**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ**  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**  
**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №5

Компьютерная графика

**Алгоритм удаления невидимых граней**

Выполнил: ст. группы ВТ-31  
Ковалёв И. Д.

Проверил: Осипов О. В.

**Белгород 2019**

**Цель работы:**

Реализация простейшего алгоритма сортировки многоугольников по глубине.

**Задания для выполнения к работе**

1. Реализовать алгоритм сортировки выпуклых многоугольников по глубине.
2. На экран вывести только 1 проекцию модели (центральную или ортографическую). Пользователь должен иметь возможность поворачивать фигуру, с использованием мыши или клавиш, удалять и приближать фигуру.
3. Фигура должна состоять из набора небольших граней. Грани должны быть соизмеримы.
4. При выводе изображения предоставить возможность пользователю видеть грани прозрачными.

Содержимое файла Mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QPainter>

#include <QMouseEvent>

#include "object.h"

#include <QMatrix4x4>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

void paintEvent(QPaintEvent \* event);

void drawObject(QPainter \* painter);

private:

Ui::MainWindow \*ui;

bool debugFlag;

Cubes obj;

Cubes translatedObject;

int range, view;

double globalScale, centralViewPoint;

QMatrix4x4 rotationMatrix, M1, M\_t;

QPointF pos, paintingPolygonPos;

void mouseMoveEvent(QMouseEvent \* event);

void mousePressEvent(QMouseEvent \* event);

static bool polygonComp(CustomPolygon a, CustomPolygon b);

};

#endif // MAINWINDOW\_H

Содержимое файла Mainwindow.cpp:

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::**MainWindow**(QWidget \*parent) : QMainWindow(parent), ui(new Ui::MainWindow){

ui->setupUi(this);

obj = Cubes(1);

translatedObject = obj;

pos = QPointF(0, 0);

}

MainWindow::~***MainWindow***(){

delete ui;

}

QColor **getRandomColor**(){

return QColor(rand() % 255, rand() % 255, rand() % 255);

}

void MainWindow::***paintEvent***(QPaintEvent \*e){

QPainter painter(this);

drawObject(&painter);

}

void MainWindow::**drawObject**(QPainter \* painter){

QVector3D center = obj.getCenter();

float scale = 35;

QVector<QPolygonF> polygons;

QPolygonF polygon;

translatedObject = obj;

QVector4D v;

QMatrix4x4 M, M1, M\_t;

M1 = {1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1};

M.translate(-center);

M\_t.scale(scale, -scale, scale);

M = M.transposed() \* M\_t.transposed();

M = M \* rotationMatrix;

M\_t.setToIdentity();

//M\_t.translate(0, 0, globalScale);

M = M \* M\_t.transposed();

M\_t.setToIdentity();

M\_t.translate(ui->br->geometry().center().x(), ui->br->geometry().center().y(), 0);

M = M \* M\_t.transposed();

for (int i = 0; i < translatedObject.obj.size(); i++)

for(int j = 0; j < translatedObject.obj[i].getPolygon().size(); j++){

v.setX(translatedObject.obj[i].getPolygon()[j].x());

v.setY(translatedObject.obj[i].getPolygon()[j].y());

v.setZ(translatedObject.obj[i].getPolygon()[j].z());

v.setW(1);

v = v \* M;

QVector3D tmpPolygon;

tmpPolygon.setX(v.x() / v.w());

tmpPolygon.setY(v.y() / v.w());

tmpPolygon.setZ(v.z() / v.w());

translatedObject.obj[i].setVector(tmpPolygon, j);

}

std::stable\_sort(translatedObject.obj.begin(), translatedObject.obj.end(), polygonComp);

M.setToIdentity();

M.translate(-1 \* ui->br->geometry().center().x(), -1 \* ui->br->geometry().center().y(), 0);

M = M.transposed() \* M1.transposed();

M\_t.setToIdentity();

M\_t.translate(ui->br->geometry().center().x(), ui->br->geometry().center().y(), 0);

