Московский авиационный институт
(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа №2

по курсу «Компьютерная графика»
Тема: «Каркасная визуализация выпуклого многогранника.
Удаление невидимых линий»

Студент: Мариничев И. А.
Группа: М8О-308Б-19
Преподаватель: Филиппов Г. С.
Оценка:

Москва
2021

1. Постановка задачи.

Разработать формат представления многогранника и процедуру его каркасной отрисовки в ортографической и изометрической проекциях. Обеспечить удаление невидимых линий и возможность пространственных поворотов и масштабирования многогранника. Обеспечить автоматическое центрирование и изменение размеров изображения при изменении размеров окна.

Вариант №8: 5–гранная прямая правильная призма

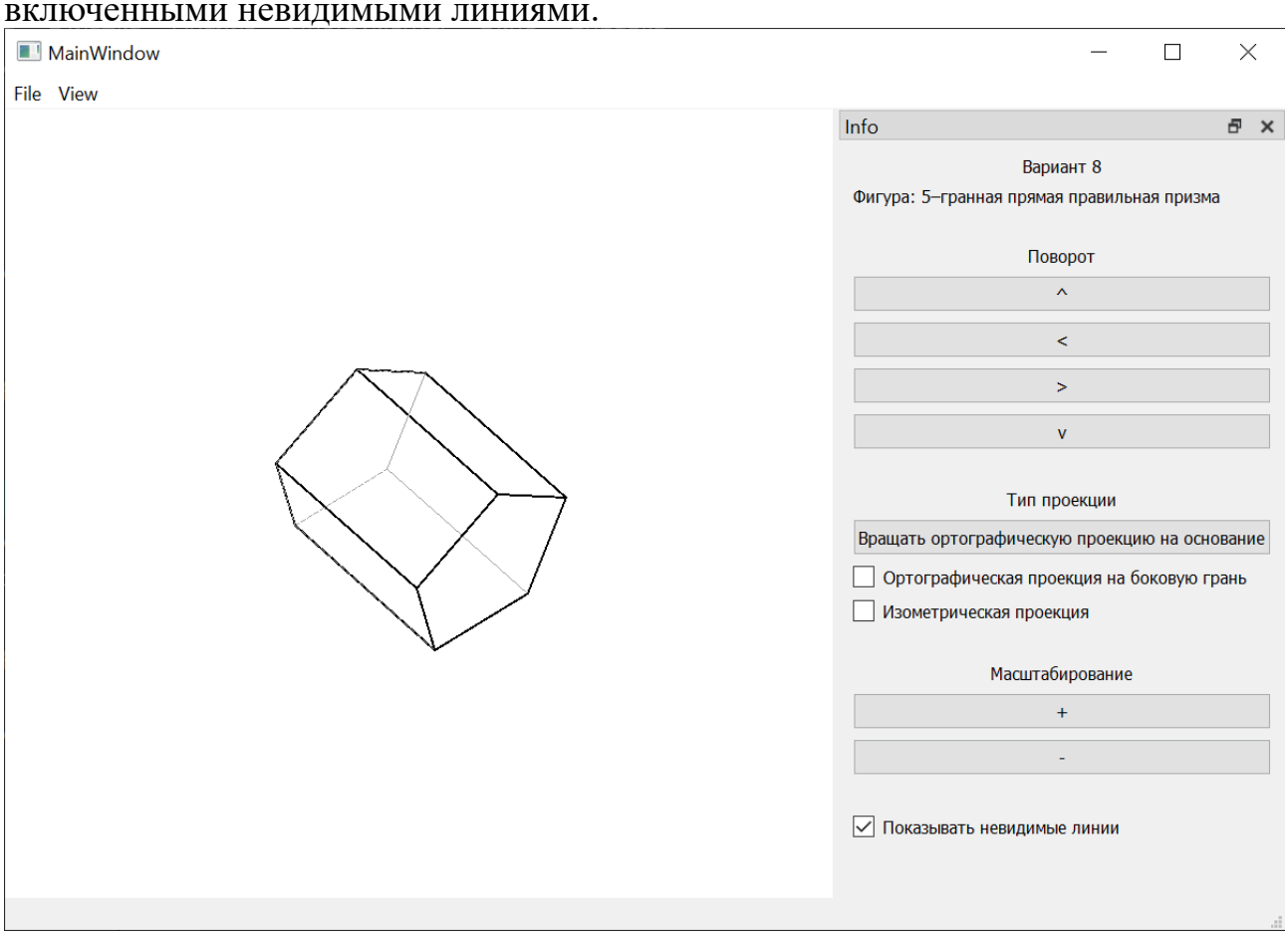
2. Описание программы.

