МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Компьютерная графика»

Тема: Формирования различных поверхностей с использованием ее пространственного разворота и ортогонального проецирования на плоскость при ее визуализации (выводе на экран дисплея)

Студент гр. 9308 _____ Степовик В.С.
Соболев М.С.

Преподаватель _____ Матвеева И.В.

Санкт-Петербург
2022

Оглавление

Цель работы

Формирование различных поверхностей с использованием её пространственного разворота и ортогонального проецирования на плоскость при её визуализации (выводе на экран дисплея).

Задание

Сформировать билинейную поверхность на основе произвольного задания её четырёх угловых точек. Обеспечить её поворот относительно осей X и Y.

Используемые ресурсы

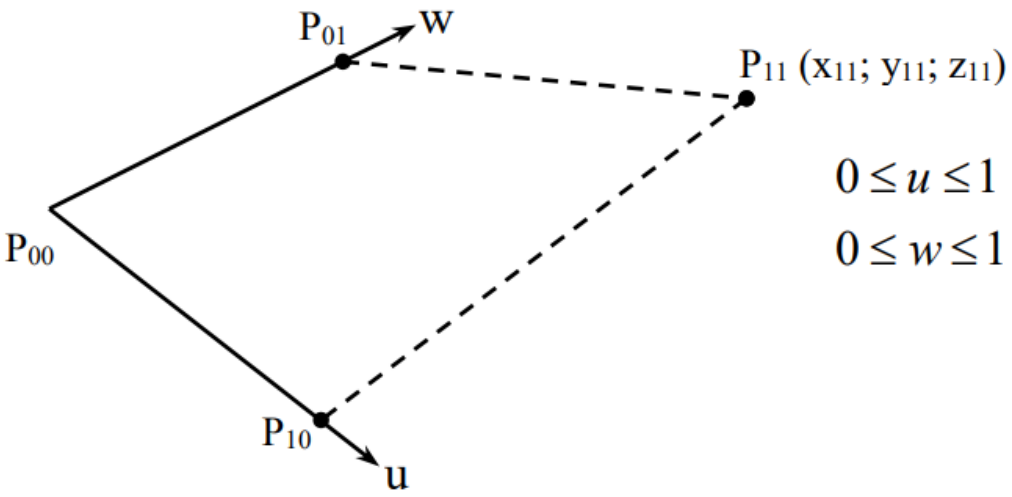
Для выполнения лабораторной работы использовался язык C++ и фреймворк Qt для визуализации.

Основные теоретические положения

Поверхности задаются параметрически от двух независимых параметров u и w (отдельно по каждому параметру), т.е. можем задавать неоднозначные поверхности (т.е. для одного и того же значения одного параметра второй может иметь несколько значений): $\overline{Q}(u,w) = f(\overline{P}_i(u,w))$ - параметрическая зависимость поверхности, позволяющая определить положение координат любой её точки в функции от значений координат этой поверхности в заданных точках. При этом значение Q(u,w) на промежутках задания параметров u и w может определяться (меняться) непрерывно, а значения $P_i(u, w)$ задаются для конкретных значений u и w.

При этом координаты любой точки (X, Y и Z), относящейся к поверхности определяются исходя из соответствующих координат (X, Y и Z) точек задания и задающей функции, которая для всех координат одинаковая, т.е. $X(u,w)\ f(X_i(u,w))$ и т.д.

1. Простейшими трехмерными поверхностями являются Билинейные поверхности, их задают на ограниченном участке. Для такого участка поверхности требуется задание в пространстве 4-х угловых точек поверхности.



Тогда уравнение билинейчатой поверхности представляется как:

$$\overline{Q}(u,w) = \overline{P_{00}}(1-u)(1-w) + \overline{P_{01}}(1-u)w + \overline{P_{10}}u(1-w) + \overline{P_{11}}uw$$

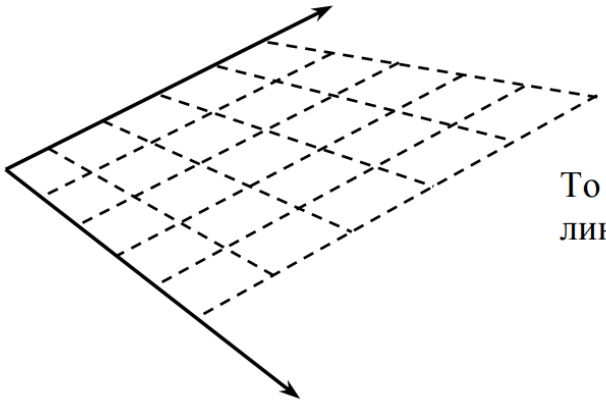
Если u = 0; w = 0, то попадаем в точку $\overline{P_{00}} = \overline{Q}(u,w)$;

Если u = 1; w = 0, то попадаем в точку $\overline{P_{10}} = \overline{Q}(u,w)$;

Если u = 1; w = 1, то попадаем в точку $\overline{P_{11}} = \overline{Q}(u,w)$.

