|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 Изучение и программная реализация алгоритма Плавающего горизонта построения трехмерных поверхностей**  **Студент Иванов В.А.**  **Группа ИУ7-42Б**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель Куров А.В.** |  |

Москва.

2020 г.

**Цель работы**

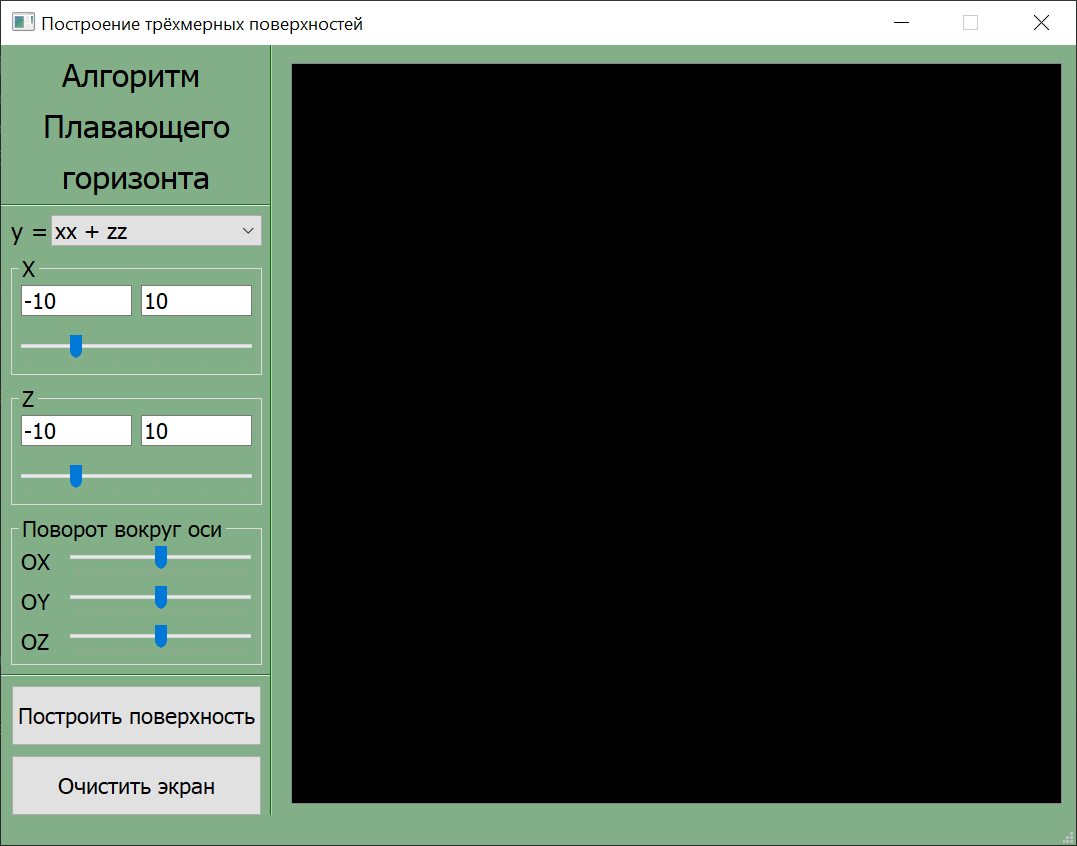
Программная реализация алгоритма Плавающего горизонта

**Описание задания**

1. Реализовать алгоритм Плавающего горизонта
2. Обеспечить возможность границ и шага по OX, OZ
3. Реализовать поворот фигуры по 3 осям

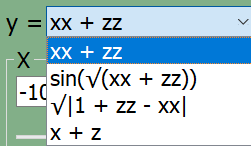
**Описание графического интерфейса**

Для использования функционала поставленных задач, был создан графический интерфейс

****

Интерфейс предоставляет возможность:

* Построение поверхности по выбранной функции
* Выбора одной из следующих функций

****

* Ввод диапазона и шага изменения x, z
* Задание угла поворота по осям OX, OY, OZ от -180 до 180 градусов

**Описание и реализация алгоритма**

Задачей данной лабораторной работы служит выполнение визуализации плоскости, заданной уравнением, используя алгоритм Плавающего горизонта.