Для решения задачи я решил использовать C++ и фреймворк Qt, в котором использовал библиотеку QPainter.

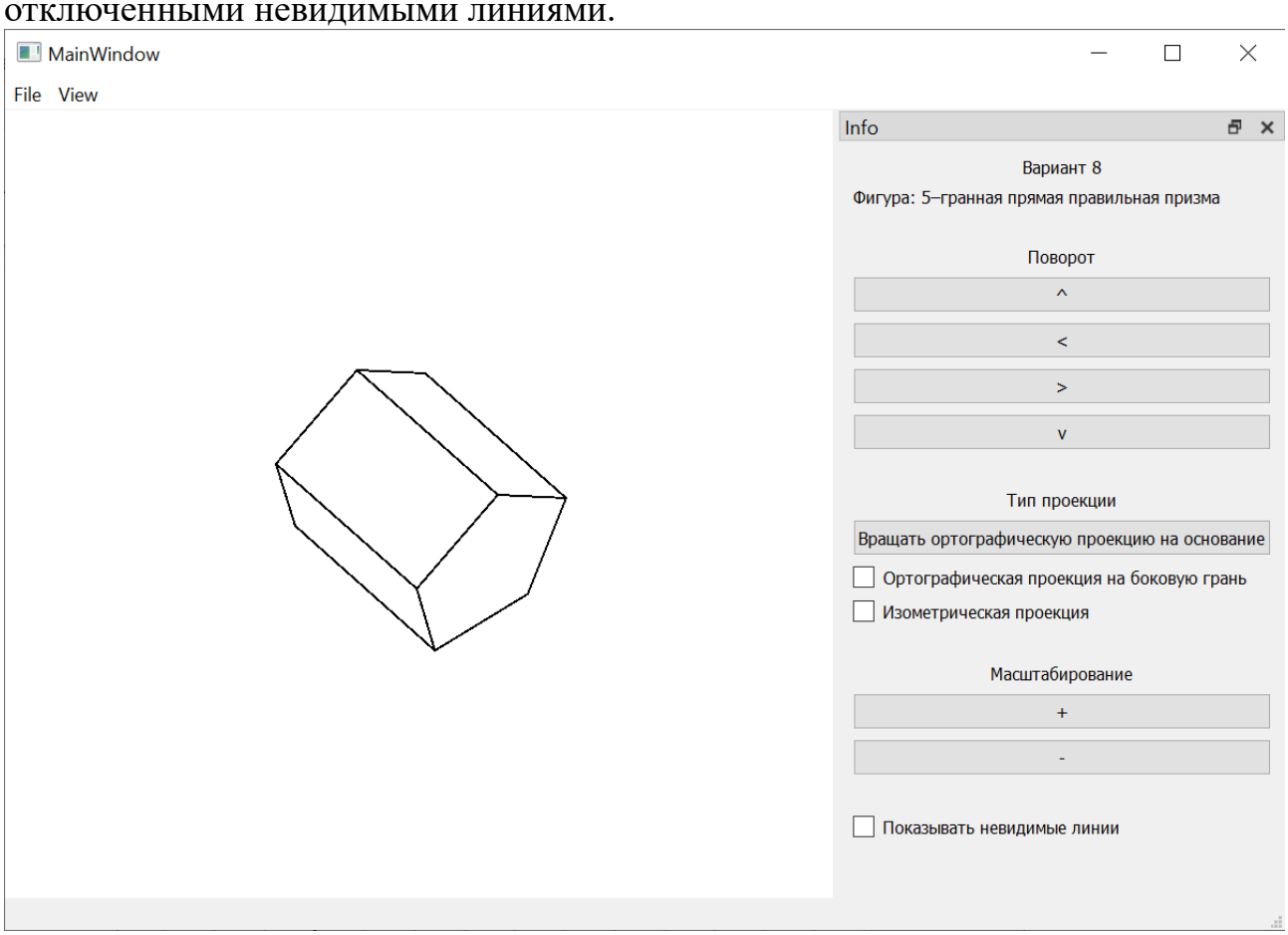
Я создал класс `polygon` для хранения полигонов, класс `prism`, представляющий фигуру пятигранная прямая призма. Такая фигура состоит из семи полигонов. Все преобразования для фигуры выполняются для каждого полигона, и в каждом полигоне преобразования выполняются для каждой точки. Так выполняются пространственные повороты фигуры и масштабирование фигуры.