Если по каждому параметру разделим на 5 участков, то получаем сетку с линейной аппроксимацией.

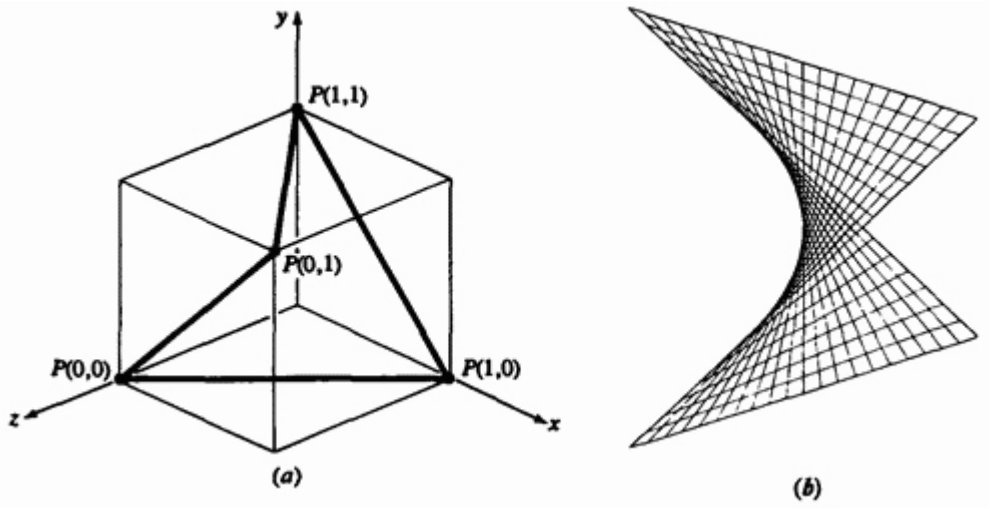
Если по каждому параметру разделим на 5 участков:



То получаем сетку с линейной аппроксимацией

Пример.

$$Q(0.5,0.5)=[0\ \ 0\ \ 1](1-0.5)(1-0.5)+[1\ \ 1\ \ 1](1-0.5)(0.5)+$$
$$+[1\ \ 0\ \ 0](0.5)(1-0.5)+[0\ \ 1\ \ 0](0.5)(0.5)=$$
$$=0.25[0\ \ 0\ \ 1]+0.25[1\ \ 1\ \ 1]+$$
$$+0.25[1\ \ 0\ \ 0]+0.25[0\ \ 1\ \ 0]=$$
$$=[0.5\ \ 0.5\ \ 0.5].$$



(a) Определяющие угловые точки; (b) билинейная поверхность.

Ход работы

Т.к. билинейная поверхность строится по 4м точкам и любая точка на поверхности определяется линейной интерполяцией между противоположными границами единичного квадрата, то имеем следующую функцию для построения точки на билинейной поверхности:

```
sPoint DrawField::calBilinearSurface(double u, double w)
{
  //https://scask.ru/a_book_mm3d.php?id=105
  Matrix<double> vold1(3, 1);
  Matrix<double> vold2(3, 1);
  Matrix<double> vold3(3, 1);
  Matrix<double> vold4(3, 1);

  vold1.set(p1.x(), 0, 0); vold2.set(p2.x(), 0, 0); vold3.set(p3.x(), 0, 0); vold4.set(p4.x(), 0, 0);
  vold1.set(p1.y(), 1, 0); vold2.set(p2.y(), 1, 0); vold3.set(p3.y(), 1, 0); vold4.set(p4.y(), 1, 0);
  vold1.set(p1.z(), 2, 0); vold2.set(p2.z(), 2, 0); vold3.set(p3.z(), 2, 0); vold4.set(p4.z(), 2, 0);

  Matrix<double> vnew1(M_rotate.multiply(vold1));
  Matrix<double> vnew2(M_rotate.multiply(vold2));
```

```
  Matrix<double> vnew3(M_rotate.multiply(vold3));
  Matrix<double> vnew4(M_rotate.multiply(vold4));

  sPoint _p1(vnew1.get(0, 0), vnew1.get(1, 0), vnew1.get(2, 0));
  sPoint _p2(vnew2.get(0, 0), vnew2.get(1, 0), vnew2.get(2, 0));
  sPoint _p3(vnew3.get(0, 0), vnew3.get(1, 0), vnew3.get(2, 0));
  sPoint _p4(vnew4.get(0, 0), vnew4.get(1, 0), vnew4.get(2, 0));

  double x_res = _p1.x()*(1-u)*(1-w)
          +  _p2.x()*(1-u)*w
          +  _p3.x()*u*(1-w)
          +  _p4.x()*u*w;
  double y_res = _p1.y()*(1-u)*(1-w)
          +  _p2.y()*(1-u)*w
          +  _p3.y()*u*(1-w)
          +  _p4.y()*u*w;
  double z_res = _p1.z()*(1-u)*(1-w)
          +  _p2.z()*(1-u)*w
          +  _p3.z()*u*(1-w)
          +  _p4.z()*u*w;

  return sPoint(x_res, y_res, z_res);
}
```