Идея алгоритма в сведении задачи построения трёхмерного изображения к двухмерному. Для этого используются секущие плоскости имеющие постоянные координаты по z. В данных секущих плоскостях производится поиск значений y для х в заданном диапазоне и шаге. После этого последовательность точек (x, y) соединяется отрезками на экране. После этого производится переход к следующему значению z, также с заданным шагом.

Главной задачей алгоритма является определение и удаление невидимых участков спроецированных линий. Эта задача осуществляется при помощи “плавающего горизонта”. Идея заключается в том, что для каждого x на экране определяется верхний и нижний горизонт – максимальное и минимальное значение y. Сами ломаные, полученные сечением плоскости высвечиваются на экран в следующем порядке – сначала те z, которые ближе к наблюдателю, то есть большие, а потом более дальние. Видимыми будут те участки сечений, которые не загорожены ранее идущими. Поэтому при отрисовке очередного сечения учитываются значения верхнего и нижнего горизонта. В случае, если некая точка лежит выше верхнего или ниже нижнего горизонта, то она будет видима, поэтому она будет отрисована, а также обновится значение соответствующего горизонта в данной точке. Иначе, точка считается невидимой.

Данный процесс отрисовки каждой точки сечения производится за счёт отрезков, соединяющих последовательность точек (x, y) полученных при сечении. В случае, если концы отрезка имеют одинаковую видимость: над верхним горизонтом или под нижним, то отрезок отрисовывается полностью, а также каждая точка отрезка обновляет значения горизонта. В случае, если обе точки являются невидимыми, отрезок не отрисовывается.

В ином случае, когда точки имеют различную видимость, можно сделать вывод о том, что отрезок видим лишь частично, а следовательно, имеет точку пересечения с горизонтом. В случае, если одна из точек невидима, то пересечение будет лишь с одним горизонтом. Иначе, отрезок будет иметь пересечение сразу с двумя горизонтами, и потребуется отрисовать сразу два отрезка. Точка пересечения находится путём прохода по отрезку до точки, которая окажется по сторону от горизонта, отличную от положения начала отрезка.

Преобразование поворота и масштабирования под экран производится после получения значений x, z и y=f(x,z). Для корректного отображения требуется найти предельные значения для x и y. Для этого до начала алгоритма отрисовки производится анализ всех узлов секущей стеки.

Для лучшего восприятия, также крайние точки каждого сечения соединены боковыми отрезками.

**Программная реализация:**

Отрисовка поверхности:  
def draw\_surface(self, func, angles, x\_par, z\_par):  
 self.angles = angles  
 x\_minmax, y\_minmax = self.find\_minmax(func, x\_par, z\_par)  
  
 x\_min, x\_max, x\_step = x\_par  
 z\_min, z\_max, z\_step = z\_par  
  
 self.min\_hor = [self.y\_res] \* self.x\_res  
 self.max\_hor = [0] \* self.x\_res  
 x\_l, y\_l, x\_r, y\_r = [-1] \* 4  
  
 for z in frange(z\_max, z\_min, -z\_step):  
 x\_pre = x\_min  
 y\_pre = func(x\_min, z)  
 x\_pre, y\_pre = self.transform(x\_pre, y\_pre, z, x\_minmax, y\_minmax)  
  
 x\_l, y\_l = self.edge\_process(x\_pre, y\_pre, x\_l, y\_l)  
 pre\_flag = self.is\_visible(x\_pre, y\_pre)  
  
 for x0 in frange(x\_min, x\_max, x\_step):  
 y = func(x0, z)  
 x, y = self.transform(x0, y, z, x\_minmax, y\_minmax)  
  
 temp\_flag = self.is\_visible(x, y)  
 if pre\_flag == temp\_flag:  
 if temp\_flag != 0:  
 self.draw\_line((x\_pre, y\_pre), (x, y))  
 self.horizon(x\_pre, y\_pre, x, y)  
 else:  
 if temp\_flag == 0:  
 if pre\_flag == 1:  
 cross\_p = self.edge\_cross(x\_pre, y\_pre, x, y, self.max\_hor)  
 else:  
 cross\_p = self.edge\_cross(x\_pre, y\_pre, x, y, self.min\_hor)  
 self.draw\_line((x\_pre, y\_pre), cross\_p)  
 self.horizon(x\_pre, y\_pre, \*cross\_p)  
  
