|  |  |  |
| --- | --- | --- |
|  | **Universitatea *Transilvania* din Brașov**  **Facultatea de Inginerie Electrică și Știința Calculatoarelor**  **Departamentul Automatică și Tehnologia Informației** |  |

**PROIECT EGC**

|  |  |
| --- | --- |
| Cocu Daniel  *Gr. 4LF321, TI 2* | homer.bmp |

**BRAȘOV, 2024**

**Cuprins**

[1. Logo Unitbv 3D 2](#_Toc9981015)

[1.1. Introducere 2](#_Toc9981016)

[1.2. Descrierea clasei Line 2](#_Toc9981017)

[1.3. Descrierea clasei ShortLine 3](#_Toc9981018)

1.4 Descrierea clasei LongVerticalLine …………………………………………………………………………………………..

1.5 Descrierea clasei DoubleLine

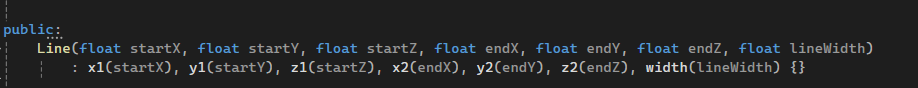
# 1.1 Introducere

Acest document oferă o explicație detaliată a codului OpenGL scris în C++, folosind biblioteca GLUT pentru a crea și desena linii, un pătrat și linii verticale într-o fereastră grafică. Codul include manipularea mouse-ului pentru a deplasa desenul pe ecran.

# 1.2Descrierea clasei Line

Clasa `Line` reprezintă o linie în spațiul 3D. Membrii săi privați sunt coordonatele celor două capete ale liniei (`x1`, `y1`, `z1` și `x2`, `y2`, `z2`) și lățimea liniei (`width`).

## Constructorul clasei Line

Constructorul inițializează coordonatele celor două capete ale liniei și lățimea acesteia.

## Funcția draw a clasei Line

A screen shot of a computer code

Description automatically generated

Această funcție desenează linia utilizând funcțiile OpenGL. `glLineWidth` setează lățimea liniei, `glBegin(GL\_LINES)` începe desenarea unei linii, `glVertex3f` specifică coordonatele unui vârf, iar `glEnd` finalizează desenarea.

# 1.3 Descrierea clasei ShortLine

Clasa `ShortLine` reprezintă o linie scurtă în spațiul 3D. Aceasta este similară cu clasa `Line`, având aceleași caracteristici și metode pentru desenarea unei linii. Clasa conține coordonatele celor două capete ale liniei și lățimea liniei.

## Membrii clasei `ShortLine`

### Membrii privați

- float x1, y1, z1; - coordonatele punctului de start al liniei.

- float x2, y2, z2; - coordonatele punctului de final al liniei.

- float width; - lățimea liniei.

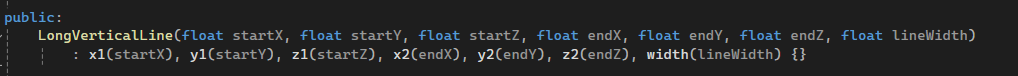
### Membrii publici

- ShortLine(float startX, float startY, float startZ, float endX, float endY, float endZ, float lineWidth);- Constructorul clasei.

- void draw() const; - Metoda pentru desenarea liniei.

## Constructorul clasei ShortLine

Constructorul clasei `ShortLine` inițializează coordonatele celor două capete ale liniei și lățimea acesteia.



## Parametri

- `float startX` - Coordonata X a punctului de start al liniei.

- `float startY` - Coordonata Y a punctului de start al liniei.

- `float startZ` - Coordonata Z a punctului de start al liniei.

- `float endX` - Coordonata X a punctului de final al liniei.

- `float endY` - Coordonata Y a punctului de final al liniei.

- `float endZ` - Coordonata Z a punctului de final al liniei.

- `float lineWidth` - Lățimea liniei.

## Metoda draw

Metoda `draw` este responsabilă pentru desenarea liniei folosind funcțiile OpenGL.

A screen shot of a computer code

Description automatically generated

## Descriere

- `glLineWidth(width);` - Setează lățimea liniei la valoarea specificată de membrul `width`.

- `glBegin(GL\_LINES);` - Începe definiția unui primitiv de tip linie.

- `glVertex3f(x1, y1, z1);` - Specifică coordonatele primului vârf al liniei.

- `glVertex3f(x2, y2, z2);` - Specifică coordonatele celui de-al doilea vârf al liniei.

