Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 10

Дисциплина Компьютерная графика

Тема Построение трехмерных поверхностей

Студент Хетагуров П.К.

Группа ИУ7-45

Оценка (баллы) _____

Преподаватель Куров А. В

Москва.
2020 г.

Цель работы:

Взучение и программная реализация алгоритма Плавающего горизонта построения трехмерных поверхностей

Задание:

Должна быть разработана программа, позволяющая осуществлять ввод пределов и шага изменения координат x, z, выбора уравнения поверхности из заранее сформированного списка, построение поверхности. Должен быть обеспечен поворот изображения (поверхности) вокруг каждой из трех координатных осей. Система координат должна быт неподвижной. Выполнить масштабирование для обеспечения размещения исходного изображения целиком в пределах поля вывода

Список уравнений поверхностей задается в отдельном модуле.

Идея алгоритма:

Заданную поверхность рассекаем плоскостями, перпендикулярными оси Z.
Эти плоскости имеют уравнения Z = const

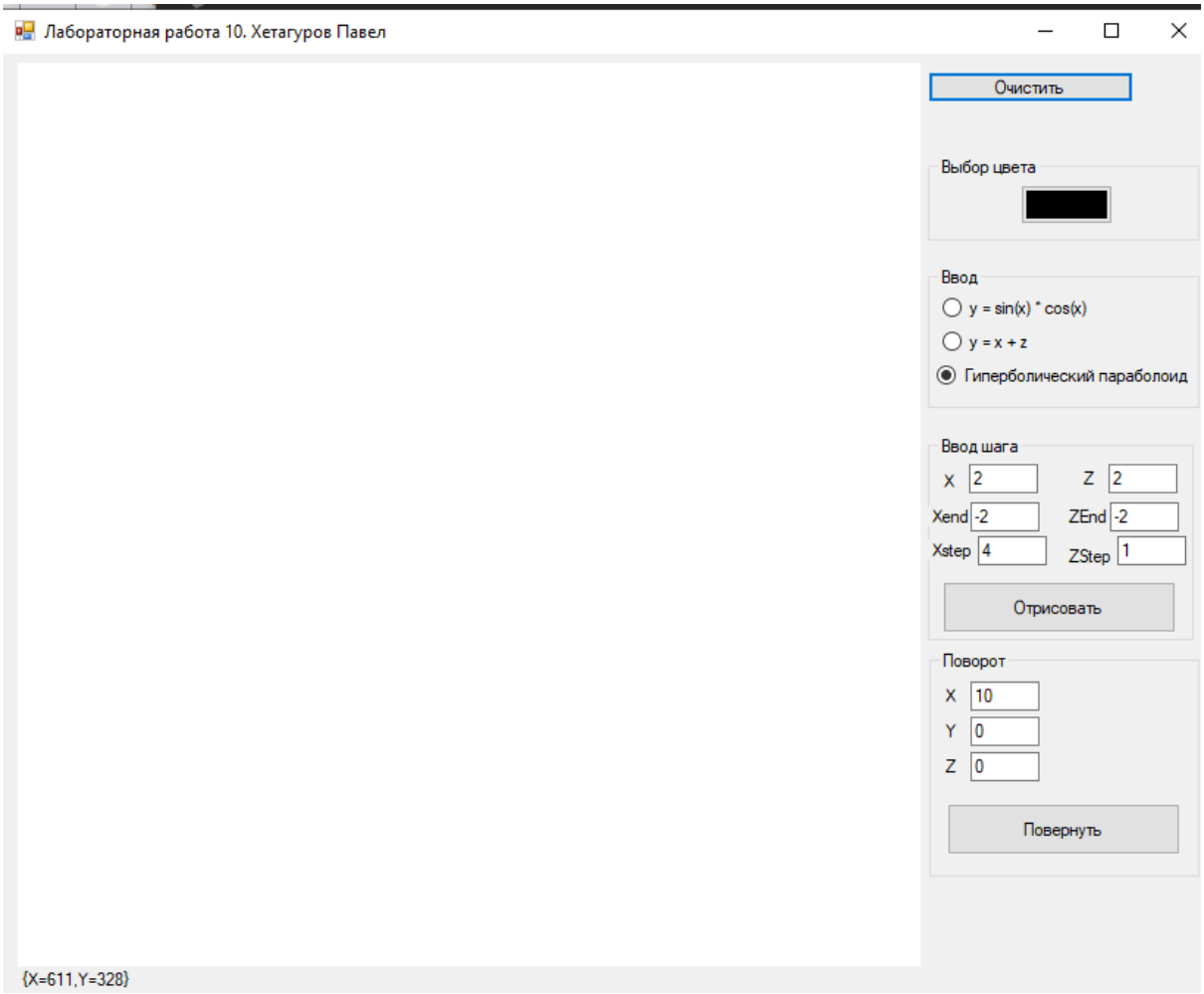
1) Рассматриваемая поверхность рассекается плоскостями, перпендикулярными оси Z. В каждом сечении получается кривая, описываемая уравнением y = f(x, z = const)
2) Полученные кривые можно спроектировать на плоскость x0y (z = 0). Изобразить видимые части каждой кривой. Изображение надо строить , начиная с кривой, полученной в ближайшем к наблюдателю сечении.