 elif temp\_flag == 1:  
 if pre\_flag == 0:  
 cross\_p = self.edge\_cross(x\_pre, y\_pre, x, y, self.max\_hor)  
 self.draw\_line(cross\_p, (x, y))  
 self.horizon(\*cross\_p, x, y)  
 else:  
 cross\_p = self.edge\_cross(x\_pre, y\_pre, x, y, self.min\_hor)  
 self.draw\_line((x\_pre, y\_pre), cross\_p)  
 self.horizon(x\_pre, y\_pre, \*cross\_p)  
  
 cross\_p = self.edge\_cross(x\_pre, y\_pre, x, y, self.max\_hor)  
 self.draw\_line(cross\_p, (x, y))  
 self.horizon(\*cross\_p, x, y)  
 else:  
 if pre\_flag == 0:  
 cross\_p = self.edge\_cross(x\_pre, y\_pre, x, y, self.min\_hor)  
 self.draw\_line((x, y), cross\_p)  
 self.horizon(\*cross\_p, x, y)  
 else:  
 cross\_p = self.edge\_cross(x\_pre, y\_pre, x, y, self.max\_hor)  
 self.draw\_line((x\_pre, y\_pre), cross\_p)  
 self.horizon(x\_pre, y\_pre, \*cross\_p)  
  
 cross\_p = self.edge\_cross(x\_pre, y\_pre, x, y, self.min\_hor)  
 self.draw\_line(cross\_p, (x, y))  
 self.horizon(\*cross\_p, x, y)  
 pre\_flag = temp\_flag  
 x\_pre = x  
 y\_pre = y  
  
 x\_r, y\_r = self.edge\_process(x, y, x\_r, y\_r)

Обработка крайних рёбер

def edge\_process(self, x, y, x\_edge, y\_edge):  
 if x\_edge != -1:  
 self.horizon(x\_edge, y\_edge, x, y)  
 self.draw\_line((x\_edge, y\_edge), (x, y))  
 return x, y

Обновление горизонтов  
def horizon(self, x1, y1, x2, y2):  
 x1, x2 = int(x1), int(x2)  
 if x2 == x1:  
 self.max\_hor[x2] = max(self.max\_hor[x2], y2)  
 self.min\_hor[x2] = min(self.min\_hor[x2], y2)  
 else:  
 k = (y2 - y1)/(x2 - x1)  
 for x in range(x1, x2):  
 y = int(k \* (x - x1) + y1)  
 self.max\_hor[x] = max(self.max\_hor[x], y)  
 self.min\_hor[x] = min(self.min\_hor[x], y)

Определение видимости

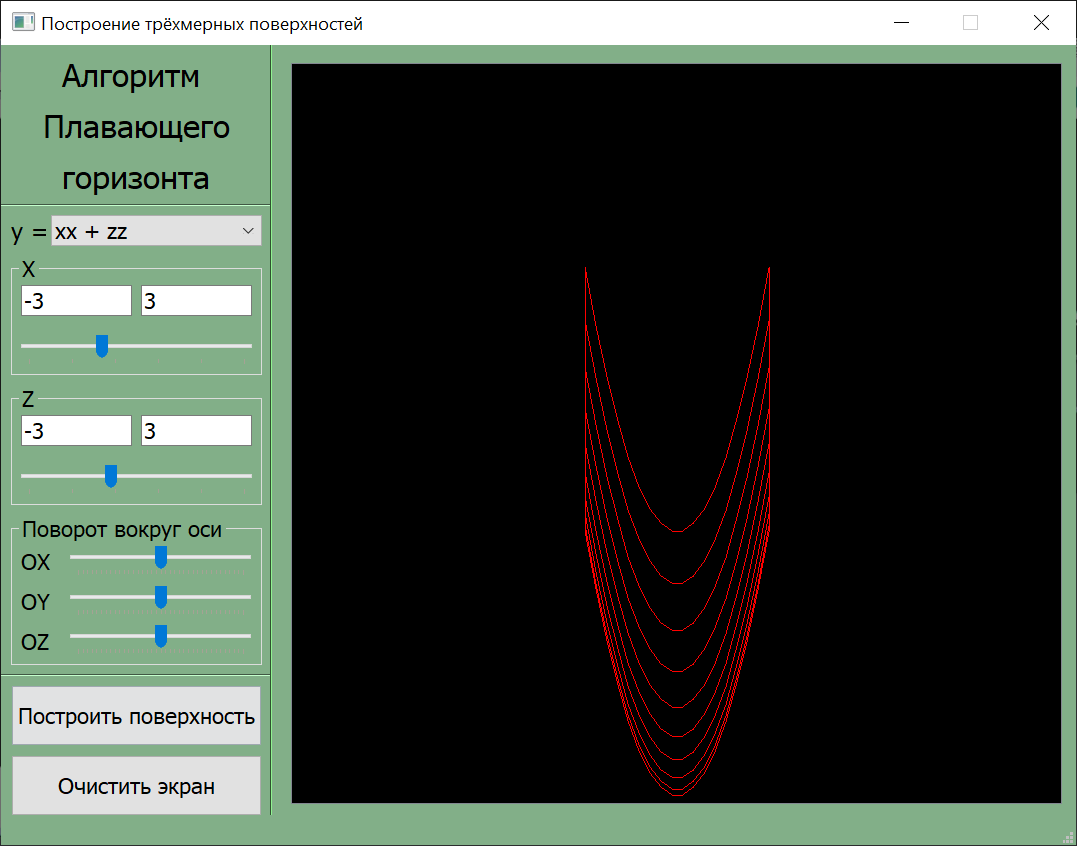
def is\_visible(self, x, y):  
 x = int(x)  
 if y < self.max\_hor[x] and y > self.min\_hor[x]: return 0  
 elif y >= self.max\_hor[x]: return 1  
 else: return -1