Вышеописанная функцию будем вызывать для комбинаций различных комбинаций независимых параметров u,w (0<=u<=1; 0<=w<=1):

```
        void DrawField::drawBilinearSurface()
{
  double u, w;

  for(w = 0; w <= 1.0; w+=0.1)
     for(u = 0; u <= 1.0; u+=0.001)
     {
```

```
            sPoint buffP = calBilinearSurface(u, w);
            putPoint(buffP.x(), buffP.y(), buffP.z());
        }

    for(u = 0; u <= 1.0; u+=0.1)
        for(w = 0; w <= 1.0; w+=0.001)
        {
            sPoint buffP = calBilinearSurface(u, w);
            putPoint(buffP.x(), buffP.y(), buffP.z());
        }
}
```

Ввиду того, что билинейная поверхность строится в трёхмерном пространстве, добавим камеру для просмотра построения со всевозможных ракурсов.

По сути камера не является отдельным объектом на сцене - это вся сцена изменятся матрицей поворота на заданное значение при нажатии соответствующей кнопки управления камерой:

```
        void DrawField::refresh_C_()
{
    sPoint vx(cam.vr());
    sPoint vy(cam.vf());
    sPoint vz(cam.vu());

    Matrix<double> C(3, 3);
    C.set(vx.x(), 0, 0); C.set(vy.x(), 0, 1); C.set(vz.x(), 0, 2);
    C.set(vx.y(), 1, 0); C.set(vy.y(), 1, 1); C.set(vz.y(), 1, 2);
    C.set(vx.z(), 2, 0); C.set(vy.z(), 2, 1); C.set(vz.z(), 2, 2);
    //std::cout << C.toString() << std::endl;
    Matrix<double> C_buff = C.inverse();

    if(C_ != NULL)
```

```
    delete C_;
  C_ = new Matrix<double>(C_buff);
}
```

Пример работы программы

Пример работы программы представлен на рисунках ниже.

Сама программа запускается через командную строку с указанием точек построения билинейной поверхности: .\lab3ex.exe p1_x p1_y p1_z p2_x p2_y p2_z p3_x p3_y p3_z p4_x p4_y p4_z, например “4 4 4 5 6 7 7 7 9 9 10 9”.

Управление происходит следующими командами:

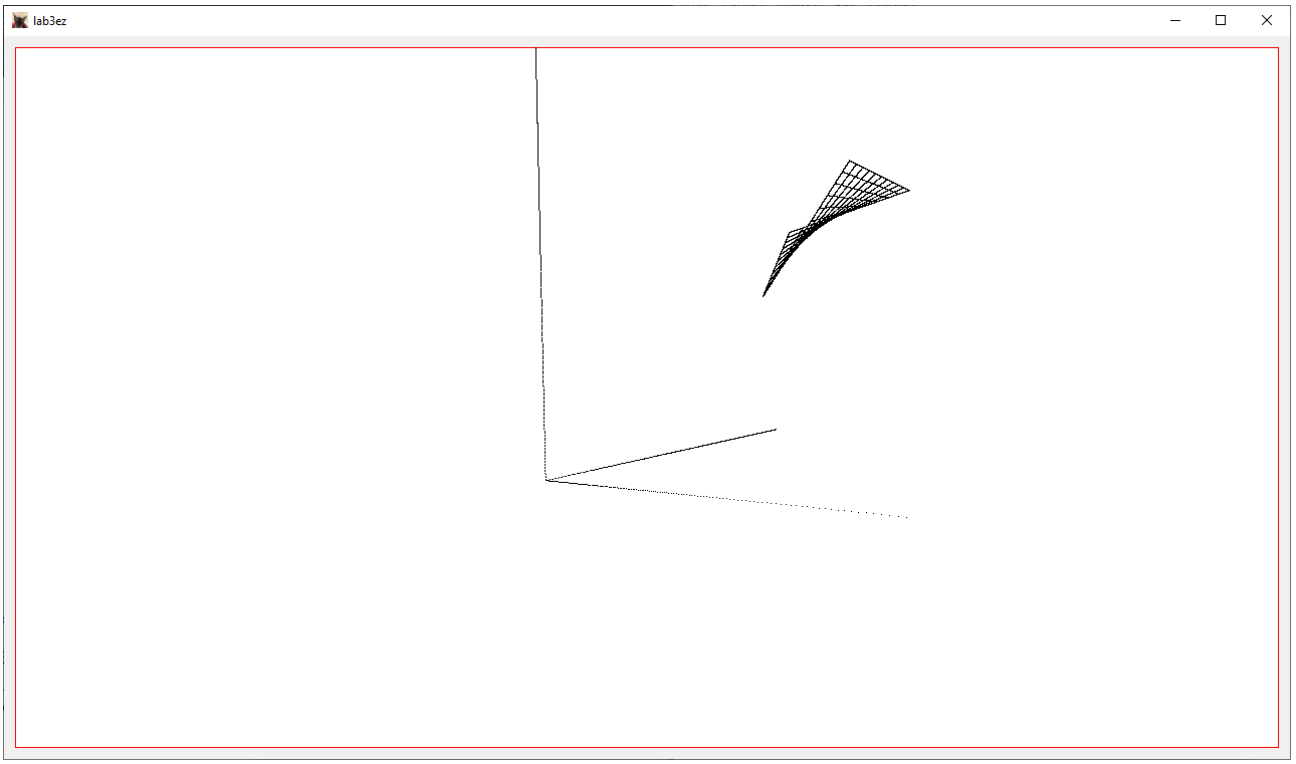
WSADQE – поворот камеры относительно текущей позиции зрителя (наклон “головы” вниз, наклон “головы” вверх, поворот “головы” влево, поворот “головы” вправо, наклон “головы” влево, наклон “головы” вправо СООТВЕТСТВЕННО);

YI<up><down><left><right> – движение по пространству относительно текущей позиции зрителя (вверх, вниз, вперёд, назад, влево, вправо СООТВЕТСТВЕННО);

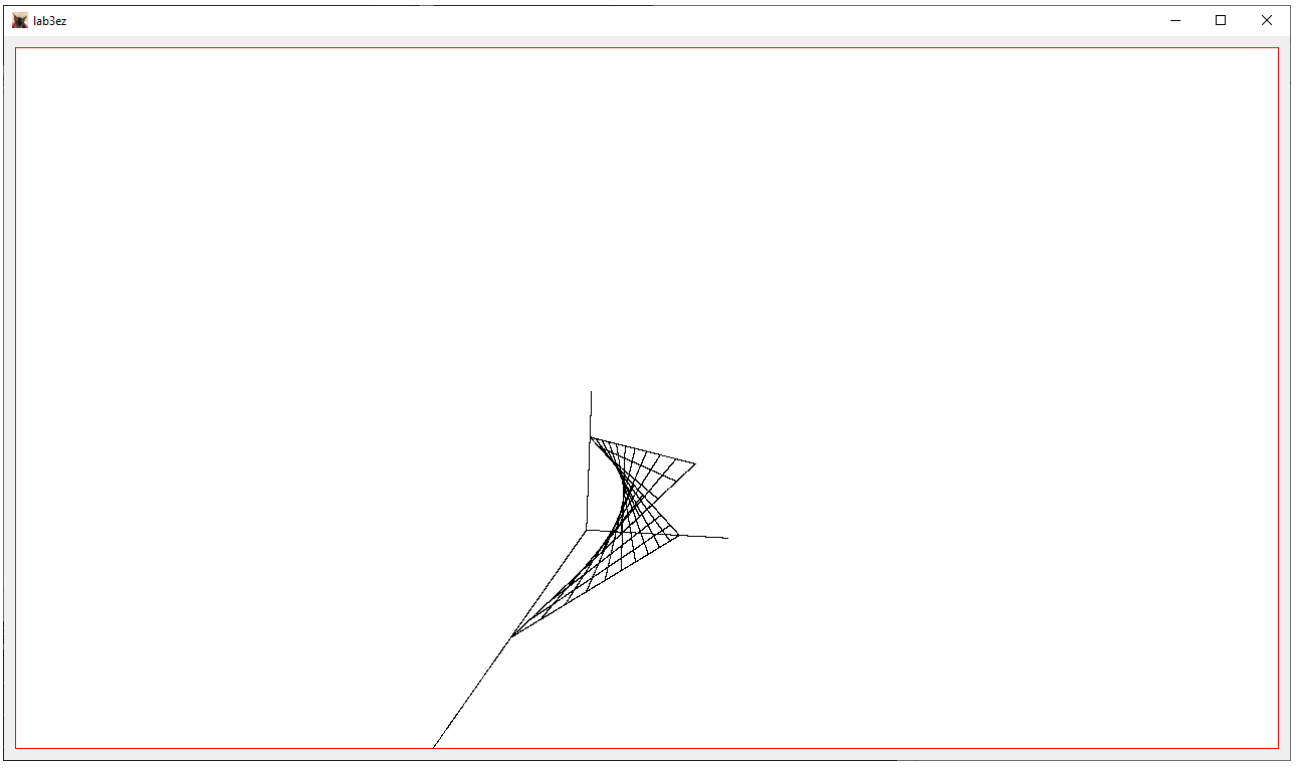
UHJK<space>C – движение по пространству относительно начала осей координат (положительно по OX, положительно по OY, отрицательно, по OX отрицательно по OY, положительно по OZ, отрицательно по OZ СООТВЕТСТВЕННО);

