МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Лабораторная работа № 4

**Моделирование параметрических кривых и поверхностей**

Выполнил студент 595 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_ Лаптев А.В.

Проверил:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Калачев А.В.

Лабораторная работа защищена

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Барнаул 2023

# Задание 1.

Необходимо разработать интерактивную OpenGL-программу, позволяющую конструировать плоские сплайны, составленные из кривых Безье.

Программа должна предоставлять возможности

* создания, удаления и изменения положения и (при необходимости) прочих параметров каркасных точек (или каркасных ломаных) для очередного сплайнового сегмента;
* обеспечения стыковки сплайновых сегментов с заданным уровнем гладкости.

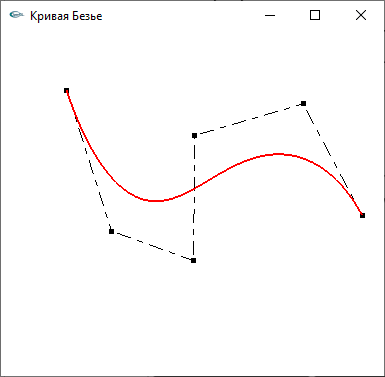
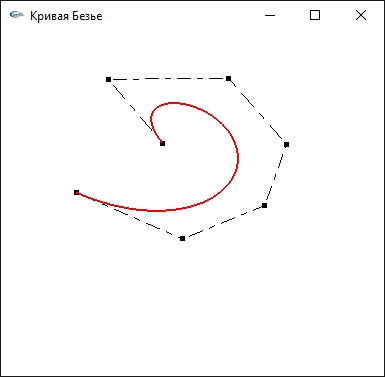
Создаваемую программу следует расценивать как инструмент рисования и моделирования, поэтому управление программой должно осуществляться преимущественно с помощью «мыши». Прибегать к явному вводу каких-либо значений следует лишь в самом крайнем случае. В качестве примеров для подражания следует рассматривать широко известные графические редакторы, реализующие инструменты построения кривых.

**Выполнение работы:**

Для инициализации OpenGL был использован glut. Текст программы находится в приложении А.

Работа программы выглядит следующим образом:

**Задание 2.**



Необходимо создать один сегмент поверхности Безье, заданный как минимум 16-ю контрольными точками (см. примеры), используя функции из библиотеки GL (glMap2[fd], glMapGrid2[fd], glEvalMesh2).

**Выполнение работы:**

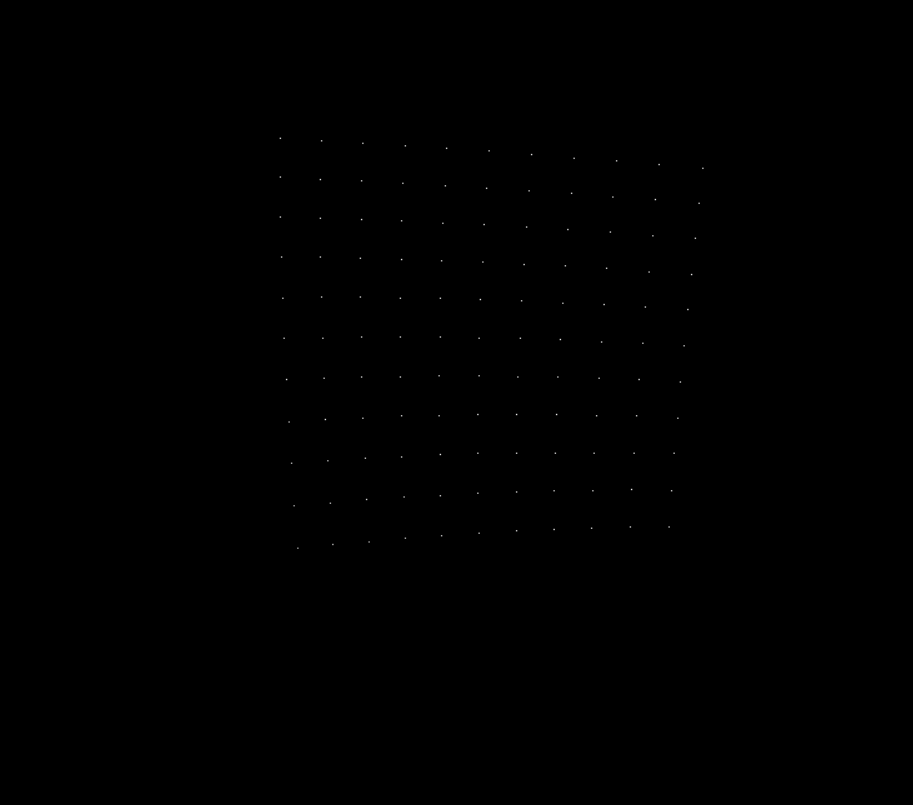
**Поверхность Безье** — параметрическая поверхность, используемая в компьютерной графике, автоматизированном проектировании, и моделировании. Это одно из распространённых пространственных обобщений кривой Безье.

