*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
|  | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(национальный исследовательский университет)***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_Компьютерные Системы и сети (ИУ6)\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по лабораторной работе № 10-11**

**Название лабораторной работы: Программирование с использованием библиотеки Qt**

**Дисциплина: Объектно-ориентированное программирование**

Студент гр. ИУ6-22Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. П. Плютто**



(Подпись, дата) (И.О. Фамилия)



Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. А. Веселовская**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

**Задание**

Сформировать файл, содержащий точки поверхности, образованной вращением функции y=sin(x) на диапазоне -pi/4…3/4 pi вокруг оси x. Вывести на экран поверхность в виде каркасной модели по прочитанным из файла значениям.

По щелчку мыши формировать метку, содержащую значения координаты, соответствующее положению метки. Обеспечить возможность перемещения меток. Обеспечить сохранение файла с координатами меток.

**Решение**

#include <QtDataVisualization>

#include <QFile>

#include <QObject>

#include <QTextStream>

#include <QtMath>

#include <QApplication>

#include <QTableWidget>

#include <QPushButton>

#include <QVBoxLayout>

#include <QHBoxLayout>

#include <QString>

#include <QLabel>

const double PI = 3.14159265358979323846;

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

{

QFile file("points.txt");

QVector3D vector;

bool flagSurf{}, flagper{};

int rowper{-1};

if (!file.open(QIODevice::WriteOnly | QIODevice::Text))

return 1;

QSurface3DSeries \*newseries = new QSurface3DSeries();

QTextStream out(&file);

QApplication app(argc, argv);

Q3DSurface surface;

surface.setFlags(surface.flags() ^ Qt::FramelessWindowHint);

QSurfaceDataArray \*data = new QSurfaceDataArray;

QSurfaceDataRow \*dataRow = new QSurfaceDataRow;

float x, y;

for (int i = -22; i <= 180; i++) {

x = i / 180.0 \* PI - PI/4;

y = qSin(x);

for (int j = 0; j < 721; j++) {

float angle = j / 180.0 \* PI;

float x\_rotated = x;

float y\_rotated = y \* qCos(angle);

float z\_rotated = y \* qSin(angle);

out << x\_rotated << " " << y\_rotated << " " << z\_rotated << "\n";

\*dataRow << QVector3D(x\_rotated, y\_rotated, z\_rotated);

}

\*data << dataRow;

dataRow = new QSurfaceDataRow;

}

file.close();

QSurface3DSeries \*series = new QSurface3DSeries();

series->setDrawMode(QSurface3DSeries::DrawSurface);

series->setFlatShadingEnabled(true);

series->setColorStyle(Q3DTheme::ColorStyleRangeGradient);

surface.addSeries(series);

series->dataProxy()->resetArray(data);

surface.axisX()->setRange(-1.5, 3);

surface.axisY()->setRange(-1.5, 3);

surface.axisZ()->setRange(-3, 6);

QLinearGradient gradient;

gradient.setColorAt(0.0, QColor(255, 0, 0)); // красный

gradient.setColorAt(0.2, QColor(255, 165, 0)); // оранжевый

gradient.setColorAt(0.3, QColor(255, 255, 0)); // желтый

gradient.setColorAt(0.4, QColor(0, 255, 0)); // зеленый

gradient.setColorAt(0.5, QColor(0, 191, 255)); // голубой

gradient.setColorAt(1.0, QColor(128, 0, 128)); // фиолетовый

series->setBaseGradient(gradient);/\*

series->setColorStyle(Q3DTheme::ColorStyleRangeGradient);

series->setDrawMode(QSurface3DSeries::DrawSurface);\*/

QTableWidget table;

table.setColumnCount(3);

table.setHorizontalHeaderLabels(QStringList() << "X" << "Y" << "Z");

QFile f1("p2.txt");

if (!f1.open(QIODevice::ReadOnly | QIODevice::Text))

return 1;

QTextStream in(&f1);

int row = 0;

while (!in.atEnd()) {

QString line = in.readLine();

QStringList fields = line.split(" ");

if (fields.size() < 3)

continue;

table.insertRow(row);

table.setItem(row, 0, new QTableWidgetItem(fields.at(0)));

table.setItem(row, 1, new QTableWidgetItem(fields.at(1)));

table.setItem(row, 2, new QTableWidgetItem(fields.at(2)));

row++;