RT – вращение относительно OX (против часовой, если смотреть от 0 в X, по часовой, если смотреть от 0 в X СООТВЕТСТВЕННО);

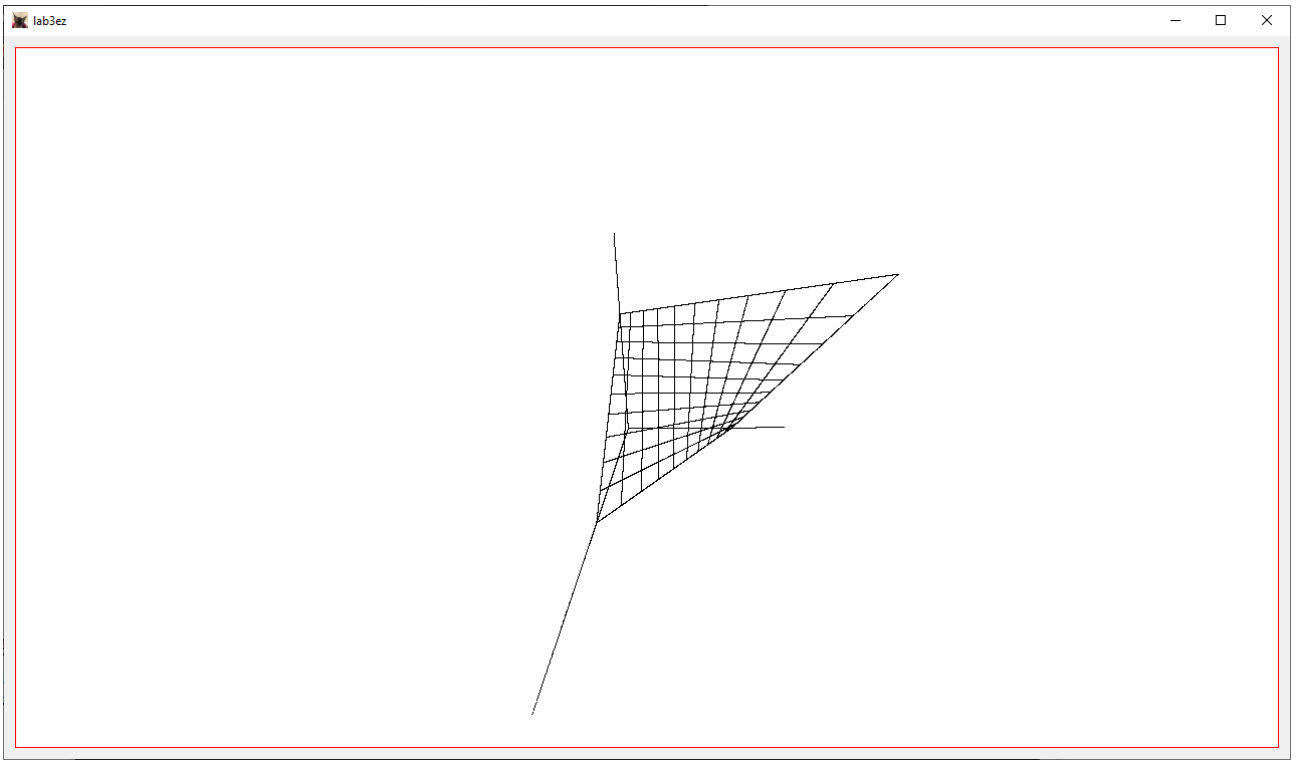
FG – вращение относительно OY (по часовой, если смотреть от 0 в Y, против часовой, если смотреть от 0 в Y СООТВЕТСТВЕННО).