При кусочном моделировании (patch modeling) для задания и изменения формы куска, представляющего собой пространственную решетку из сплайнов или многоугольников, применяется сеть контрольных точек. Эти точки управления, также известные как контрольные вершины (control vertices — CV) оказывают на гибкую поверхность куска подобное магнитному влияние, при котором поверхность растягивается в том или ином направлении. Кроме того, куски можно и дальше подразделять на элементы для достижения большего разрешения и «сшивать» друг с другом, тем самым создавая сложные объёмные поверхности. Так же, как и сплайновые, кусочные модели используются при создании органических форм.

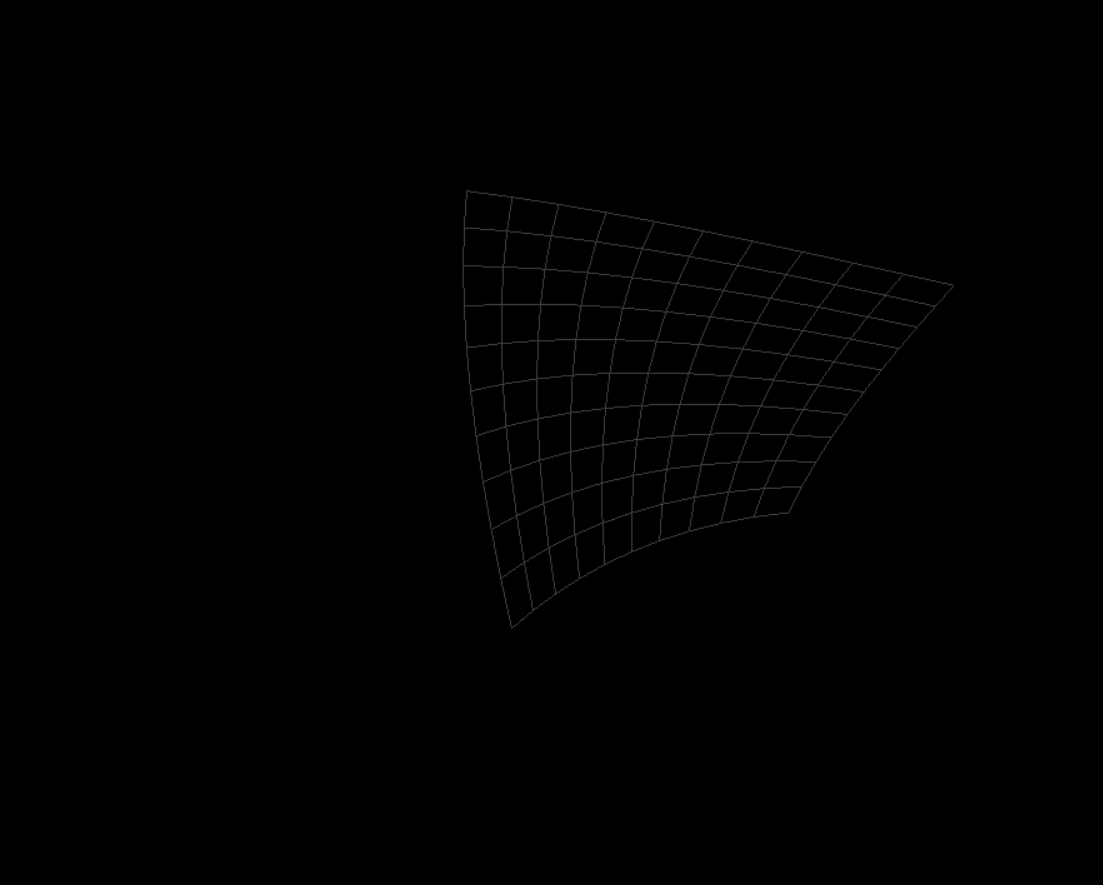
Уравнение поверхности:

Код программы находится в приложении Б. Работа программы:

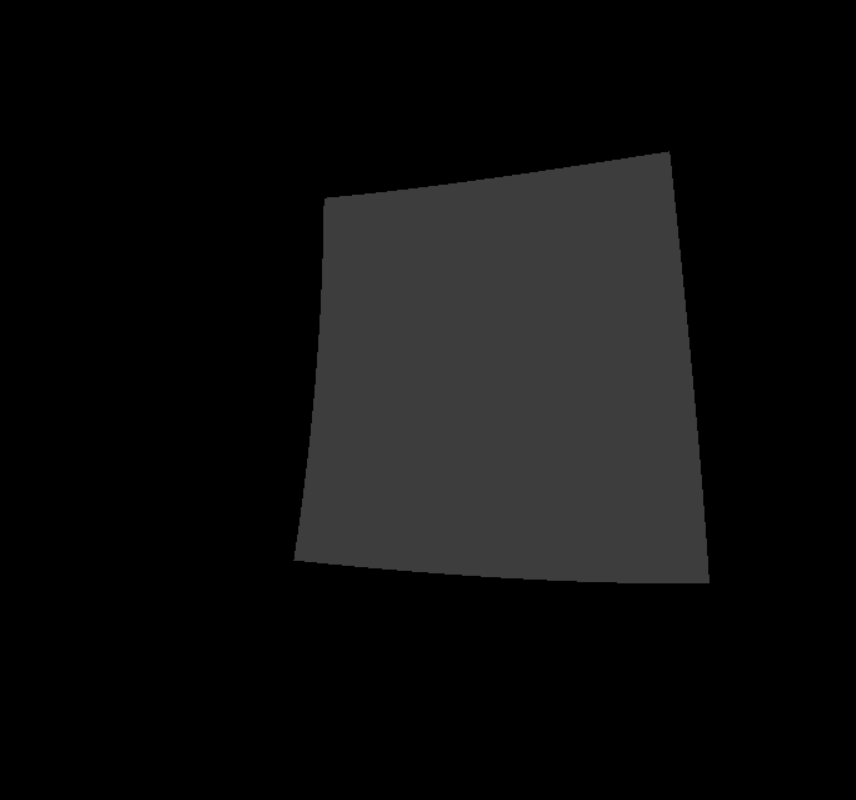
GL\_POINT



GL\_LINE



GL\_FILL



**Вывод:** Мной были изучены основы моделирования с помощью параметрических кривых, поверхностей и инструментов их реализации с использованием OpenGL. Были сделаны программа-редактор кривой Безье и поверхность sin(x+y) в виде точек, которая вращается, что дает возможность рассмотреть ее со всех сторон.

## Приложение А.

#include "stdafx.h" #include "glut.h"

using namespace std;

float point[350][3]; int n = 0, i, j; bool tP = false; void Draw();

void Mouse(int button, int state, int x, int y); void MenuChek(int v);

void CurveBezier(); void Line();

void main() {

//Инициализируем режим отображения окна OpenGL

//GLUT\_DOUBLE - окно с двойной буферизацией

//GLUT\_RGB - режим RGBA glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(350, 350); //устанавливаем размер окна glutCreateWindow("Кривая Безье"); //создаем окно с заголовком glutDisplayFunc(Draw); //устанавливаем функцию отрисовки glutMouseFunc(Mouse); //устанавливаем функцию обработки нажатий мыши

//Установить цвет и значение альфа, используемые при очистке буферов цвета glClearColor(0, 0, 0, 0); //цвет фона (RGBA) glMatrixMode(GL\_PROJECTION); //Определяем стек матриц (матрица проекций) glLoadIdentity(); //установить текущую матрицу равной еденичной

//Установить границы объема отсечения glOrtho(0, 350, 0, 350, 0, 1);

glColor3f(1.0, 0, 1.0); //Установить текущий цвет (R,G,B)

glutCreateMenu(MenuChek); //Меню вызываемое нажатием ПКМ glutAddMenuEntry("Удалить последнюю точку", 0); //Пункт меню glutAddMenuEntry("Очистить", 1); //Пункт меню glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutMainLoop(); //Запуск основного цикла обработки GLUT

}

void Draw() {

glClear(GL\_COLOR\_BUFFER\_BIT); //Очистка буфера цвета Line();

if (n > 1)

CurveBezier();

glutSwapBuffers(); //Переключить буферы в режиме двойной буферизации

}

void Mouse(int button, int state, int x, int y) { if (button == GLUT\_LEFT\_BUTTON) {

if (tP && (x > 0 && x < 350) && (350 > y&& y < 700)) { point[j][0] = x;

point[j][1] = 350 - y;

}

else if (state == GLUT\_DOWN && n != 0 && !tP) for (i = 0; i <= n; i++)

if ((x<point[i][0] + 30 && x>point[i][0] - 30) && (320 -

y<point[i][1] && 380 - y>point[i][1])) {

tP = true; j = i; break;