}

file.close();

QPushButton button1("Выбрать");

QPushButton button2("Сохранить");

QPushButton button3("Переместить");

QLabel \*label = new QLabel(

"Справка: Для того, чтобы добавить метку кликнете на фигуру в месте, где необходима метка, затем нажмите\n"

"кнопку Выбрать. Для того, чтобы выбрать существующую метку выберете её в таблице, затем нажмите Выбрать.\n"

"Что бы переместить выбранную метку нажмите переместить, затем выберите новое расположение на моделе\n"

"и нажмите Выбрать. Чтобы сохранить метки нажмите кнопку сохранить."); // создание метки с текстом "Метка"

label->setFont(QFont("Arial", 12)); // настройка шрифта

label->setAlignment(Qt::AlignCenter); // настройка выравнивания текста

button1.setEnabled(0);

button3.setEnabled(0);

surface.resize(1000,1000);

QWidget mainWidget;

QVBoxLayout layoutwt(&mainWidget);

QHBoxLayout layout;

layoutwt.addWidget(label);

layoutwt.addLayout(&layout);

QWidget \*surfaceContainer = QWidget::createWindowContainer(&surface);

surfaceContainer->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);

layout.addWidget(surfaceContainer);

table.setFixedSize(400,800);

QVBoxLayout layout2;

layout.addLayout(&layout2);

layout2.addWidget(&table);

QHBoxLayout buttonLayout;

layout2.addLayout(&buttonLayout);

buttonLayout.addWidget(&button1);

buttonLayout.addWidget(&button2);

buttonLayout.addWidget(&button3);

layout.addLayout(&buttonLayout);

mainWidget.resize(1900,1000);

mainWidget.show();

QObject::connect(series, &QSurface3DSeries::selectedPointChanged, [&series, &flagSurf, &vector, &button1](const QPoint &pos)

{

if (pos != QPoint(-1,-1)){

vector = series->dataProxy()->itemAt(pos)->position();

flagSurf = 1;

} else

flagSurf = 0;

button1.setEnabled(flagSurf);

});

QObject::connect(&table, &QTableWidget::cellClicked, [&table, &flagSurf, &vector, &button1](int row, int column)

{

vector = QVector3D(table.item(row, 0)->text().toFloat(),

table.item(row, 1)->text().toFloat(),

table.item(row, 2)->text().toFloat());

flagSurf = 0;

button1.setEnabled(1);

});

QObject::connect(&button1, &QPushButton::clicked, [&vector, &surface, &newseries, &flagper, &rowper,

&button1, &button3, &flagSurf, &table]()

{

int n = 25;

int ut = 15;

float dx = vector[0],

dy = vector[1],

dz = vector[2];

QSurfaceDataRow \*dataRow = new QSurfaceDataRow;

QSurfaceDataArray \*data = new QSurfaceDataArray;

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

float theta = 2 \* PI \* i / (n-1);

for (int j = 0; j < n; j++) {

float phi = PI \* j / (n-1);

float x = qSin(phi) \* qCos(theta)/ut;

float y = qSin(phi) \* qSin(theta)/ut;

float z = qCos(phi)/ut;

\*dataRow << QVector3D(x+dx,y+dy,z+dz);

}

\*data << dataRow;

dataRow = new QSurfaceDataRow;

}

surface.addSeries(newseries);

newseries->dataProxy()->resetArray(data);

QLinearGradient gradient2;

gradient2.setColorAt(0.0, QColor(0, 191, 255)); // голубой

gradient2.setColorAt(1.0, QColor(128, 0, 128)); // фиолетовый

newseries->setBaseGradient(gradient2);

newseries->setColorStyle(Q3DTheme::ColorStyleRangeGradient);

newseries->setDrawMode(QSurface3DSeries::DrawSurface);

button1.setEnabled(0);

button3.setEnabled(1);

if (flagSurf){

if (!flagper){

int row = table.rowCount();

table.insertRow(row);

table.setItem(row, 0, new QTableWidgetItem(QString::number(vector[0])));

table.setItem(row, 1, new QTableWidgetItem(QString::number(vector[1])));

table.setItem(row, 2, new QTableWidgetItem(QString::number(vector[2])));

