**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Курсовая работа**

по «Компьютерной графике»

Тема: «Линейчатая поверхность Кунса»

Студент: Мариничев И. А.

Группа: М8О-308Б-19

Преподаватель: Филиппов Г. С.

Оценка:

Москва

2021

**1. Постановка задачи.**

Составить и отладить программу, обеспечивающую каркасную визуализацию порции поверхности заданного типа. Исходные данные готовятся самостоятельно и вводятся из файла или в панели ввода данных. Должна быть обеспечена возможность тестирования программы на различных наборах исходных данных. Программа должна обеспечивать выполнение аффинных преобразований для заданной порции поверхности, а также возможность управлять количеством изображаемых параметрических линий. Для визуализации параметрических линий поверхности разрешается использовать только функции отрисовки отрезков в экранных координатах.

**Вариант №8**: Линейчатая поверхность Кунса (границы – кривые Безье 3D 2-й степени)

**2. Описание программы.**

Рассмотрим способ построения поверхности . Пусть известны четыре граничные кривые Тогда билинейная смешивающая функция равна:

Однако, проверив этот результат в угловых точках куска поверхности, например:

и на границах:

Так как угловые точки учитываются дважды, правильный результат можно получить только с помощью вычитания дополнительных членов, возникающих из-за удвоения угловых точек:

Граничными кривыми в моём варианте являются кривые Безье 2-й степени, они строятся в соответствии со следующей формулой:

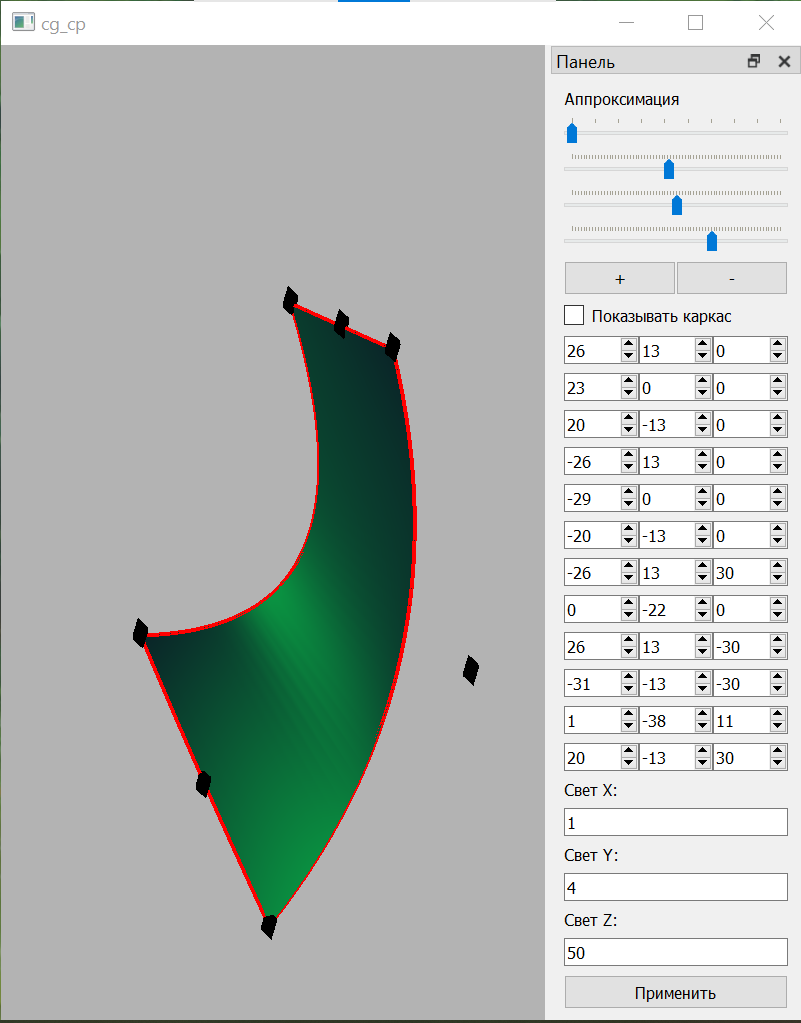
,

где — опорные точки, .