3. Демонстрация работы программы.

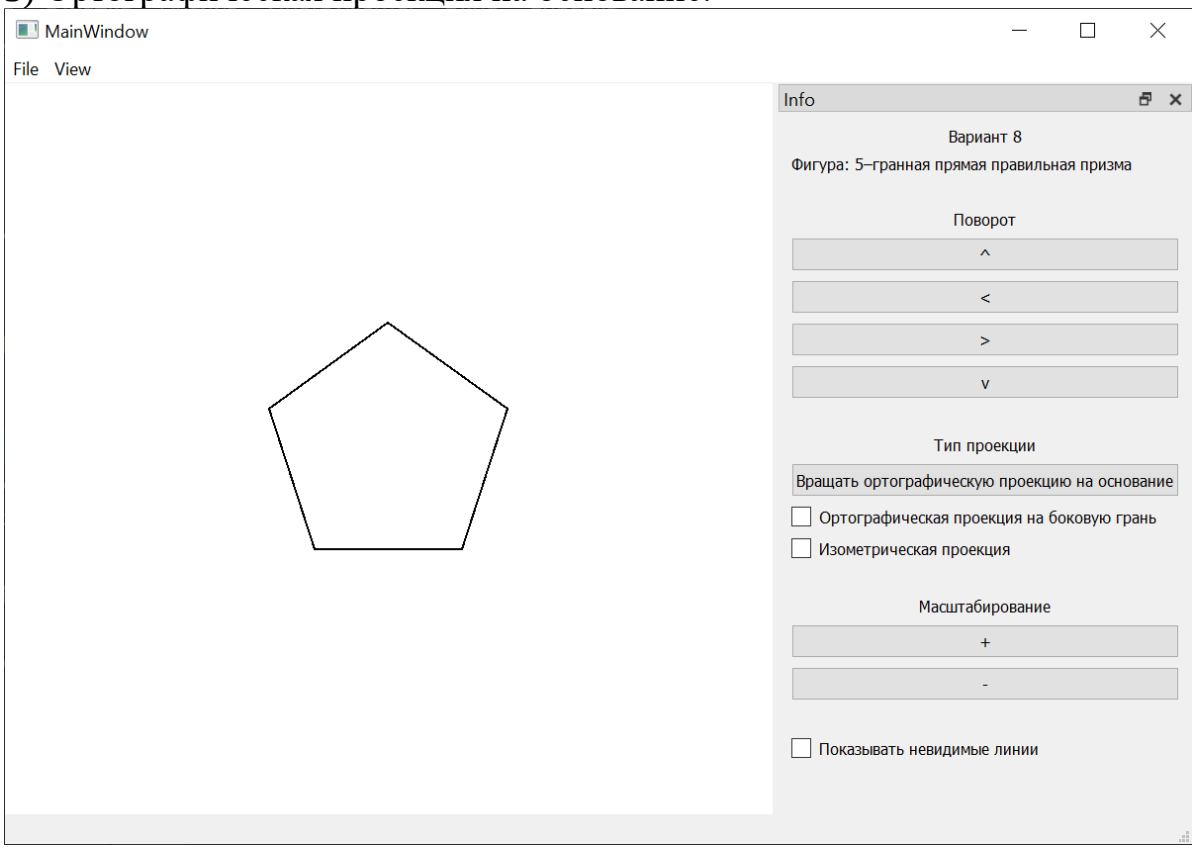
1) Вид фигуры после нескольких поворотов, масштабирования и с включенными невидимыми линиями.