- `glEnd();` - Încheie definiția primitivului de tip linie.

### Descrierea clasei `LongVerticalLine`

Clasa `LongVerticalLine` reprezintă o linie verticală lungă în spațiul 3D. Membrii săi și funcțiile sunt, de asemenea, similare cu cei ai clasei `Line`.

## Descrierea clasei DoubleLine

Clasa `DoubleLine` reprezintă două linii paralele în spațiul 3D. Constructorul său calculează coordonatele pentru a doua linie adiacentă celei originale.

Constructorul clasei DoubleLine

A computer screen with white text

Description automatically generatedConstructorul inițializează coordonatele și lățimea pentru prima linie și calculează coordonatele pentru a doua linie.

## Funcția dra a clasei DoubleLine

A screen shot of a computer program

Description automatically generated

Această funcție desenează cele două linii folosind funcțiile OpenGL. Desenarea celor două linii se face prin specificarea coordonatelor vârfurilor în interiorul aceluiași `glBegin` și `glEnd`.

## Variabile globale

A screen shot of a computer code

Description automatically generated

Aceste variabile globale includ pointeri către obiectele de tip `Line`, `ShortLine`, `LongVerticalLine`, `DoubleLine`, dimensiunile ferestrei și variabile pentru gestionarea intrării mouse-ului.

## Funcția display

A screen shot of a computer code

Description automatically generated

Funcția `display` curăță buffer-ul de culoare și mută originea sistemului de coordonate cu `glTranslatef` pentru a permite deplasarea desenului. Apoi, desenează toate liniile și pătratul. `glPopMatrix` revine la matricea inițială, iar `glFlush` asigură că toate comenziile OpenGL sunt executate.

## Funcția init

A screen shot of a computer screen

Description automatically generated

Funcția `init` inițializează fundalul ferestrei la negru (`glClearColor`), setează modul matricei la `GL\_PROJECTION`, resetează matricea curentă la identitate (`glLoadIdentity`) și definește un volum de vizualizare ortografic bidimensional (`gluOrtho2D`).

# Funcțiile de gestionare a mouse-ului

## Funcția mousePressedFunc

A computer screen shot of a program code

Description automatically generated

Funcția `mousePressedFunc` gestionează evenimentele de apăsare și eliberare a butonului stâng al mouse-ului. Când butonul stâng este apăsat, `mousePressed` devine `true` și coordonatele mouse-ului sunt salvate. Când butonul este eliberat, `mousePressed` devine `false`.

## Funcția mouseMovedFunc

A computer screen with white text

Description automatically generated

Funcția `mouseMovedFunc` gestionează evenimentele de mișcare a mouse-ului atunci când butonul stâng este apăsat. Actualizează offset-urile de translație și repostează cererea de redisplay.

## Funcția main

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

float startX = 100.0;

float startY = windowHeight - 100.0;

float startZ = 0.0;

float endX = 700.0;

float endY = windowHeight - 100.0;

float endZ = 0.0;

float lineWidth = 40.0;

line1 = new Line(startX, startY, startZ, endX, endY, endZ, lineWidth);

float secondLineStartY = startY - 50.0;

float secondLineEndY = endY - 50.0;

line2 = new Line(startX, secondLineStartY, startZ, endX, secondLineEndY, endZ, lineWidth);

float squareSize = 100.0;

float squareStartX = (startX + endX - squareSize) / 2;

float squareStartY = secondLineStartY - 150.0;

square[0] = new ShortLine(squareStartX, squareStartY, startZ, squareStartX + squareSize, squareStartY, startZ, lineWidth);

square[1] = new ShortLine(squareStartX + squareSize, squareStartY, startZ, squareStartX + squareSize

, squareStartY + squareSize, startZ, lineWidth);

square[2] = new ShortLine(squareStartX + squareSize, squareStartY + squareSize, startZ, squareStartX, squareStartY + squareSize, startZ, lineWidth);

square[3] = new ShortLine(squareStartX, squareStartY + squareSize, startZ, squareStartX, squareStartY, startZ, lineWidth);

float verticalLineStartXRight = squareStartX + squareSize + 50.0;

float verticalLineStartY = squareStartY + squareSize;

float verticalLineEndXRight = verticalLineStartXRight;

float verticalLineEndY = verticalLineStartY - (2 \* squareSize) - 50.0;

verticalLine1Right = new LongVerticalLine(verticalLineStartXRight, verticalLineStartY, startZ, verticalLineEndXRight, verticalLineEndY, startZ, lineWidth);