Кривая, полученная в сечении ближайшей плоскостью является видимой.
Кривая, полученная во втором сечении, тоже будет видима.
Вторая кривая расположена либо выше первой, либо ниже первой. Частный случай - кривые могут совпадать, тогда получим одну кривую.

Начиная с третьей кривой, надо решать задачу определения видимости точек кривой.
Задача решается в пространстве изображений (в экранной системе координат).

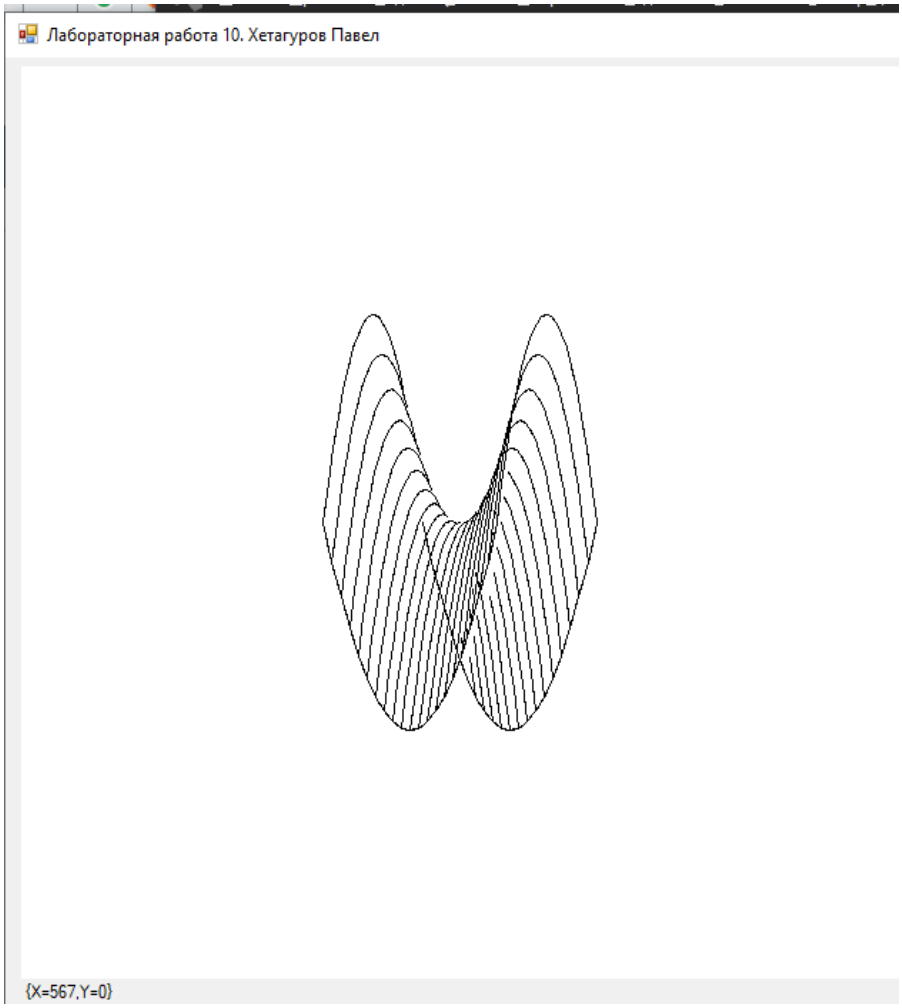
Практическая часть

Интерфейс:

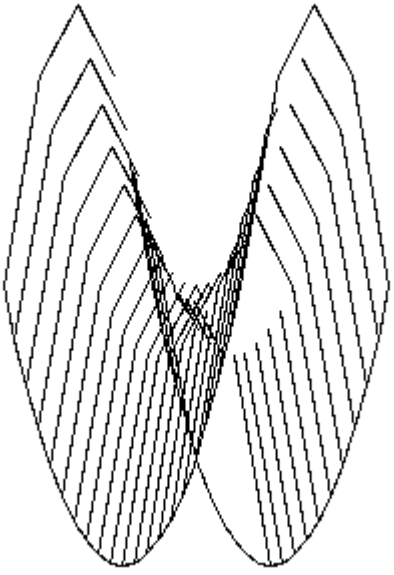


Примеры работы

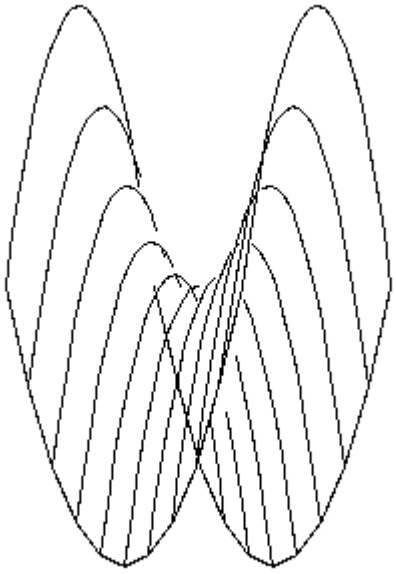
Гиперболический параболоид:



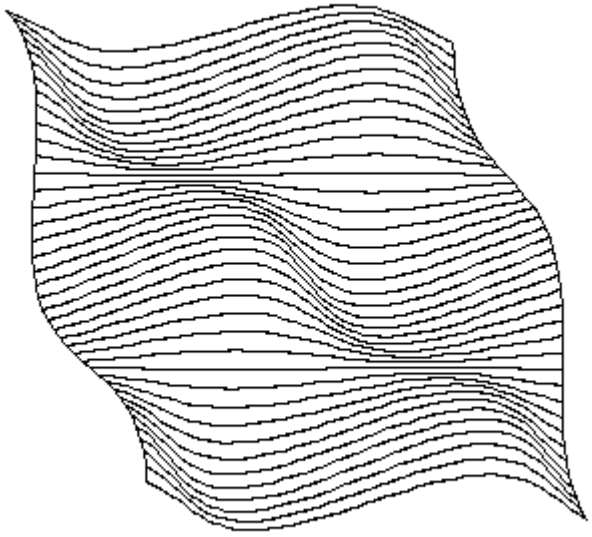
При измененияx шага можно получить разную детализацию:
при увеличении шага по X:



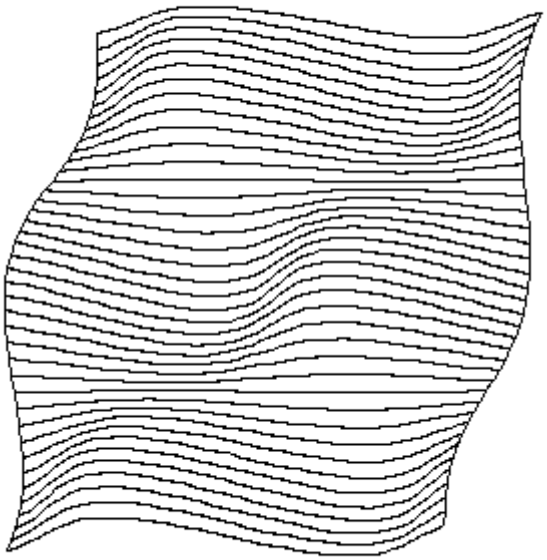
При увеличении шага по Z:

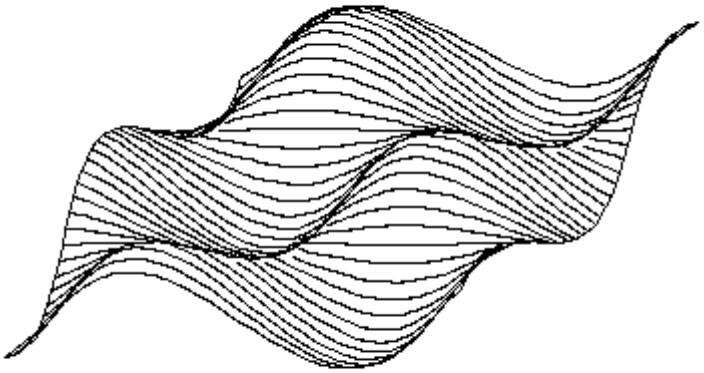


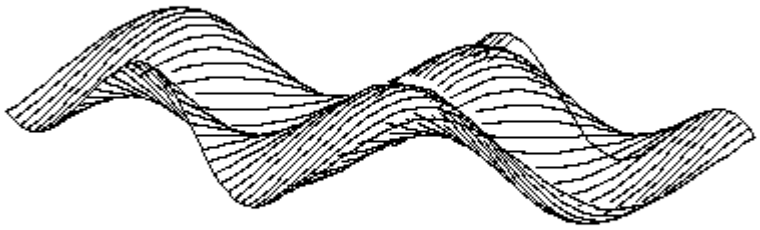
Поверхность y = sin(x) * cos(x)

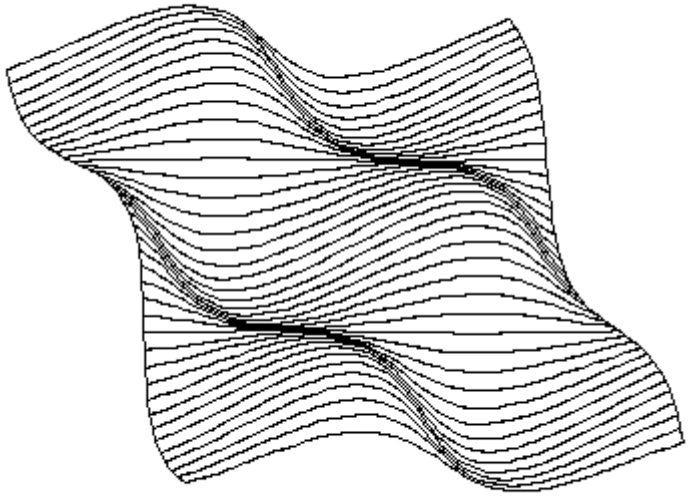


Повороты вокруг X с шагом 40 градусов:

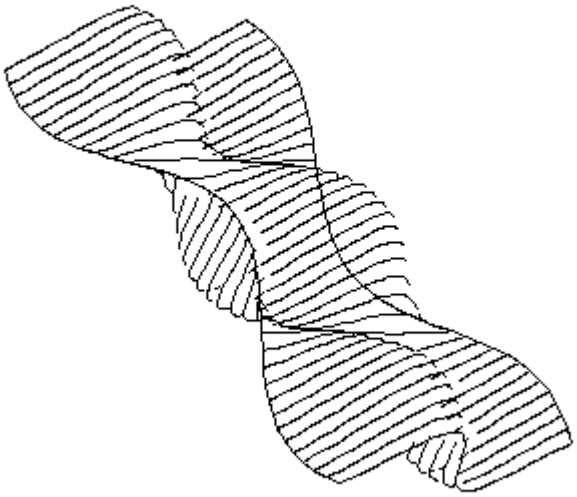


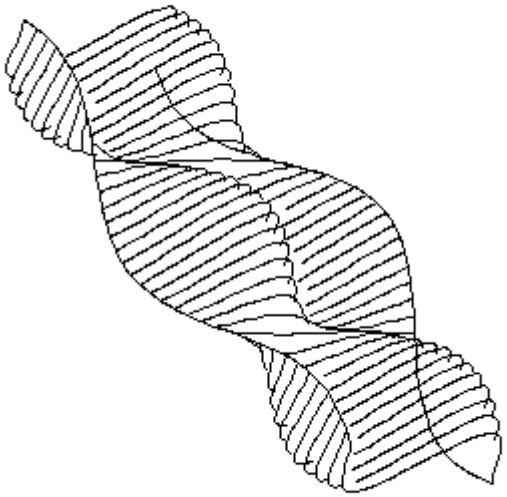


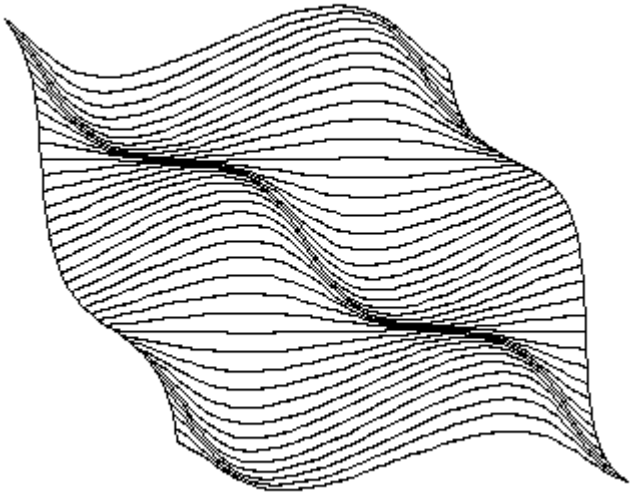




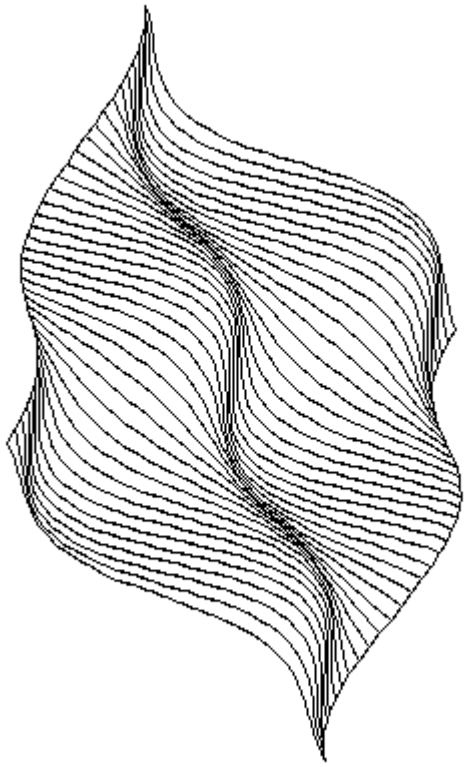
Вокруг Y:







Вокруг oZ:



Код:

```
        public void HorizontDraw(DotsDistance borderX, DotsDistance borderZ, double
ox, double oy, double oz, Size size, Graphics painter, Pen pen)
        {
            screenSize = size;
            PrepareArraysFill();  // заполняем массив top – нулями, массив bottom –
максимальным значением y

            Point left = new Point(-1, -1);
            Point right = new Point(-1, -1);
            Point previos = new Point(0, 0);

            // начинаем просмотр от ближайшей к наблюдателю
            for (double z = borderZ.max; z >= borderZ.min; z -= borderZ.step)
            {
                double yTemp = function2d(borderX.min, z);

                previos = Transform(borderX.min, yTemp, z, ox, oy, oz);  // обработка
поворотов
                ProcessEdge(previos, ref left, painter, pen);           // обработка
левого ребра
                int prevVisible = Visible(previos);

                for (double x = borderX.min; x <= borderX.max; x += borderX.step)
                {
                    Point current = new Point(0, 0);
                    Point intersection = new Point(0, 0);

                    yTemp = function2d(x, z);
                    current = Transform(x, yTemp, z, ox, oy, oz);     // для обработки
поворотов
                    int curVisible = Visible(current);
```

```
                    if (prevVisible == curVisible) // если видимость не изменилась
                    {
                        if (prevVisible != 0) // и точка видима
                        {
                            HorizonsUpdate(previos, current, painter, pen);  //
отрисовать участок и обновить массивы горизонтов
                        }
                    }
                    else if (curVisible == 0) // если точка стала невидима
                    {
                        if (prevVisible == 1)  // а была выше top
                        {
                            intersection = GetIntersection(previos, current, top);
                        }
                        else            // была ниже bottom
                        {
                            intersection = GetIntersection(previos, current, bottom);
                        }

                        // отрисовать
                        HorizonsUpdate(previos, intersection, painter, pen);
                    }
                    else if (curVisible == 1) // если точка стала видима и выше top
                    {
                        if (prevVisible == 0) // а была невидима
                        {
                            intersection = GetIntersection(previos, current, top);
                            HorizonsUpdate(previos, intersection, painter, pen);
                        }
                        else           // была ниже bottom (два пересечения)
                        {
                            intersection = GetIntersection(previos, current, top);
                            HorizonsUpdate(previos, intersection, painter, pen);
                            intersection = GetIntersection(previos, current, bottom);
                            HorizonsUpdate(intersection, current, painter, pen);
                        }
                    }
                    else   // аналогично предыдущему
                    {
                        if (prevVisible == 0)
                        {
                            intersection = GetIntersection(previos, current, bottom);
                            HorizonsUpdate(previos, intersection, painter, pen);
                        }
                        else
                        {
                            intersection = GetIntersection(previos, current, top);
                            HorizonsUpdate(previos, intersection, painter, pen);
                            intersection = GetIntersection(previos, current, bottom);
                            HorizonsUpdate(intersection, current, painter, pen);
                        }
                    }
                    prevVisible = curVisible; // видимость предыдущей точки
                    previos = current;
                }
                ProcessEdge(previos, ref right, painter, pen); // обработать правое
ребро

            }
        }
```

```
int Visible(Point dot)
    {
        int Visible = -1;

        if (dot.Y < top[dot.X] && dot.Y > bottom[dot.X])
        {
            Visible = 0;
        }
        if (dot.Y >= top[dot.X])
        {
            Visible = 1;
        }

        return Visible;
    }

void ProcessEdge(Point current, ref Point past, Graphics painter, Pen pen)
    {
        if (past.X != -1) // если точка не первая, соединить с предыдущей
        {
             painter.DrawLine(pen, current, past);
        }
        past = current;
    }
```