*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**

1. Создадим и откроем файл для записи данных, запишем все данные в таком порядке:
   1. 3 цифры через пробел – одна из точек каркасной модели
   2. Затем перенос строки для разделения точек
   3. ……….
   4. В конце цикла добавляем 1 и перенос строки для разделения “строк” точек

1. QFile file("points.txt");

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

3. return 1;

4. QTextStream out(&file);

5.

6. float x, y;

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

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

9. y = qSin(x);

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

11. float angle = j / 180.0 \* PI;

12. float x\_rotated = x;

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

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

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

16. }

17. out<<"1\n";

18. }

19.

20. file.close();

21.

22.

1. Создадим полотно и массивы данных, настроим полотно

1. Q3DSurface surface;

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

3. surface.resize(1000,1000);

4. QSurfaceDataArray \*data = new QSurfaceDataArray;

5. QSurfaceDataRow \*dataRow = new QSurfaceDataRow;

6.

1. Откроем созданный ранее файл и запишем все “строки” точек в dataRow, а затем dataRow запишем в data, тем самым образуя “двумерный массив” данных для создания каркасной модели.

1. if (!file.open(QIODevice::ReadOnly | QIODevice::Text))

2. return 1;

3. QTextStream inf(&file);

4.

5. while (!inf.atEnd()) {

6. QString line = inf.readLine();

7. QStringList coordinates = line.split(" ");

8. if(coordinates.size() == 1){

9. \*data << dataRow;

10. dataRow = new QSurfaceDataRow;

11. } else if (coordinates.size() == 3){

12. float x\_rotated = coordinates[0].toFloat();

13. float y\_rotated = coordinates[1].toFloat();

14. float z\_rotated = coordinates[2].toFloat();

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

16. }

17. }

18.

19. file.close();

20.

1. Создадим насколько переменных для lambda-функций.

1. QVector3D vector;

2. bool flagSurf{}, flagper{};

3. int rowper{-1};

4.

1. Создадим 2 каркасные модели одну для синусойды и вторую для сферы-метки и настроим их.

1. QSurface3DSeries \*series = new QSurface3DSeries();

2. QSurface3DSeries \*newseries = new QSurface3DSeries();

3.

4. series->setDrawMode(QSurface3DSeries::DrawSurface);

5. series->setFlatShadingEnabled(true);

6. series->setColorStyle(Q3DTheme::ColorStyleRangeGradient);

7. surface.addSeries(series);

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

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

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

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

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

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

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

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

16.

17. QLinearGradient gradient;

18.

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

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

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

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

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

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

25.

26. series->setBaseGradient(gradient);

27.

1. Создадим таблицу для сохранения выбранных меток и настроим ее

1. QTableWidget table;

2.

3. table.setColumnCount(3);

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

5. table.setFixedSize(400,800);

6.

1. Откроем файл для записи меток и занесем все сохраненные раннее метки в таблицу

1. QFile f1("p2.txt");

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

3. return 1;

4. QTextStream in(&f1);

5.

6. int row = 0;

7. while (!in.atEnd()) {

8. QString line = in.readLine();

9. QStringList fields = line.split(" ");

10. if (fields.size() < 3)

11. continue;

12. table.insertRow(row);

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

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

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

16. row++;

17. }

18.

19. file.close();

20.

1. Создадим и настроим кнопки для меток, а также напишем небольшую справку.

1. QPushButton button1("Выбрать");

2. QPushButton button2("Сохранить");

3. QPushButton button3("Переместить");

4. QLabel \*label = new QLabel(

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

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

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

8. "и нажмите Выбрать. Чтобы сохранить метки нажмите кнопку сохранить.");

9.

10. label->setFont(QFont("Arial", 12));

11. label->setAlignment(Qt::AlignCenter);

12. button1.setEnabled(0);

13. button3.setEnabled(0);

14.

1. Расположим все элементы, создав layoutы

1. QWidget mainWidget;

2. QVBoxLayout layoutwt(&mainWidget);

3. QHBoxLayout layout;

4.

5. layoutwt.addWidget(label);

6. layoutwt.addLayout(&layout);

7.

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

9.

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

11.

12. layout.addWidget(surfaceContainer);

13.

14. QVBoxLayout layout2;

15.

16. layout.addLayout(&layout2);

17. layout2.addWidget(&table);

18.

19. QHBoxLayout buttonLayout;

20.