verticalLine2Right = new LongVerticalLine(verticalLineStartXRight + 100.0, verticalLineStartY, startZ, verticalLineEndXRight + 100.0, verticalLineEndY, startZ, lineWidth);

float verticalLineStartXLeft = squareStartX - 50.0;

float verticalLineEndXLeft = verticalLineStartXLeft;

verticalLine1Left = new LongVerticalLine(verticalLineStartXLeft, verticalLineStartY, startZ, verticalLineEndXLeft, verticalLineEndY, startZ, lineWidth);

verticalLine2Left = new LongVerticalLine(verticalLineStartXLeft - 100.0, verticalLineStartY, startZ, verticalLineEndXLeft - 100.0, verticalLineEndY, startZ, lineWidth);

doubleLine = new DoubleLine(startX, squareStartY - 50, startZ, endX, squareStartY - 50, endZ, lineWidth);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(windowWidth, windowHeight);

glutInitWindowPosition(100, 100);

glutCreateWindow("Desenare Linii Orizontale, Patrat si Linii Verticale");

init();

glutDisplayFunc(display);

glutMouseFunc(mousePressedFunc);

glutMotionFunc(mouseMovedFunc);

glutMainLoop();

delete line1;

delete line2;

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

delete square[i];

}

delete verticalLine1Right;

delete verticalLine2Right;

delete verticalLine1Left;

delete verticalLine2Left;

delete doubleLine;

return 0;

}

Funcția `main` este punctul de intrare al programului. Inițializează coordonatele și dimensiunile pentru liniile și pătratul ce urmează să fie desenate, creează obiectele respective și configurează fereastra GLUT. Apelează funcțiile `glutInit`, `glutInitDisplayMode`, `glutInitWindowSize`, `glutInitWindowPosition`, `glutCreateWindow` pentru a inițializa și crea fereastra grafică. Setează funcțiile de afișare și de gestionare a mouse-ului, apoi începe bucla principală GLUT cu `glutMainLoop`.

# Cod complet

#include <GL/glut.h>

class Line {

private:

float x1, y1, z1;

float x2, y2, z2;

float width;

public:

Line(float startX, float startY, float startZ, float endX, float endY, float endZ, float lineWidth)

: x1(startX), y1(startY), z1(startZ), x2(endX), y2(endY), z2(endZ), width(lineWidth) {}

void draw() const {

glLineWidth(width);

glBegin(GL\_LINES);

glVertex3f(x1, y1, z1);

glVertex3f(x2, y2, z2);

glEnd();

}

};

class ShortLine {

private:

float x1, y1, z1;

float x2, y2, z2;

float width;

public:

ShortLine(float startX, float startY, float startZ, float endX, float endY, float endZ, float lineWidth)

: x1(startX), y1(startY), z1(startZ), x2(endX), y2(endY), z2(endZ), width(lineWidth) {}

void draw() const {

glLineWidth(width);

glBegin(GL\_LINES);

glVertex3f(x1, y1, z1);

glVertex3f(x2, y2, z2);

glEnd();

}

};

class LongVerticalLine {

private:

float x1, y1, z1;

float x2, y2, z2;

float width;

public:

LongVerticalLine(float startX, float startY, float startZ, float endX, float endY, float endZ, float lineWidth)

: x1(startX), y1(startY), z1(startZ), x2(endX), y2(endY), z2(endZ), width(lineWidth) {}

void draw() const {

glLineWidth(width);

glBegin(GL\_LINES);

glVertex3f(x1, y1, z1);

glVertex3f(x2, y2, z2);

glEnd();

}

};

class DoubleLine {

private:

float x1, y1, z1;

float x2, y2, z2;

float x3, y3, z3;

float x4, y4, z4;

float width;

public:

DoubleLine(float startX, float startY, float startZ, float endX, float endY, float endZ, float lineWidth)

: x1(startX), y1(startY), z1(startZ), x2(endX), y2(endY), z2(endZ), width(lineWidth) {

y3 = startY;

z3 = startZ;

x3 = startX + 50.0;

y4 = endY;

z4 = endZ;

x4 = endX + 50.0;

}

void draw() const {

glLineWidth(width);

glBegin(GL\_LINES);

glVertex3f(x1, y1, z1);

glVertex3f(x2, y2, z2);

glVertex3f(x3, y3, z3);

glVertex3f(x4, y4, z4);

glEnd();

}

};

Line\* line1;

Line\* line2;

ShortLine\* square[4];

LongVerticalLine\* verticalLine1Right;

LongVerticalLine\* verticalLine2Right;