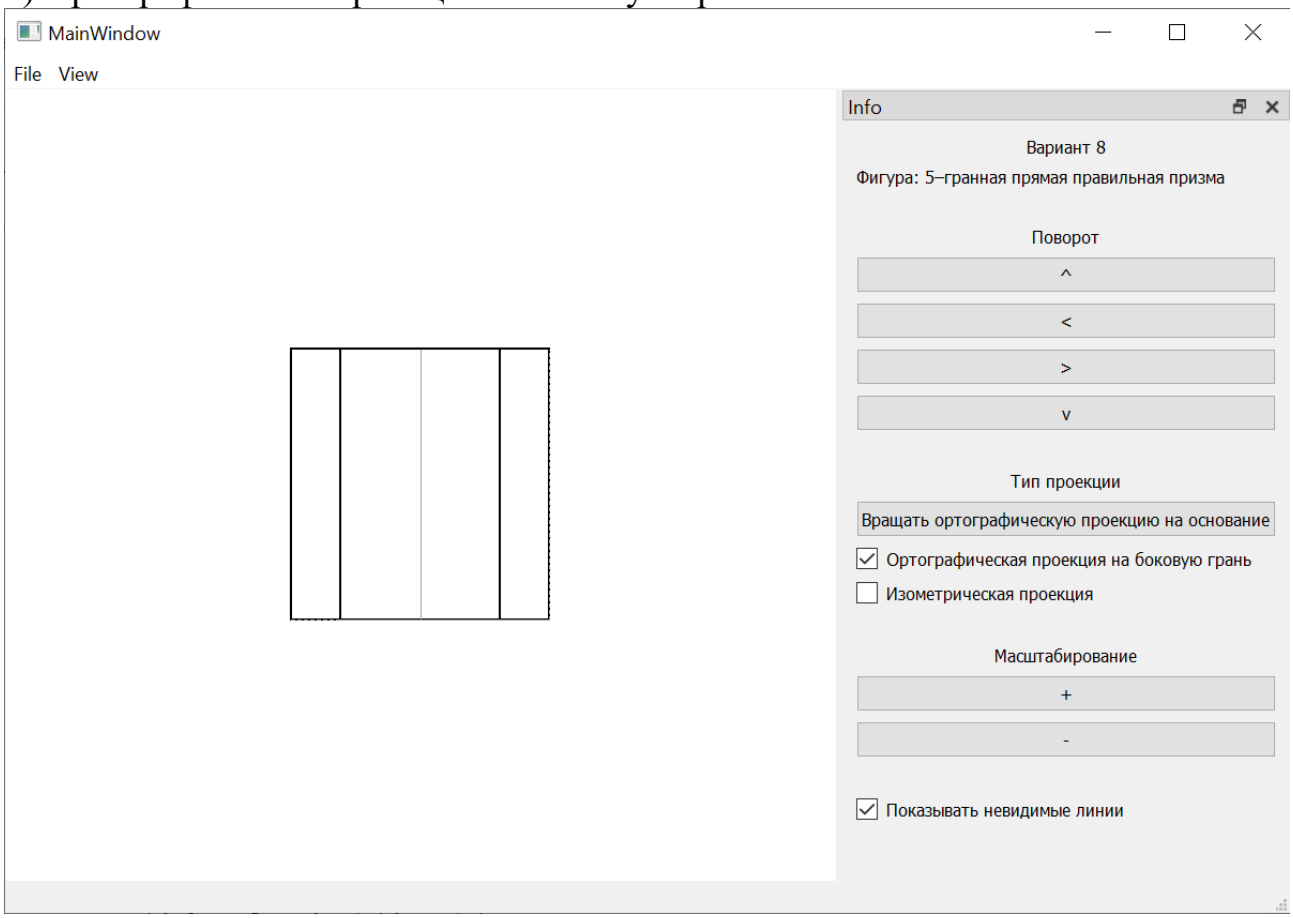
2) То же расположение фигуры, с тем же масштабированием, но с отключенными невидимыми линиями.