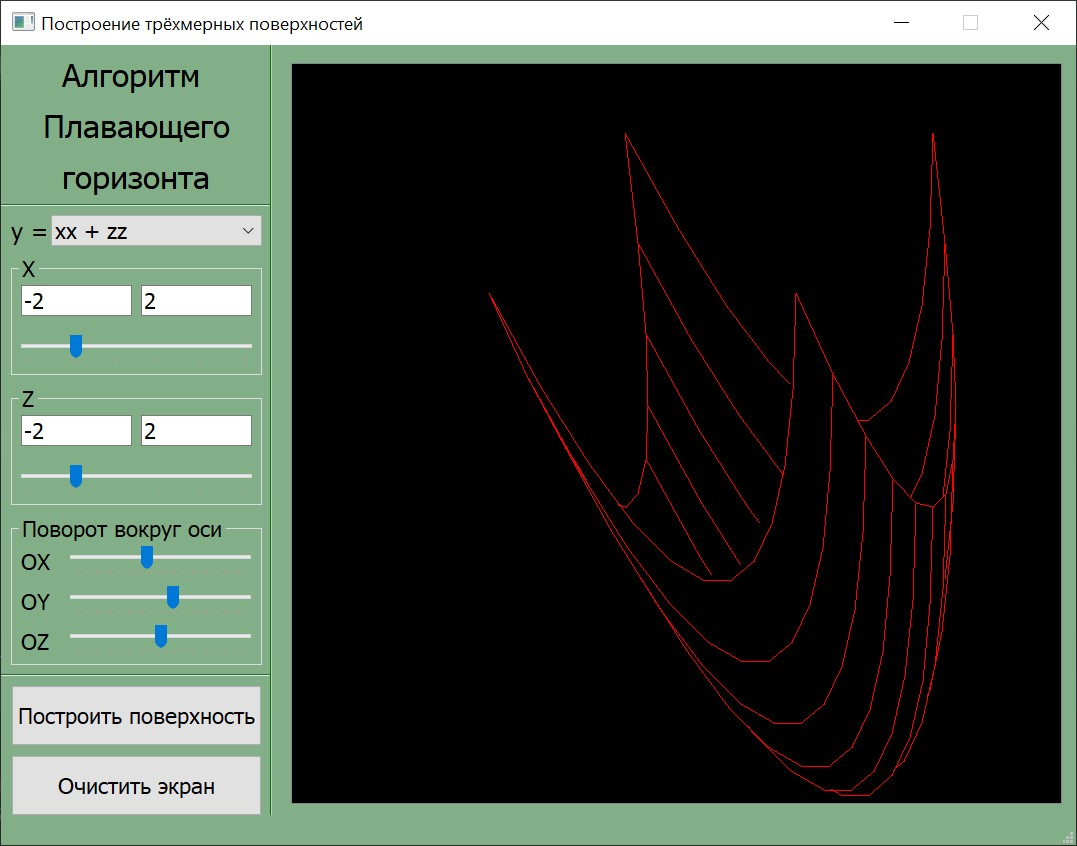
Нахождение пересечения отрезка с горизонтом

