ui	32-bit unsigned integer	unsigned int sau unsigned long	GLuint,GLenum, GLbitfield

GLUT include o serie de rutine pentru generarea de obiecte geometrice 3D ușor de recunoscut.

The routines generate normals appropriate for lighting but do not generate texture coordinates (except for the teapot).

- 11.1 glutSolidSphere, glutWireSphere
- 11.2 glutSolidCube, glutWireCube
- 11.3 glutSolidCone, glutWireCone
- 11.4 glutSolidTorus, glutWireTorus
- 11.5 glutSolidDodecahedron, glutWireDodecahedron
- 11.6 glutSolidOctahedron, glutWireOctahedron
- 11.7 glutSolidTetrahedron, glutWireTetrahedron
- 11.8 glutSolidIcosahedron, glutWireIcosahedron
- 11.9 glutSolidTeapot, glutWireTeapot

Afişarea obiectelor 3D predefinite

GLUT conţine funcţii pentru afişarea următoarelor obiecte 3D:

con	icosaedru	teapot
cub	octaedru	tetraedru
dodecaedru	sfera	tor

Aceste obiecte pot fi afişate prin familii de curbe sau ca obiecte solide.

Exemplu: funcţii de desenare cub, sferă si tor prin două familii de curbe şi ca solide.

Desenare cub de latură size prin două familii de curbe

void **glutWireCube**(GLdouble *size*);

Desenare cub solid de latură size

void **glutSolidCube**(GLdouble *size*);

Desenare sferă prin două familii de curbe

void **glutWireSphere**(GLdouble *radius*, GLint *slices*, GLint *stacks*);

Desenare sferă solidă

void **glutSolidSphere**(GLdouble *radius*, GLint *slices*, GLint *stacks*);

Desenare tor prin două familii de curbe

void **glutWireTorus**(GLdouble *innerRadius*, GLdouble *outerRadius*, GLint *nsides*, GLint *rings*);

Desenare tor solid

void **glutSolidTorus**(GLdouble *innerRadius*, GLdouble *outerRadius*, GLint *nsides*, GLint *rings*);

Desenare Teapot

void **glutWireTeapot**(GLdouble *size*);

void **glutSolidTeapot**(GLdouble *size*);

Alte funcţii sunt:

void glutWireIcosahedron(void);
void glutSolidIcosahedron(void);

void glutWireOctahedron(void);
void glutSolidOctahedron(void);

void glutWireTetrahedron(void);
void glutSolidTetrahedron(void);

void glutWireDodecahedron(GLdouble radius);
void glutSolidDodecahedron(GLdouble radius);

void glutWireCone(GLdouble radius, GLdouble height, GLint slices,GLint stacks);
void glutSolidCone(GLdouble radius, GLdouble height, GLint slices,GLint stacks);

Toate aceste obiecte sunt desenate centrate în originea sistemului de coordonate real.
În momentul în care se fac modificări asupra unui obiect poate apare efectul de “pâlpâire” a imaginii.
Pentru evitarea acestui efect se asociază ferestrei aplicaţiei un buffer dublu. Astfel, într-un buffer se păstrează imaginea nemodificată (imaginea ce este afişată pe ecran), iar în cel de-al doilea se construieşte imaginea modificată. În momentul în care s-a terminat construirea imaginii modificate se interschimbă buffer-ele.