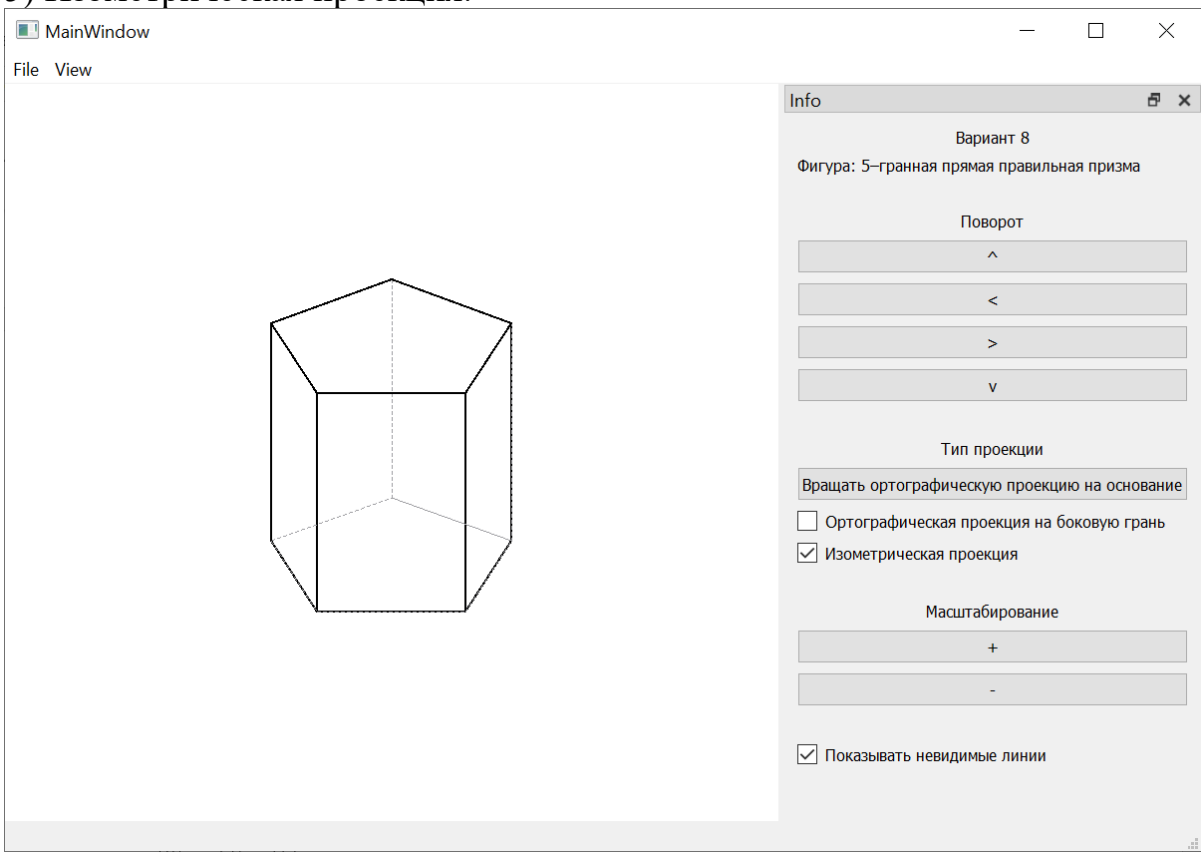
3) Ортографическая проекция на основание.



4) Ортографическая проекция на боковую грань.



5) Изометрическая проекция.



4. Основной код программы.