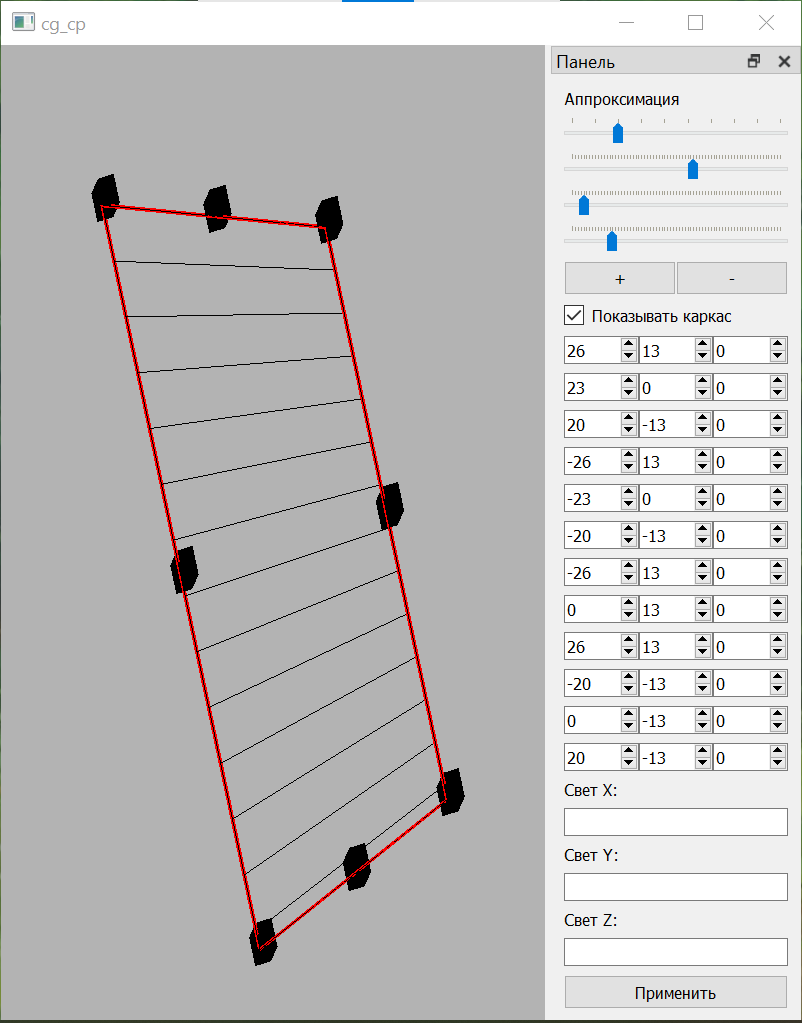
Для решения задачи я использовал C++, фреймворк Qt и OpenGL.

**3. Демонстрация работы программы.**

1) Вид поверхности при высокой аппроксимации, заданными материалами и наличии света на сцене.



2) Вид поверхности при включённом отображении каркаса.

****

**4. Основной код программы.**

1) Метод для отрисовки кривой Безье.

void display::**drawBezierCurve**(*const* std::vector<QVector4D> &points) {

glColor3f(1.f, 0.f, 0.f);

float prevX = points[0].x() + SQUARE\_SIZE / 2, prevY = points[0].y() - SQUARE\_SIZE / 2;

float prevZ = points[0].z();

glLineWidth(4.f);

glBegin(GL\_LINE\_STRIP);

glVertex3f(prevX, prevY, prevZ);

*for* (double t = *static\_cast*<double>(step); t < 1.; t += *static\_cast*<double>(step)) {

float x = *static\_cast*<float>(std::pow((1. - t), 2.)) \* (points[0].x() + SQUARE\_SIZE / 2) +

2.f \* *static\_cast*<float>(t) \* *static\_cast*<float>(1. - t) \* (points[1].x() + SQUARE\_SIZE / 2) +

*static\_cast*<float>(std::pow(t, 2.)) \* (points[2].x() + SQUARE\_SIZE / 2);

float y = *static\_cast*<float>(std::pow((1. - t), 2.)) \* (points[0].y() - SQUARE\_SIZE / 2) +

