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»**

|  |  |
| --- | --- |
| Студент: | Аксенов А.Е. |
| Группа: | М80-308Б-18 |
| Преподаватель: | Филиппов Г.С. |
| Оценка: |  |
| Дата: |  |

Москва

2020

**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.

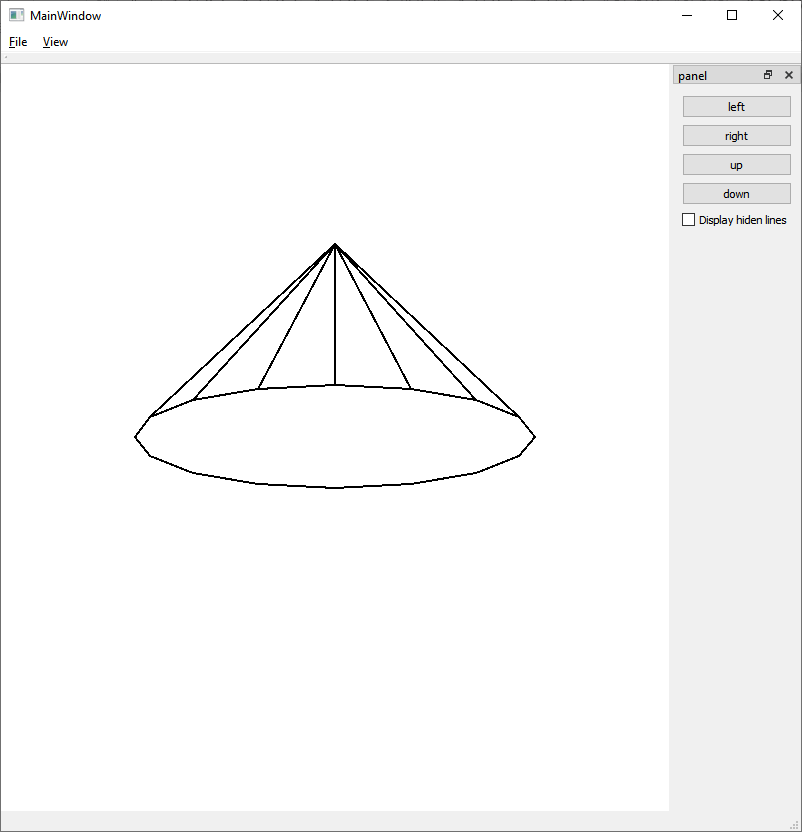
Вариант №16: 16 – гранная прямая правильная пирамида.