}

if (state == GLUT\_UP) { if (!tP) {

point[n][0] = x;

point[n][1] = 350 - y; n++;

}

tP = false;

}

}

glutPostRedisplay(); //Обновить текущее окно

}

void MenuChek(int v) {

if (v == 0 && n > 0)

n--;

else if (v == 1)

n = 0;

glutPostRedisplay();

}

void Line() {

glPointSize(5); glColor3f(0.75, 0, 1);

//Контрольные точки glBegin(GL\_POINTS);

for (i = n - 1; i >= 0; i--) glVertex2fv(point[i]);

glEnd();

//Пунктирные линии glLineStipple(2, 58360); glEnable(GL\_LINE\_STIPPLE); glBegin(GL\_LINES);

for (i = 0; i < n - 1; i++) { glVertex2fv(point[i]); glVertex2fv(point[i + 1]);

}

glEnd(); glDisable(GL\_LINE\_STIPPLE); glPointSize(1);

}

void CurveBezier() {

//glMap1f - Функция оценки, генерирующая координаты.

//Параметры:

// GL\_MAP1\_VERTEX\_3 - тип генерируемых данных,

// 0.0f - нижняя граница параметра u (первая точка),

// 100.0f - верхняя граница(последняя точка),

// 3 - расстояние между точками данных,

// n - число точек,

// &P[0][0] - массив контрольных точек glMap1f(GL\_MAP1\_VERTEX\_3, 0.0f, 100.0f, 3, n, &point[0][0]); glEnable(GL\_MAP1\_VERTEX\_3);

glLineWidth(2); glColor3f(0.0f, 1.0f, 0.0f);

glBegin(GL\_LINE\_STRIP); //точки соединяются ломанной линией for (i = 0; i <= 100; i++)

//Оценка кривой в точке. Функция принимает параметрическое значение, и вычисляет точку

glEvalCoord1f(float(i)); glEnd();

glLineWidth(1);

}

## Приложение Б.

#include <stdlib.h> #include "glut.h" #include <math.h> #include <stdio.h>

#pragma warning (disable:4996) GLfloat ctrlpoints[6][6][3];

void initlights(void)

{

GLfloat ambient[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat position[] = { 0.0, 0.0, 2.0, 1.0 };

GLfloat mat\_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 }; GLfloat mat\_shininess[] = { 50.0 }; glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, ambient); glLightfv(GL\_LIGHT0, GL\_POSITION, position); glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse); glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular); glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);

}

unsigned int angle=0; void display(void)

{

int i, j; while (true)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glPushMatrix(); glRotatef(angle++, 1, 1, 0);

glEvalMesh2(GL\_POINT, 0, 10, 0, 10); // изменение отображения glPopMatrix();

glFlush();

\_sleep(10);

}

}

void init(void)

{

float x, y; int i, j;

for (i = 0; i < 6; i++)

{

x = 3.1415 / 5.0 \* (float)i; for (j = 0; j < 6; j++)

{

y = 3.1415 / 5.0 \* (float)j;

ctrlpoints[i][j][0] = x/2; ctrlpoints[i][j][1] = y/2; ctrlpoints[i][j][2] = sinf(x+y)/2;

}

}

glClearColor(0.0, 0.0, 0.0, 0.0); glEnable(GL\_DEPTH\_TEST); glMap2f(GL\_MAP2\_VERTEX\_3, 0, 4, 3, 4,

0, 4, 18, 4, &ctrlpoints[0][0][0]); glEnable(GL\_MAP2\_VERTEX\_3); glEnable(GL\_AUTO\_NORMAL); glMapGrid2f(10, 0.0, 4.0, 10, 0.0, 4.0);

initlights(); /\* for lighted version only \*/

}

void reshape(int w, int h)

{

glViewport(100, 0, 1000, 1000); glMatrixMode(GL\_PROJECTION); glLoadIdentity();

glOrtho(-1.0, 2.0, -1.0, 2.0, -2.0, 2.0);

glMatrixMode(GL\_MODELVIEW); glLoadIdentity();

}

void keyboard(unsigned char key, int x, int y)

{

switch (key) {

case 27:

exit(0); break;

}

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH); glutInitWindowSize(1000, 1000);

glutInitWindowPosition(0, 0); glutCreateWindow(argv[0]); init(); glutReshapeFunc(reshape); glutDisplayFunc(display); glutKeyboardFunc(keyboard); glutMainLoop();

return 0;

}