1) Основные методы класса polygon

```
vector<double> polygon::get_normal() {
    vector<double> first = {
        verticies[1][0] - verticies[0][0],
        verticies[1][1] - verticies[0][1],
        verticies[1][2] - verticies[0][2]
    };
    vector<double> second = {
        verticies[verticies.size() - 1][0] - verticies[0][0],
        verticies[verticies.size() - 1][1] - verticies[0][1],
        verticies[verticies.size() - 1][2] - verticies[0][2]
    };
    vector<double> normal = {
        first[1] * second[2] - second[1] * first[2],
        second[0] * first[2] - first[0] * second[2],
        first[0] * second[1] - second[0] * first[1]
    };

    return normal;
}

void polygon::draw(QPainter *ptr, int center_x, int center_y) {
    for (size_t i = 0; i < verticies.size() - 1; i++) {
        ptr->drawLine(static_cast<int>(verticies[i][0] + center_x),
                      static_cast<int>(verticies[i][1] + center_y),
                      static_cast<int>(verticies[i + 1][0] + center_x),
                      static_cast<int>(verticies[i + 1][1] + center_y));
    }
    ptr->drawLine(static_cast<int>(verticies[0][0] + center_x),
                  static_cast<int>(verticies[0][1] + center_y),
                  static_cast<int>(verticies[verticies.size()    -    1][0]    +
center_x),
                  static_cast<int>(verticies[verticies.size()    -    1][1]    +
center_y));
}
```

2) Основные методы класса prism

```
void prism::draw(QPainter *ptr, int center_x, int center_y) {
    for (auto p : polygons) {
        auto p_normal = p.get_normal();
        if (p_normal[2] > 0) { // if z coordinate of normal is greater than zero
            p.draw(ptr, center_x, center_y);
        } else if (displayHiddenLines) {
            QPen new_pen(Qt::gray, 1, Qt::DashLine);
            QPen old_pen = ptr->pen();
            ptr->setPen(new_pen);
            p.draw(ptr, center_x, center_y);
            ptr->setPen(old_pen);
        }
    }
}
```

3) Изменение масштаба фигуры

```
void MainWindow::zoom_in() {
    double scaling_coefficient = 1.05;
    double coef_x = scaling_coefficient;
    double coef_y = scaling_coefficient;
    double coef_z = scaling_coefficient;
    vector<vector<double>> matrix_s{
        vector<double>{coef_x, 0, 0, 0},
        vector<double>{0, coef_y, 0, 0},
        vector<double>{0, 0, coef_z, 0},
```

```
        vector<double>{0, 0, 0, 1}
    };
    display_ptr->get_prism().change_all_polygons(matrix_s);

    display_ptr->update();
}

void MainWindow::zoom_out() {
    double scaling_coefficient = 0.95;
    double coef_x = scaling_coefficient;
    double coef_y = scaling_coefficient;
    double coef_z = scaling_coefficient;
    vector<vector<double>> matrix_s{
        vector<double>{coef_x, 0, 0, 0},
        vector<double>{0, coef_y, 0, 0},
        vector<double>{0, 0, coef_z, 0},
        vector<double>{0, 0, 0, 1}
    };
    display_ptr->get_prism().change_all_polygons(matrix_s);

    display_ptr->update();
}
```

4) Функии поворотов фигуры

```
void MainWindow::turn_right() {
    vector<vector<double>> matrix_Ry{
        vector<double>{cos(M_PI / 12.), 0, -sin(M_PI / 12.), 0},
        vector<double>{0, 1, 0, 0},
        vector<double>{sin(M_PI / 12.), 0, cos(M_PI / 12.), 0},
        vector<double>{0, 0, 0, 1}
    };

    display_ptr->get_prism().change_all_polygons(matrix_Ry);
    display_ptr->update();
}

void MainWindow::turn_up() {
    vector<vector<double>> matrix_Rx{
        vector<double>{1, 0, 0, 0},
        vector<double>{0, cos(M_PI / 12.), sin(M_PI / 12.), 0},
        vector<double>{0, -sin(M_PI / 12.), cos(M_PI / 12.), 0},
        vector<double>{0, 0, 0, 1}
    };

    display_ptr->get_prism().change_all_polygons(matrix_Rx);
    display_ptr->update();
}
```

5. Выводы.

В ходе данной лабораторной работы я смог отрисовать трехмерную каркасную модель, обеспечил ее повороты в пространстве, масштабирование и возможность скрывать/показывать невидимые линии. Для реализации данного функционала я применил теоретические знания, полученные в курсе линейной алгебры.