21. layout2.addLayout(&buttonLayout);

22. buttonLayout.addWidget(&button1);

23. buttonLayout.addWidget(&button2);

24. buttonLayout.addWidget(&button3);

25. layout.addLayout(&buttonLayout);

26.

27. mainWidget.resize(1900,1000);

28. mainWidget.show();

29.

30.

1. Напишем обработчик сигналов нажатий на кнопки

1. // Сигнал: выбрана точка на каркасной модели

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

3. {

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

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

6. flagSurf = 1;

7. } else

8. flagSurf = 0;

9. button1.setEnabled(flagSurf);

10. });

11.

12. // Сигнал: выбрана ячейка таблицы

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

14. {

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

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

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

18. flagSurf = 0;

19. button1.setEnabled(1);

20. });

21.

22. // Сигнал: нажата кнопка выбрать

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

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

25. {

26. int n = 25;

27. int ut = 15;

28. float dx = vector[0],

29. dy = vector[1],

30. dz = vector[2];

31. QSurfaceDataRow \*dataRow = new QSurfaceDataRow;

32. QSurfaceDataArray \*data = new QSurfaceDataArray;

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

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

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

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

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

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

39. float z = qCos(phi)/ut;

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

41. }

42. \*data << dataRow;

43. dataRow = new QSurfaceDataRow;

44. }

45. surface.addSeries(newseries);

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

47. QLinearGradient gradient2;

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

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

50. newseries->setBaseGradient(gradient2);

51. newseries->setColorStyle(Q3DTheme::ColorStyleRangeGradient);

52. newseries->setDrawMode(QSurface3DSeries::DrawSurface);

53. button1.setEnabled(0);

54. button3.setEnabled(1);

55. if (flagSurf){

56. if (!flagper){

57. int row = table.rowCount();

58. table.insertRow(row);

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

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

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

62. } else{

63. flagper = 0;

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

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

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

67. }

68. }

69. });

70.

71. // Сигнал: нажата кнопка сохранить

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

73. {

74. QFile f2("p2.txt");

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

76. QTextStream out(&f2);

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

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

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

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

81. }

82. f2.close();

83. }

84. });

85.

86. // Сигнал: нажата кнопка переместить

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

88. {

89. button1.setEnabled(0);

90. button3.setEnabled(0);

91. flagper = 1;

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

93. QList Y = table.findItems(QString::number(vector[1]), Qt::MatchFlag(flagper));

94. QList Z = table.findItems(QString::number(vector[2]), Qt::MatchFlag(flagper));

95. rowper = -1;

96. for (auto x:X){

97. for (auto y:Y){

98. for (auto z:Z){

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

100. break;

101. }

102. if(rowper+1) break;

103. }

104. if(rowper+1) break;

105. }

106. });

107.

1. #include <QtDataVisualization>

2. #include <QFile>

3. #include <QObject>

4. #include <QTextStream>

5. #include <QtMath>

6. #include <QApplication>

7. #include <QTableWidget>

8. #include <QPushButton>

9. #include <QVBoxLayout>

10. #include <QHBoxLayout>

11. #include <QString>

12. #include <QLabel>

13.

14. const double PI = 3.14159265358979323846;

15.

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

17. {

18. QApplication app(argc, argv);

19.

20. // Открываем файл на запись и записываем данные

21. QFile file("points.txt");

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

23. return 1;

24. QTextStream out(&file);

25.

26. float x, y;

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

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

29. y = qSin(x);

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

31. float angle = j / 180.0 \* PI;

32. float x\_rotated = x;

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

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

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

36. }

37. out<<"1\n";

38. }

39.

40. file.close();

41.

42. // Создаем полотно и массивы данных

43. Q3DSurface surface;

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

45. surface.resize(1000,1000);

46. QSurfaceDataArray \*data = new QSurfaceDataArray;

47. QSurfaceDataRow \*dataRow = new QSurfaceDataRow;

48.

49. // Открываем файлы для чтения и записываем все в массивы,

50. // причем при нахождении 1 записываем плостость в основной массив

51. if (!file.open(QIODevice::ReadOnly | QIODevice::Text))

52. return 1;

53. QTextStream inf(&file);

54.

55. while (!inf.atEnd()) {

56. QString line = inf.readLine();

57. QStringList coordinates = line.split(" ");

58. if(coordinates.size() == 1){

59. \*data << dataRow;

60. dataRow = new QSurfaceDataRow;

61. } else if (coordinates.size() == 3){

62. float x\_rotated = coordinates[0].toFloat();

63. float y\_rotated = coordinates[1].toFloat();

64. float z\_rotated = coordinates[2].toFloat();

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

66. }

67. }

68.