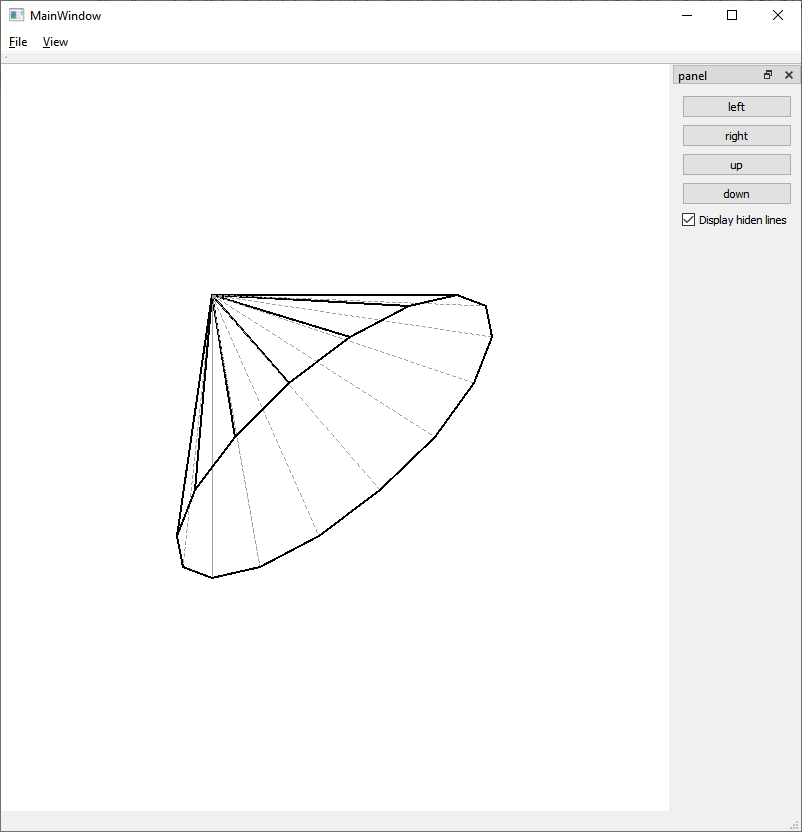
**2. Решение задачи.**

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

Я создал класс Polygon для хранения полигонов, класс Pyramid, представляющий фигуру 16 – гранной пирамиды. Такая фигура состоит из семнадцати полигонов. Все преобразования для фигуры выполняются для каждой полигона, и в каждом полигоне преобразования выполняются для каждой точки. Так выполняются пространственные повороты фигуры и масштабирование фигуры.

**3. Демонстрация работы программы.**

****



**4. Листинг программы.**

**polygon.h:**

#ifndef POLYGON\_H

#define POLYGON\_H

#include <vector>

#include <QPainter>

class **Polygon**

{

std::vector<std::vector<double>> verticies;

public:

**Polygon**();

**Polygon**(const std::vector<std::vector<double>> &v);

std::vector<double> **get\_normal**();

void **change\_verticies**(const std::vector<std::vector<double>> &v);

void **add\_vertex**(const std::vector<double> &v);

void **add\_vertex**(double x, double y, double z, double d);

void **clear\_verticies**();

void **draw**(QPainter \*ptr, int center\_x, int center\_y);

};

#endif // POLYGON\_H

**polygon.cpp:**

#include "polygon.h"

#include <QTextStream>

Polygon::**Polygon**()

{

}

Polygon::**Polygon**(const std::vector<std::vector<double>> &v) : verticies(v) {}

std::vector<double> Polygon::**get\_normal**() {

std::vector<double> first = {

verticies[1][0] - verticies[0][0],

verticies[1][1] - verticies[0][1],

verticies[1][2] - verticies[0][2]

};

std::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]

};

std::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::**change\_verticies**(const std::vector<std::vector<double>> &v) {

for (auto &it: verticies) {

std::vector<double> res(4);

for (size\_t i = 0; i < 4; i++) {

for (size\_t j = 0; j < 4; j++) {

res[i] += v[i][j] \* it[j];

}

}

it = res;

}

}

void Polygon::**add\_vertex**(const std::vector<double> &v) {

verticies.push\_back(v);

}

void Polygon::**add\_vertex**(double x, double y, double z, double d) {

verticies.push\_back(std::vector<double>{x, y, z, d});

}

void Polygon::**clear\_verticies**() {

verticies.clear();

}

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));

}

**pyramid.h:**

#ifndef PYRAMID\_H

#define PYRAMID\_H

#include "polygon.h"

class **Pyramid**

{

private:

std::vector<Polygon> polygons;

bool displayHidenLines;

public:

**Pyramid**();

**Pyramid**(const std::vector<Polygon> &p);

void **set\_displayHidenLines**(bool b);

bool **get\_displayHidenLines**();

void **change\_all\_polygons**(const std::vector<std::vector<double>> &v);

void **add\_polygon**(const Polygon &p);

void **draw**(QPainter \*ptr, int center\_x, int center\_y);

};

#endif // PYRAMID\_H

**pyramid.cpp:**

#include "pyramid.h"

Pyramid::**Pyramid**() : displayHidenLines(false)

{

}

Pyramid::**Pyramid**(const std::vector<Polygon> &p) : Pyramid() {

polygons = p;

}

void Pyramid::**set\_displayHidenLines**(bool b) {

displayHidenLines = b;

}

bool Pyramid::**get\_displayHidenLines**() {

return displayHidenLines;

}

void Pyramid::**change\_all\_polygons**(const std::vector<std::vector<double>> &v) {

for (auto &it: polygons) {

it.change\_verticies(v);

}

}

void Pyramid::**add\_polygon**(const Polygon &p) {

polygons.push\_back(p);

}

void Pyramid::**draw**(QPainter \*ptr, int center\_x, int center\_y) {

for (auto p : polygons) {

auto p\_normal = p.get\_normal();

if (p\_normal[2] > 0) {

p.draw(*ptr*, center\_x, center\_y);

} else {

if (displayHidenLines) {

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);

}

}

}

}

**5. Вывод.**

В ходе выполнения данной лабораторной работы я научился строить трехмерные модели, освежил свои знания по линейной алгебре.