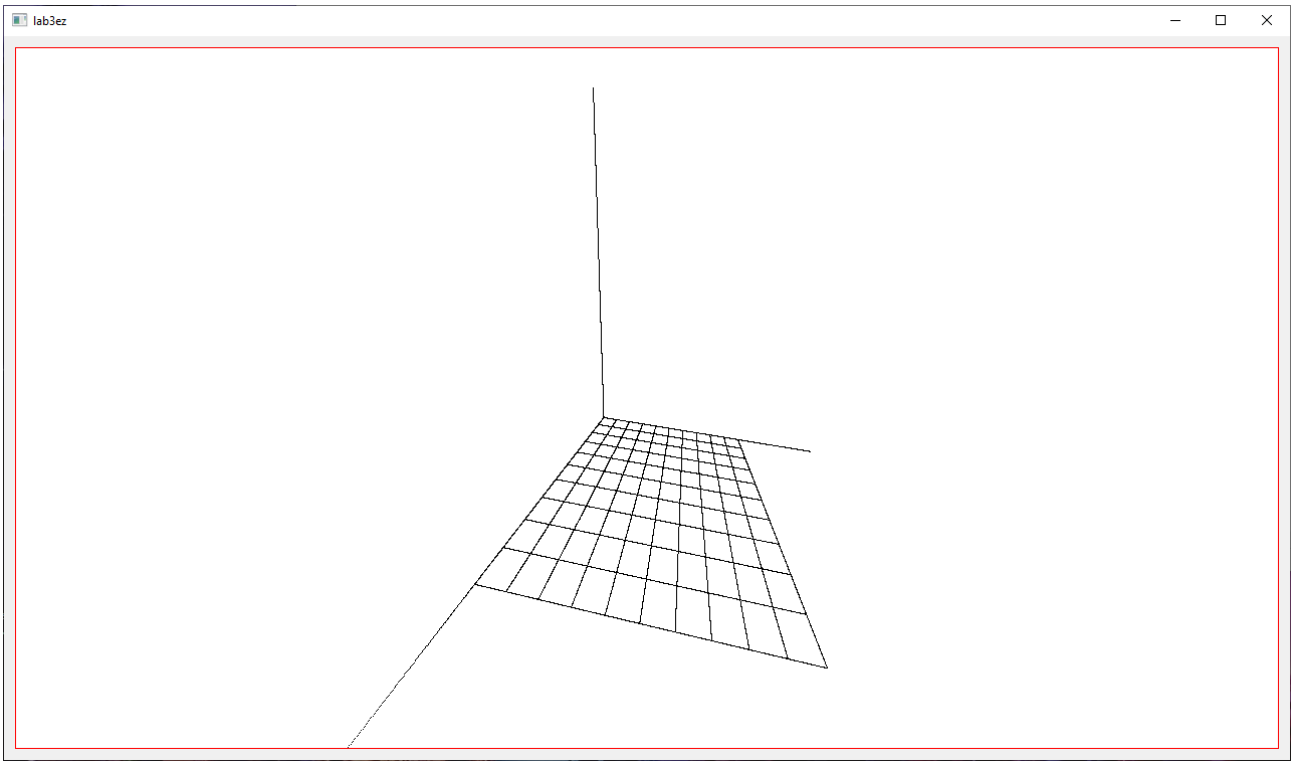
Пример 1. “4 4 4 5 6 7 7 7 9 9 10 9”