} else{

flagper = 0;

table.setItem(rowper, 0, new QTableWidgetItem(QString::number(vector[0])));

table.setItem(rowper, 1, new QTableWidgetItem(QString::number(vector[1])));

table.setItem(rowper, 2, new QTableWidgetItem(QString::number(vector[2])));

}

}

});

QObject::connect(&button2, &QPushButton::clicked, [&table]()

{

QFile f2("p2.txt");

if (f2.open(QIODevice::WriteOnly | QIODevice::Text)){

QTextStream out(&f2);

for(int i = 0; i < table.rowCount(); i++){

out << table.item(i, 0)->text() << " "

<< table.item(i, 1)->text() << " "

<< table.item(i, 2)->text() << "\n";

}

f2.close();

}

});

QObject::connect(&button3, &QPushButton::clicked, [&button1, &button3, &flagper, &rowper, &table, &vector]()

{

button1.setEnabled(0);

button3.setEnabled(0);

flagper = 1;

QList X = table.findItems(QString::number(vector[0]), Qt::MatchFlag(flagper));

QList Y = table.findItems(QString::number(vector[0]), Qt::MatchFlag(flagper));

QList Z = table.findItems(QString::number(vector[0]), Qt::MatchFlag(flagper));

rowper = -1;

for (auto x:X){

for (auto y:Y){

for (auto z:Z){

if (x->row() == y->row() == z->row()) rowper = z->row();

break;

}

if(rowper+1) break;

}

if(rowper+1) break;

}

});

surface.activeTheme()->setAmbientLightStrength(0.8);

surface.activeTheme()->setWindowColor(QColor::fromRgb(50,50,50));

surface.activeTheme()->setHighlightLightStrength(0);

surface.activeTheme()->setBackgroundEnabled(0);

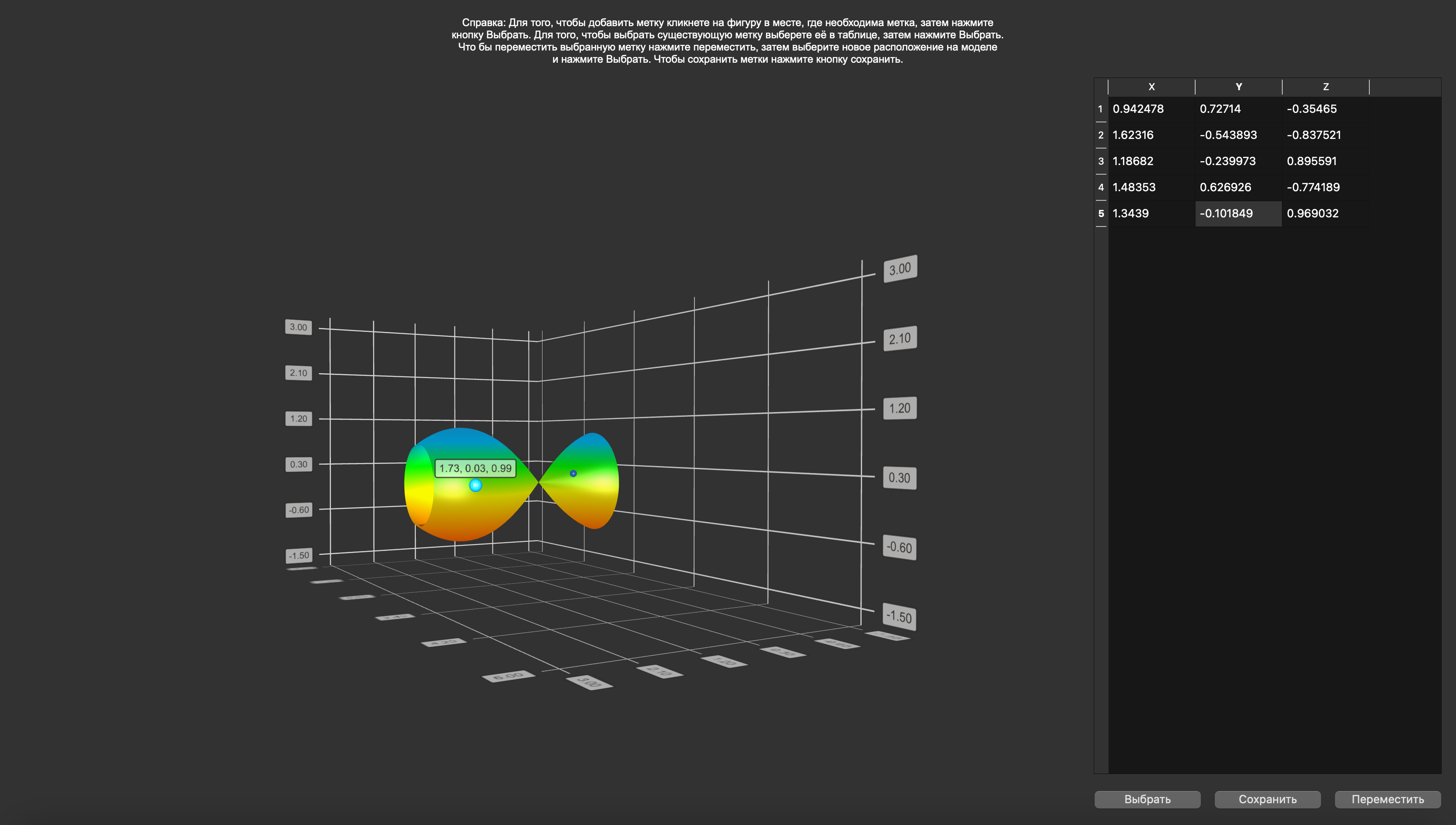
Q3DScene \*scene = surface.scene();