69. file.close();

70.

71. // Создаем глобальные переменные для будущих функций-слотов

72. QVector3D vector;

73. bool flagSurf{}, flagper{};

74. int rowper{-1};

75.

76. // Создаем 2 каркасные модели: одну для синусойды и вторую для сферы-метки

77. QSurface3DSeries \*series = new QSurface3DSeries();

78. QSurface3DSeries \*newseries = new QSurface3DSeries();

79.

80. // Настраиваем каркасную модель и полотно

81. series->setDrawMode(QSurface3DSeries::DrawSurface);

82. series->setFlatShadingEnabled(true);

83. series->setColorStyle(Q3DTheme::ColorStyleRangeGradient);

84. surface.addSeries(series);

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

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

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

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

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

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

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

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

93.

94. // Создаем градиент для каркасной модели

95. QLinearGradient gradient;

96.

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

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

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

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

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

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

103.

104. series->setBaseGradient(gradient);

105.

106. //-----------------------конец 10 лабораторной--------------------

107.

108. // Создаем таблицу для меток

109. QTableWidget table;

110.

111. table.setColumnCount(3);

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

113. table.setFixedSize(400,800);

114.

115. // Открываем файлик меток, читаем его и заносим значения в таблицу

116. QFile f1("p2.txt");

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

118. return 1;

119. QTextStream in(&f1);

120.

121. int row = 0;

122. while (!in.atEnd()) {

123. QString line = in.readLine();

124. QStringList fields = line.split(" ");

125. if (fields.size() < 3)

126. continue;

127. table.insertRow(row);

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

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

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

131. row++;

132. }

133.

134. file.close();

135.

136. // Создаем и настраиваем кнопки, дополнительно пишем справку по работе с моделью

137. QPushButton button1("Выбрать");

138. QPushButton button2("Сохранить");

139. QPushButton button3("Переместить");

140. QLabel \*label = new QLabel(

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

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

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

144. "и нажмите Выбрать. Чтобы сохранить метки нажмите кнопку сохранить.");

145.

146. label->setFont(QFont("Arial", 12));

147. label->setAlignment(Qt::AlignCenter);

148. button1.setEnabled(0);

149. button3.setEnabled(0);

150.

151. // Создаем главный виджет и располагаем на нем все элементы

152. QWidget mainWidget;

153. QVBoxLayout layoutwt(&mainWidget);

154. QHBoxLayout layout;

155.

156. layoutwt.addWidget(label);

157. layoutwt.addLayout(&layout);

158.

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

160.

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

162.

163. layout.addWidget(surfaceContainer);

164.

165. QVBoxLayout layout2;

166.

167. layout.addLayout(&layout2);

168. layout2.addWidget(&table);

169.

170. QHBoxLayout buttonLayout;

171.

172. layout2.addLayout(&buttonLayout);

173. buttonLayout.addWidget(&button1);

174. buttonLayout.addWidget(&button2);

175. buttonLayout.addWidget(&button3);

176. layout.addLayout(&buttonLayout);

177.

178. mainWidget.resize(1900,1000);

179. mainWidget.show();

180.

181. // Сигнал: выбрана точка на каркасной модели

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

183. {

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

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

186. flagSurf = 1;

187. } else

188. flagSurf = 0;

189. button1.setEnabled(flagSurf);

190. });

191.

192. // Сигнал: выбрана ячейка таблицы

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

194. {

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

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

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

198. flagSurf = 0;

199. button1.setEnabled(1);

200. });

201.

202. // Сигнал: нажата кнопка выбрать

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

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

205. {

206. int n = 25;

207. int ut = 15;

208. float dx = vector[0],

209. dy = vector[1],

210. dz = vector[2];

211. QSurfaceDataRow \*dataRow = new QSurfaceDataRow;

212. QSurfaceDataArray \*data = new QSurfaceDataArray;

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

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

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

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

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

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

219. float z = qCos(phi)/ut;

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

221. }

222. \*data << dataRow;

223. dataRow = new QSurfaceDataRow;

224. }

225. surface.addSeries(newseries);

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

227. QLinearGradient gradient2;

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

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

230. newseries->setBaseGradient(gradient2);

231. newseries->setColorStyle(Q3DTheme::ColorStyleRangeGradient);

232. newseries->setDrawMode(QSurface3DSeries::DrawSurface);

233. button1.setEnabled(0);

234. button3.setEnabled(1);

235. if (flagSurf){

236. if (!flagper){

237. int row = table.rowCount();

238. table.insertRow(row);

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

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

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

242. } else{

243. flagper = 0;

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

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

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

247. }

248. }

249. });

250.