M = M \* M\_t.transposed();

for (int i = 0; i < translatedObject.obj.size(); i++){

for(int j = 0; j < translatedObject.obj[i].getPolygon().size(); j++){

v.setX(translatedObject.obj[i].getPolygon().at(j).x());

v.setY(translatedObject.obj[i].getPolygon().at(j).y());

v.setZ(translatedObject.obj[i].getPolygon().at(j).z());

v.setW(1);

v = v \* M;

polygon << QPointF(v.x()/v.w(), v.y()/v.w());

}

polygons << polygon;

polygon.clear();

}

QColor red = getRandomColor(); //Ударим по логике силой мысли!

red.setAlpha(150);

for (int i = polygons.size() - 1; i >= 0; i--){

if (polygons[i].containsPoint(paintingPolygonPos, Qt::FillRule::OddEvenFill)

&& !translatedObject.getObject()[i].isColored()){

qDebug() << "!!!" << i ;

paintingPolygonPos = QPointF(0, 0);

obj.obj[translatedObject.obj[i].getNumber()].setColor(red);

qDebug() << translatedObject.obj[i].getNumber() << i ;

}

}

for (int i = 0; i < translatedObject.obj.size(); i++){

painter->setBrush(translatedObject.obj[i].getColor());

painter->drawPolygon(polygons[i]);

}

}

bool MainWindow::**polygonComp**(CustomPolygon a, CustomPolygon b){

return a.getDepth() <= b.getDepth();

}

void MainWindow::***mouseMoveEvent***(QMouseEvent \* event){

double k = 5;

QPointF dp = event->pos() - pos;

pos = event->pos();

rotationMatrix.rotate(- dp.x() / k, QVector3D(0, 1, 0));

rotationMatrix.rotate(dp.y() / k, QVector3D(1, 0, 0));

repaint();

}

void MainWindow::***mousePressEvent***(QMouseEvent \* event){

pos = event->pos();

if (event->modifiers() == Qt::CTRL){

paintingPolygonPos = pos;

qDebug() << "poimal! :" << paintingPolygonPos;

}

repaint();

repaint();

}

Содержимое файла custompolygon.h:

#ifndef CUSTOMPOLYGON\_H

#define CUSTOMPOLYGON\_H

#include <QVector>

#include <QVector3D>

#include <QColor>

class **CustomPolygon**{

QVector <QVector3D> polygon;

QColor color;

float depth;

bool isClicked;

int number;

public:

**CustomPolygon**();

**CustomPolygon**(QVector<QVector3D>);

QVector <QVector3D> **getPolygon**();

float **getDepth**();

QColor **getColor**();

void **setVector**(QVector3D v, int pos);

void **setColor**(QColor);

void **setClicked**();

void **setNumber**(int);

int **getNumber**();

int **size**();

bool **isColored**(){return isClicked;}

QVector3D operator [](int i);

};

#endif // CUSTOMPOLYGON\_H

Содержимое файла custompolygon.сpp:

#include "custompolygon.h"

CustomPolygon::**CustomPolygon**(){

}

CustomPolygon::**CustomPolygon**(QVector<QVector3D> newPolygon){

polygon = newPolygon;

depth = 0;

for (int i = 0; i < newPolygon.size(); i++){

depth += newPolygon[i].z();

}

color = Qt::blue;

color.setAlpha(150);

isClicked = false;

}

QVector <QVector3D> CustomPolygon::**getPolygon**(){

return polygon;

}

float CustomPolygon::**getDepth**(){

depth = 0;

for (int i = 0; i < polygon.size(); i++){

depth += polygon[i].z();

}

return depth;

}

void CustomPolygon::**setVector**(QVector3D v, int pos){

polygon[pos] = v;

}

QColor CustomPolygon::**getColor**(){

return color;

}

void CustomPolygon::**setColor**(QColor c){

color = c;

}

int CustomPolygon::**size**(){

return polygon.size();

}

QVector3D CustomPolygon::operator [] (int i){

return polygon[i];

}

void CustomPolygon::**setClicked**(){

isClicked = true;

}

void CustomPolygon::**setNumber**(int n){

number = n;

}

int CustomPolygon::**getNumber**(){

return number;

}