2.f \* *static\_cast*<float>(t) \* *static\_cast*<float>(1. - t) \* (points[1].y() - SQUARE\_SIZE / 2) +

*static\_cast*<float>(std::pow(t, 2.)) \* (points[2].y() - SQUARE\_SIZE / 2);

float z = *static\_cast*<float>(std::pow((1. - t), 2.)) \* (points[0].z()) +

2.f \* *static\_cast*<float>(t) \* *static\_cast*<float>(1. - t) \* (points[1].z()) +

*static\_cast*<float>(std::pow(t, 2.)) \* (points[2].z());

glVertex3f(x, y, z);

prevX = x;

prevY = y;

prevZ = z;

*if* (t + *static\_cast*<double>(step) >= 1.) {

x = points[2].x() + SQUARE\_SIZE / 2;

y = points[2].y() - SQUARE\_SIZE / 2;

z = points[2].z();

glVertex3f(x, y, z);

}

}

glEnd();

}

2) Методы для отрисовки точек линейчатой поверхности.

float display::**calcX**(double t, double v) {

double x = (1 - v) \* (std::pow(1 - t, 2) \* *static\_cast*<double>(points1[0].x() + SQUARE\_SIZE / 2) +

2 \* t \* (1 - t) \* *static\_cast*<double>(points1[1].x() + SQUARE\_SIZE / 2) +

std::pow(t, 2) \* *static\_cast*<double>(points1[2].x() + SQUARE\_SIZE / 2)) +

v \* (std::pow(1 - t, 2) \* *static\_cast*<double>(points2[0].x() + SQUARE\_SIZE / 2) +

2 \* t \* (1 - t) \* *static\_cast*<double>(points2[1].x() + SQUARE\_SIZE / 2) +

std::pow(t, 2) \* *static\_cast*<double>(points2[2].x() + SQUARE\_SIZE / 2));

*return* *static\_cast*<float>(x);

}

float display::**calcY**(double t, double v) {

double y = (1 - v) \* (std::pow(1 - t, 2) \* *static\_cast*<double>(points1[0].y() - SQUARE\_SIZE / 2) +

2 \* t \* (1 - t) \* *static\_cast*<double>(points1[1].y() - SQUARE\_SIZE / 2) +

std::pow(t, 2) \* *static\_cast*<double>(points1[2].y() - SQUARE\_SIZE / 2)) +

v \* (std::pow(1 - t, 2) \* *static\_cast*<double>(points2[0].y() - SQUARE\_SIZE / 2) +

2 \* t \* (1 - t) \* *static\_cast*<double>(points2[1].y() - SQUARE\_SIZE / 2) +

std::pow(t, 2) \* *static\_cast*<double>(points2[2].y() - SQUARE\_SIZE / 2));

*return* *static\_cast*<float>(y);

}

float display::**calcZ**(double t, double v) {

double z = (1 - v) \* (std::pow(1 - t, 2) \* *static\_cast*<double>(points1[0].z()) +

2 \* t \* (1 - t) \* *static\_cast*<double>(points1[1].z()) +

std::pow(t, 2) \* *static\_cast*<double>(points1[2].z())) +

v \* (std::pow(1 - t, 2) \* *static\_cast*<double>(points2[0].z()) +

2 \* t \* (1 - t) \* *static\_cast*<double>(points2[1].z()) +

std::pow(t, 2) \* *static\_cast*<double>(points2[2].z()));

*return* *static\_cast*<float>(z);

}

**5. Выводы.**

В ходе выполнения данной курсовой работы была реализована программа, позволяющая моделировать линейчатую поверхность Кунса, в качестве границ которой выступают кривые Безье 2-й степени. Для выполнения данного задания мне пригодились, навыки, полученные в 4, 5 и 7 лабораторных работах.

**6. Список литературы.**

1. Д. Роджерс, Дж. Адамс. Математические основы машинной графики, – М.: из-во “Мир”, 2001.

2. Голованов Н.Н. Геометрическое моделирование. – М.: Издательство физико-математической литературы, 2002.

3. Шишкин Е.В., Плис А.И. Кривые и поверхности на экране компьютера. Руководство по сплайнам для пользователя.