251. // Сигнал: нажата кнопка сохранить

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

253. {

254. QFile f2("p2.txt");

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

256. QTextStream out(&f2);

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

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

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

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

261. }

262. f2.close();

263. }

264. });

265.

266. // Сигнал: нажата кнопка переместить

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

268. {

269. button1.setEnabled(0);

270. button3.setEnabled(0);

271. flagper = 1;

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

273. QList Y = table.findItems(QString::number(vector[1]), Qt::MatchFlag(flagper));

274. QList Z = table.findItems(QString::number(vector[2]), Qt::MatchFlag(flagper));

275. rowper = -1;

276. for (auto x:X){

277. for (auto y:Y){

278. for (auto z:Z){

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

280. break;

281. }

282. if(rowper+1) break;

283. }

284. if(rowper+1) break;

285. }

286. });

287. Q3DScene \*scene = surface.scene();

288.

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

290. series->setFlatShadingEnabled(true);

291. return app.exec();

292. }

293.

294.

295.

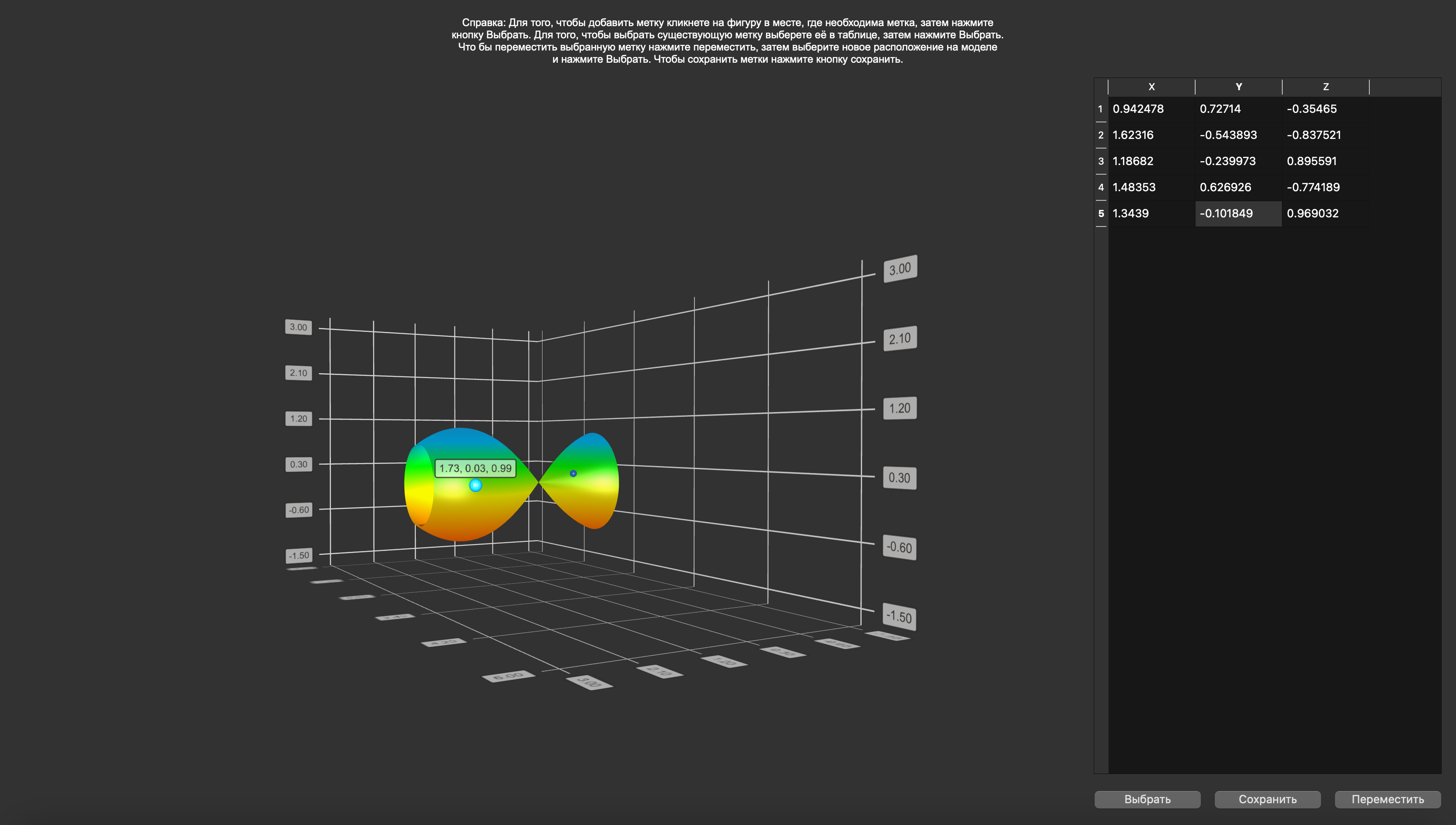


Рисунок : Работа программы

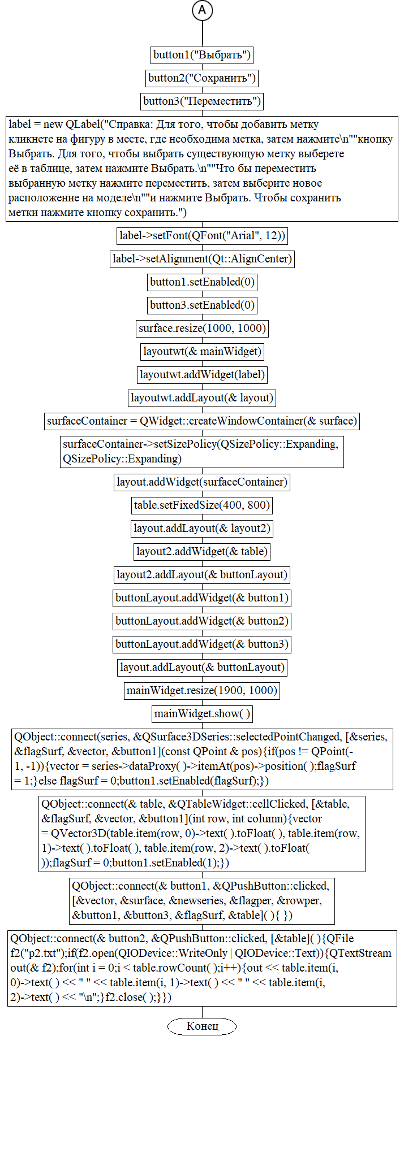
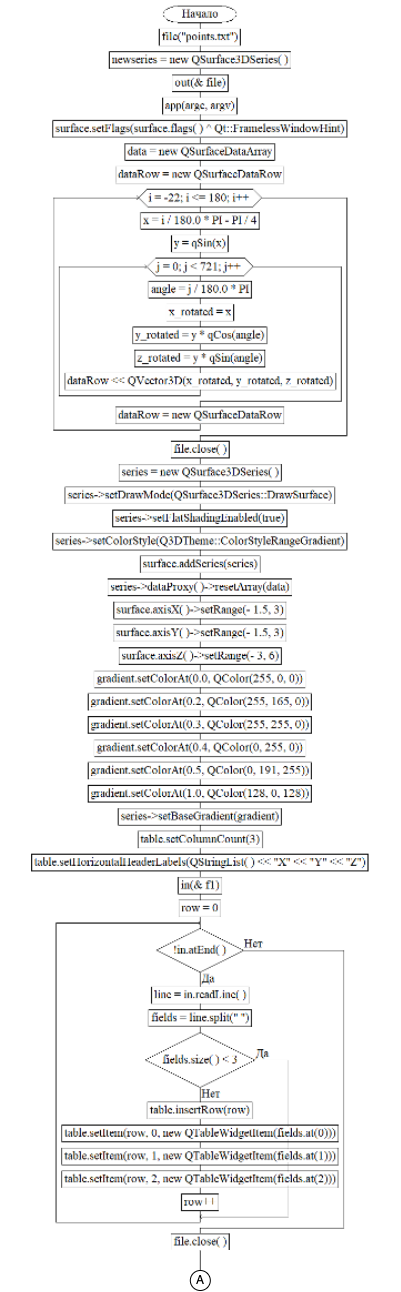


Рисунок : Схема алгоритма для кода программы

Вывод: Были изучены программирование операций с файлами и интерактивная графика в С++.