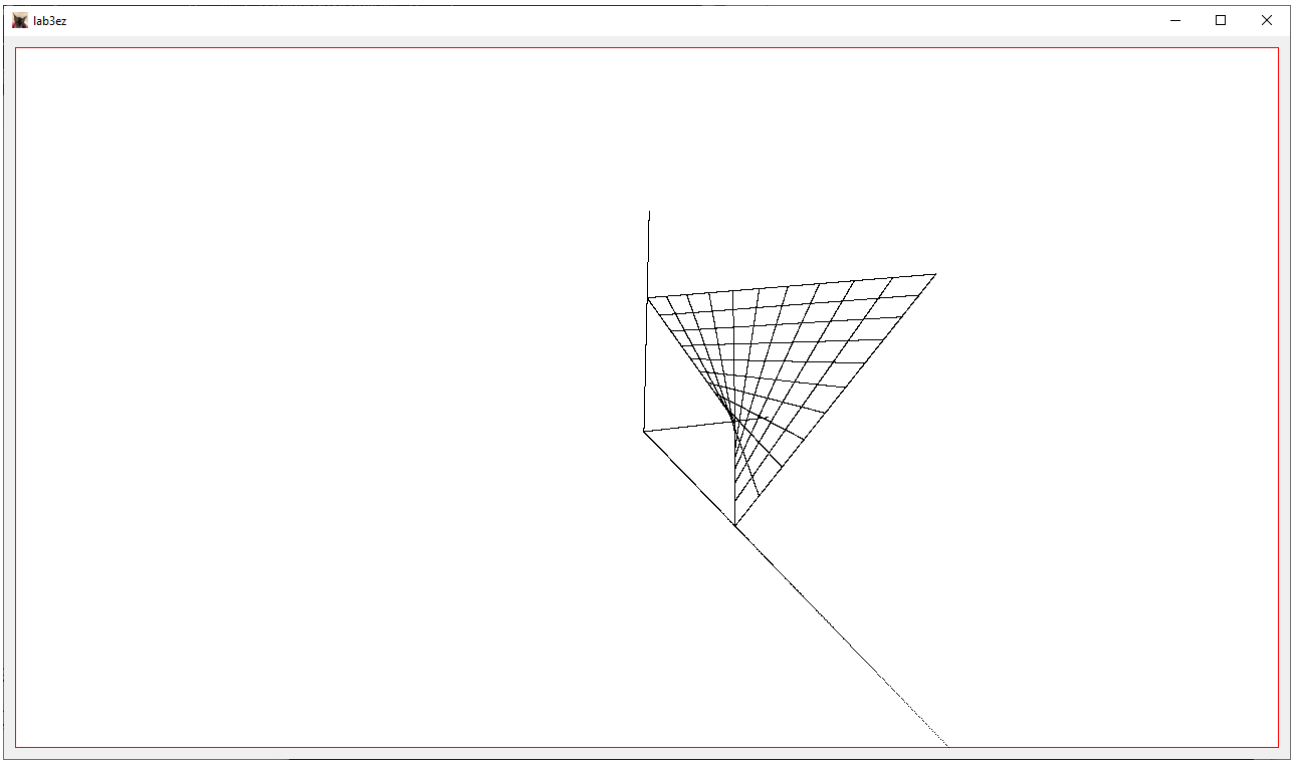
Пример 2. “0 0 20 20 20 20 20 0 0 0 20 0”



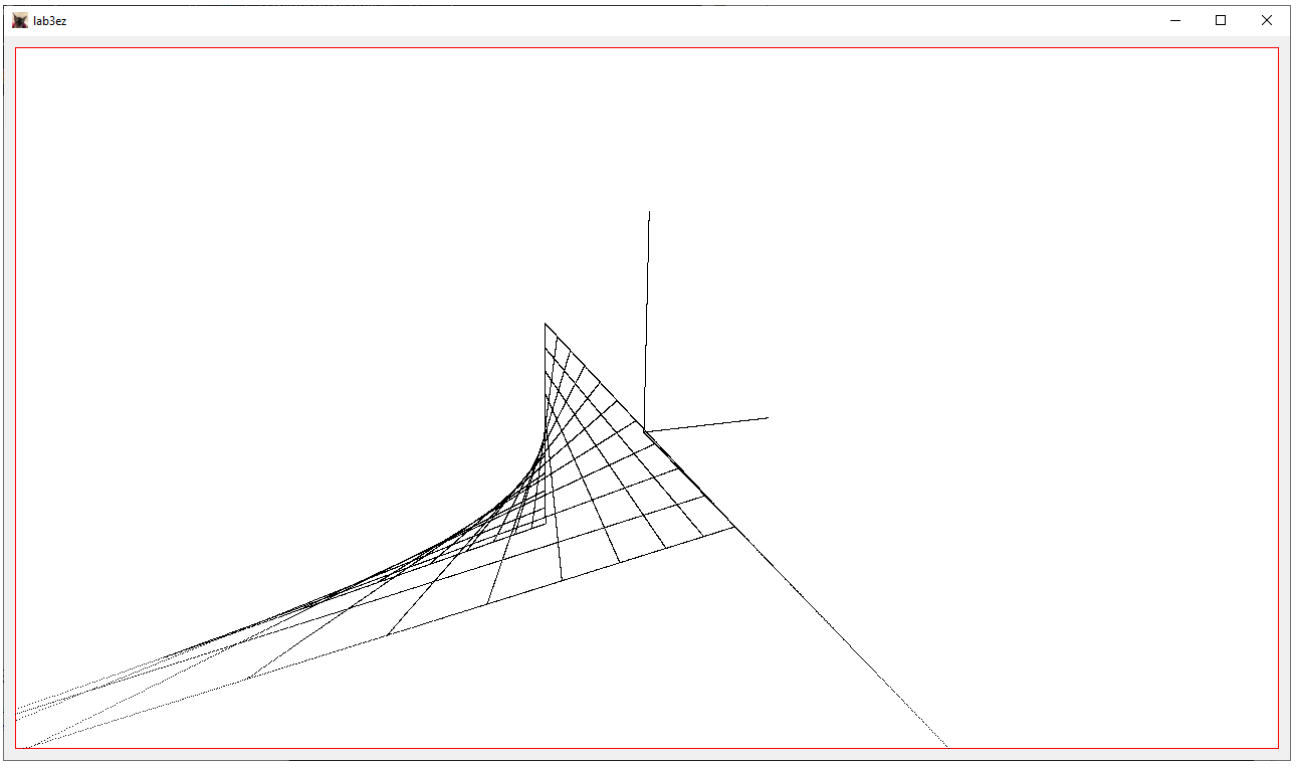
Пример 3. “20 0 0 20 20 20 0 20 0 0 0 20”