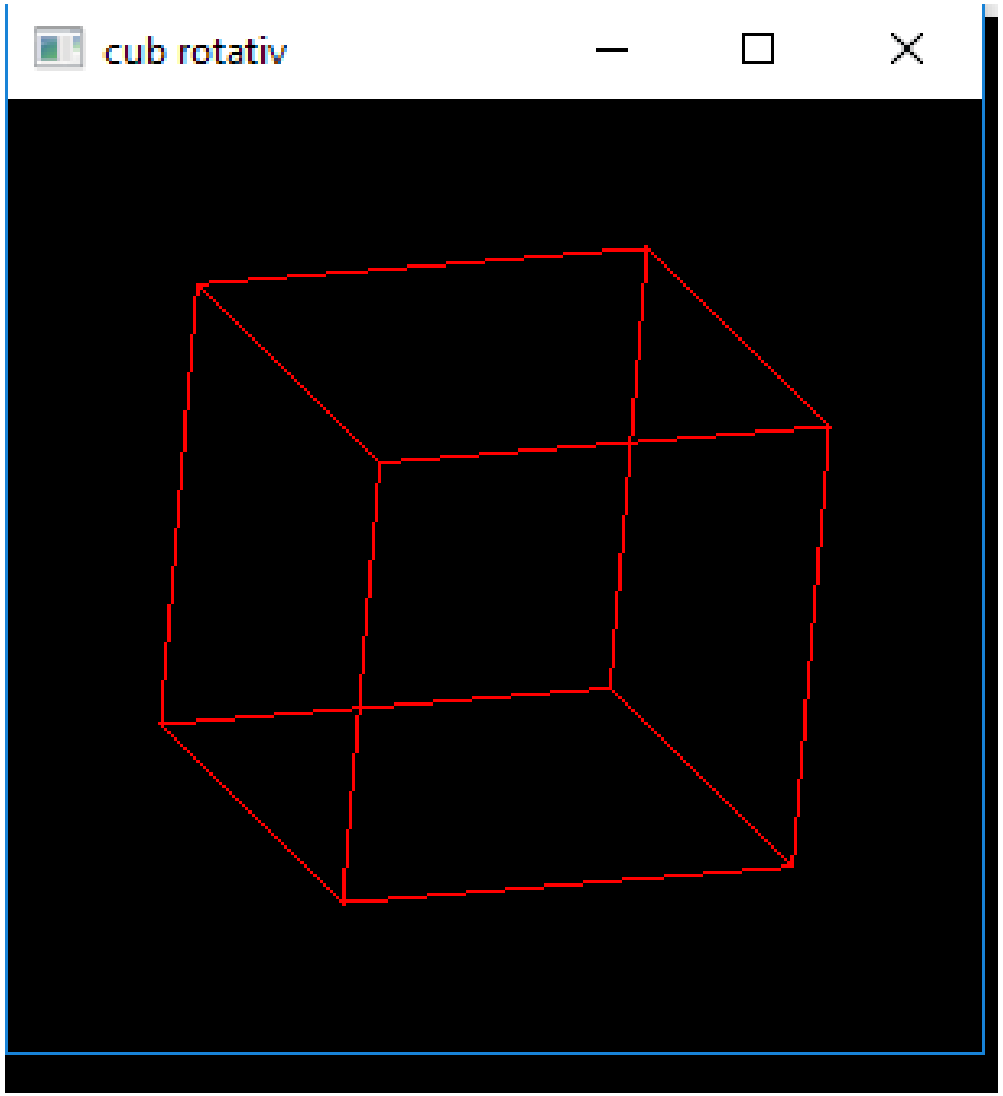
Pentru interschimbarea bufferelor se foloseşte funcţia: **glutSwapBuffers :**

```
glTranslated(x,y,z);
glTranslatef(x,y,z);
```

void glutSwapBuffers(void) ;

```
#include "stdafx.h"
#include <gl/freeglut.h>
 int GAngle = 30;

void display(void) {
	glClear(GL_COLOR_BUFFER_BIT);

	glLoadIdentity();// Initializeaza sistemul de coordonate
glRotated(GAngle, 1, 1, 1);//face o rotatie a unghiului in jurul vectorilor
x,y,z
	glColor3f(1, 0, 0);
	glutWireCube(0.5);//cub schelet
	GAngle = GAngle + 1;
	glFlush();
}
 void Timer(int extra) {
	glutPostRedisplay();//fereastra curenta este reafisata
	glutTimerFunc(30, Timer, 0);//seteaza timer-ul pentru fereastra curenta
}
```

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

        glutInit(&argc, argv);

        //glutInitWindowSize(640, 480);
        glutCreateWindow("cub rotativ");
        glutDisplayFunc(display);
        glutTimerFunc(0, Timer, 0);
        glutMainLoop();
        return 0;
}
```



```
#include "stdafx.h"
#include <gl/freeglut.h>


void Display_my_pot()
{
static float alpha = 20;

glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0.1, 0.9, 1.0);
glLoadIdentity();
```

glPopMatrix ();

```
// Pop matricea veche fără transformări.
// este salvează ecranul curent pentru o stiva
glRotatef(alpha, 1.9, 0.6, 0);
glutWireTeapot(0.3);
glPushMatrix ();  // Setați matricea curentă pe stivă
//încărca datele din stivă
 glFlush();
alpha = alpha + 0.1;

glutPostRedisplay();
}
```

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

glutInit(&argc, argv);

//glutInitWindowSize(640, 480);
glutInitWindowPosition(10, 10);
glutCreateWindow("Teapot");
glutDisplayFunc(Display_my_pot);
glutMainLoop();
return 0;
}
```