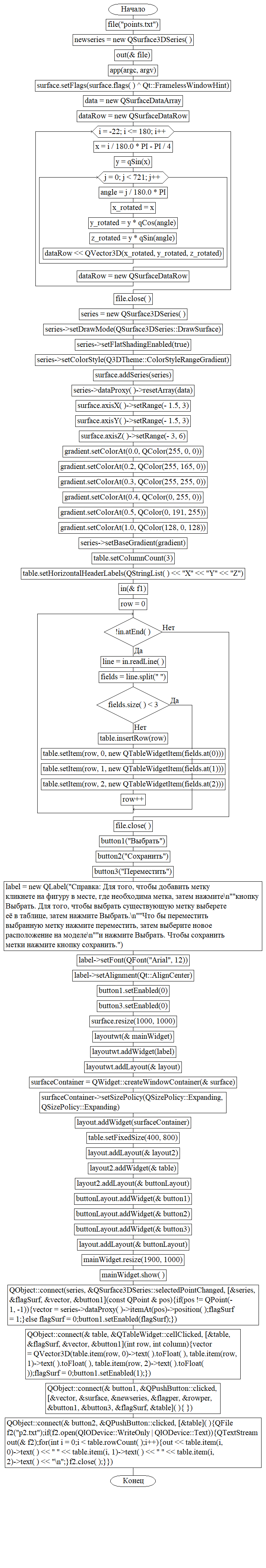
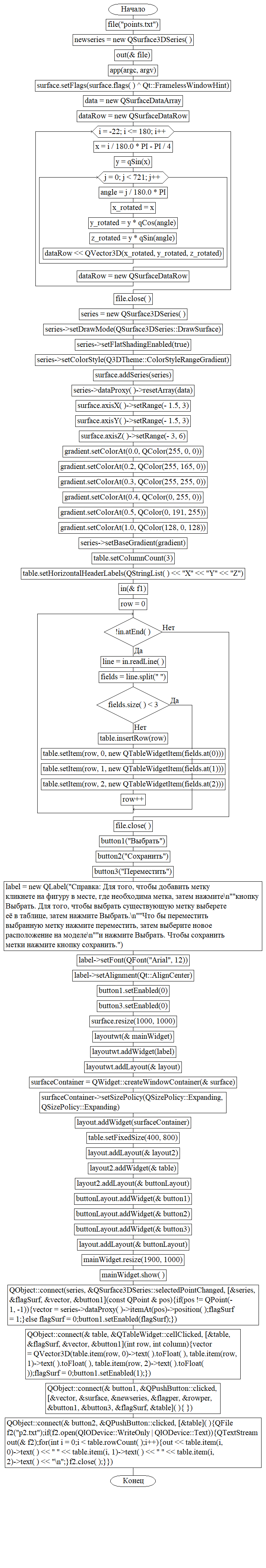
scene->activeCamera()->setCameraPreset(Q3DCamera::CameraPresetFront);

series->setFlatShadingEnabled(true);

return app.exec();

}





Вывод: Я научился использовать трехмерную графику в Qt