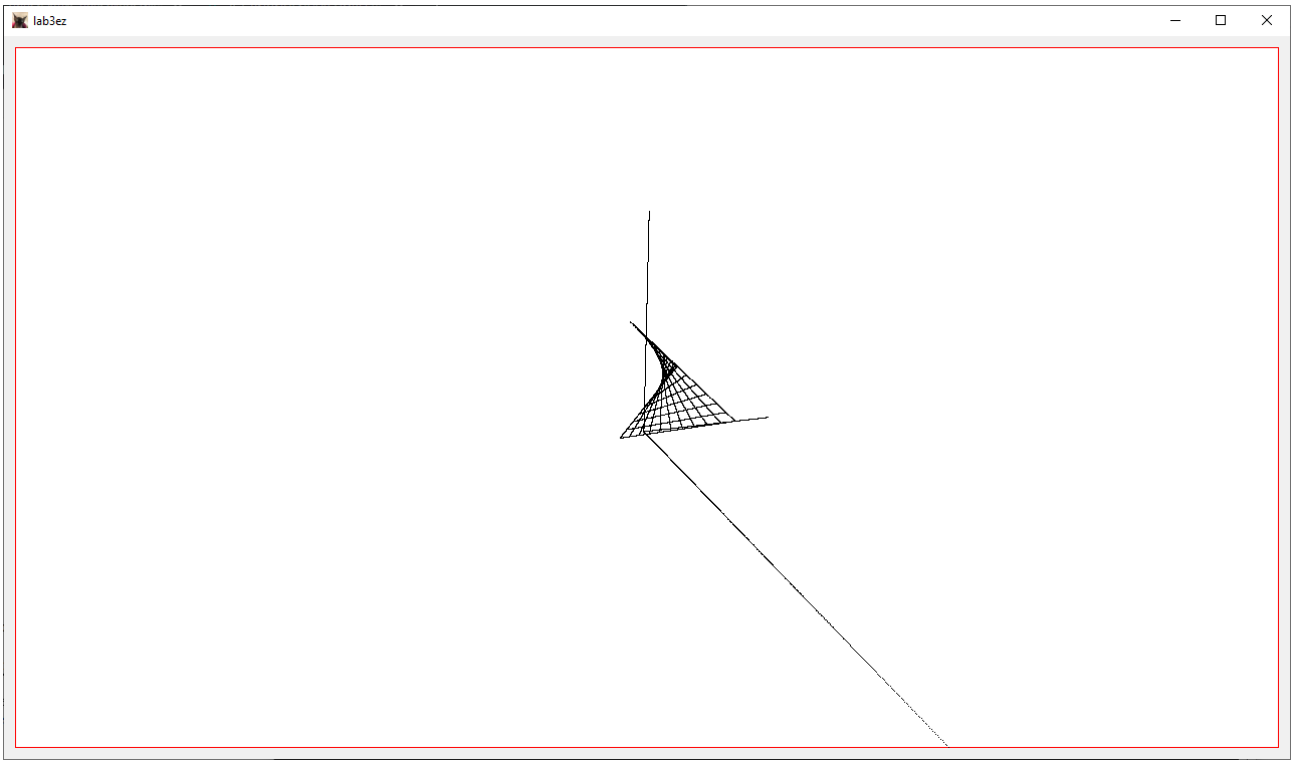
Пример 4. “0 0 0 20 0 0 0 20 0 20 20 0”. Вырожденный случай, плоскость



Пример 5. “0 0 20 20 20 20 20 0 0 0 20 0”



Пример 5. “0 0 20 20 20 20 20 0 0 0 20 0”. Вращение относительно OX



Пример 5. “0 0 20 20 20 20 20 0 0 0 20 0”. Вращение относительно OY

Вывод

При выполнении лабораторной работы были изучены формирования различных поверхностей с использованием её пространственного разворота и ортогонального проецирования на плоскость при её визуализации. В частности, исследована билинейная поверхность и ее построение в пространстве.