LongVerticalLine\* verticalLine1Left;

LongVerticalLine\* verticalLine2Left;

DoubleLine\* doubleLine;

int windowWidth = 800;

int windowHeight = 600;

bool mousePressed = false;

int lastMouseX, lastMouseY;

float offsetX = 0.0f, offsetY = 0.0f; float offsetZ = 0.0f;

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glPushMatrix();

glTranslatef(offsetX, offsetY, 0);

line1->draw();

line2->draw();

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

square[i]->draw();

}

verticalLine1Right->draw();

verticalLine2Right->draw();

verticalLine1Left->draw();

verticalLine2Left->draw();

glPopMatrix();

glFlush();

}

void init() {

glClearColor(0.0, 0.0, 0.0, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, windowWidth, 0.0, windowHeight);

}

void mousePressedFunc(int button, int state, int x, int y) {

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

mousePressed = true;

lastMouseX = x;

lastMouseY = y;

}

else if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_UP) {

mousePressed = false;

}

}

void mouseMovedFunc(int x, int y) {

if (mousePressed) {

offsetX += (x - lastMouseX);

offsetY -= (y - lastMouseY);

offsetZ += (y - lastMouseY);

lastMouseX = x;

lastMouseY = y;

glutPostRedisplay();

}

}

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

float startX = 100.0;

float startY = windowHeight - 100.0;

float startZ = 0.0;

float endX = 700.0;

float endY = windowHeight - 100.0;

float endZ = 0.0;

float lineWidth = 40.0;

line1 = new Line(startX, startY, startZ, endX, endY, endZ, lineWidth);

float secondLineStartY = startY - 50.0;

float secondLineEndY = endY - 50.0;

line2 = new Line(startX, secondLineStartY, startZ, endX, secondLineEndY, endZ, lineWidth);

float squareSize = 100.0;

float squareStartX = (startX + endX - squareSize) / 2;

float squareStartY = secondLineStartY - 150.0;

square[0] = new ShortLine(squareStartX, squareStartY, startZ, squareStartX + squareSize, squareStartY, startZ, lineWidth);

square[1] = new ShortLine(squareStartX + squareSize, squareStartY, startZ, squareStartX + squareSize, squareStartY + squareSize, startZ, lineWidth);

square[2] = new ShortLine(squareStartX + squareSize, squareStartY + squareSize, startZ, squareStartX, squareStartY + squareSize, startZ, lineWidth);

square[3] = new ShortLine(squareStartX, squareStartY + squareSize, startZ, squareStartX, squareStartY, startZ, lineWidth);

float verticalLineStartXRight = squareStartX + squareSize + 50.0;

float verticalLineStartY = squareStartY + squareSize;

float verticalLineEndXRight = verticalLineStartXRight;

float verticalLineEndY = verticalLineStartY - (2 \* squareSize) - 50.0;

verticalLine1Right = new LongVerticalLine(verticalLineStartXRight, verticalLineStartY, startZ, verticalLineEndXRight, verticalLineEndY, startZ, lineWidth);

verticalLine2Right = new LongVerticalLine(verticalLineStartXRight + 100.0, verticalLineStartY, startZ, verticalLineEndXRight + 100.0, verticalLineEndY, startZ, lineWidth);

float verticalLineStartXLeft = squareStartX - 50.0;

float verticalLineEndXLeft = verticalLineStartXLeft;

verticalLine1Left = new LongVerticalLine(verticalLineStartXLeft, verticalLineStartY, startZ, verticalLineEndXLeft, verticalLineEndY, startZ, lineWidth);

verticalLine2Left = new LongVerticalLine(verticalLineStartXLeft - 100.0, verticalLineStartY, startZ, verticalLineEndXLeft - 100.0, verticalLineEndY, startZ, lineWidth);

doubleLine = new DoubleLine(startX, squareStartY - 50, startZ, endX, squareStartY - 50, endZ, lineWidth);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(windowWidth, windowHeight);

glutInitWindowPosition(100, 100);

glutCreateWindow("Desenare Linii Orizontale, Patrat si Linii Verticale");

init();

glutDisplayFunc(display);

glutMouseFunc(mousePressedFunc);

glutMotionFunc(mouseMovedFunc);

glutMainLoop();

delete line1;

delete line2;

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

delete square[i];

}

delete verticalLine1Right;

delete verticalLine2Right;

delete verticalLine1Left;

delete verticalLine2Left;

delete doubleLine;

return 0;

}

# Rezultat

A black and white logo

Description automatically generated

A black and white logo

Description automatically generated