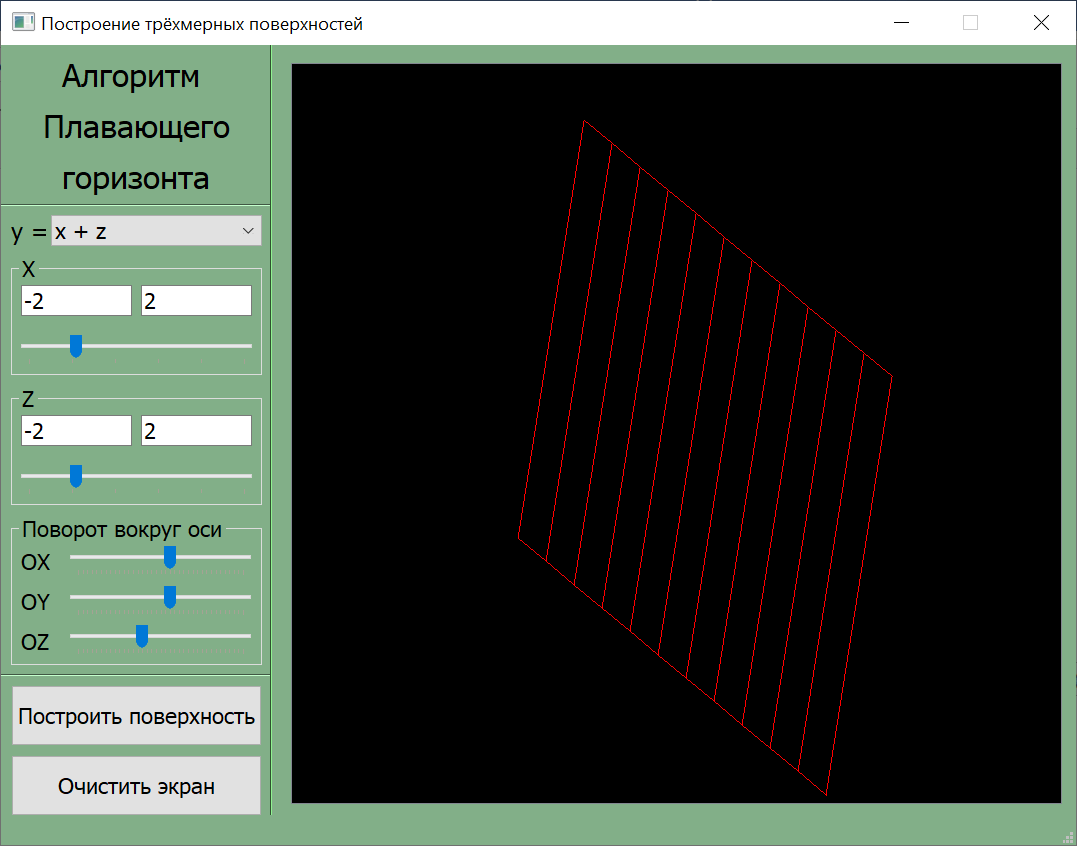
def edge\_cross(self, x1, y1, x2, y2, hor\_arr):  
 x1, x2 = int(x1), int(x2)  
 if x1 == x2:  
 xi = x2  
 yi = hor\_arr[x2]  
 else:  
 dx = x2 - x1  
 dy = y2 - y1  
 k = dy / dx  
  
 ysign = sign(y1 + k - hor\_arr[x1 + 1])  
 csign = ysign  
  
 yi = y1  
 xi = x1  
 while ysign == csign and xi < x2:  
 yi += k  
 xi += 1  
 csign = sign(yi - hor\_arr[xi])  
 if fabs(yi - k - hor\_arr[xi-1]) <= fabs(yi - hor\_arr[xi-1]):  
 yi -= k  
 xi -= 1  
 return (xi, yi)

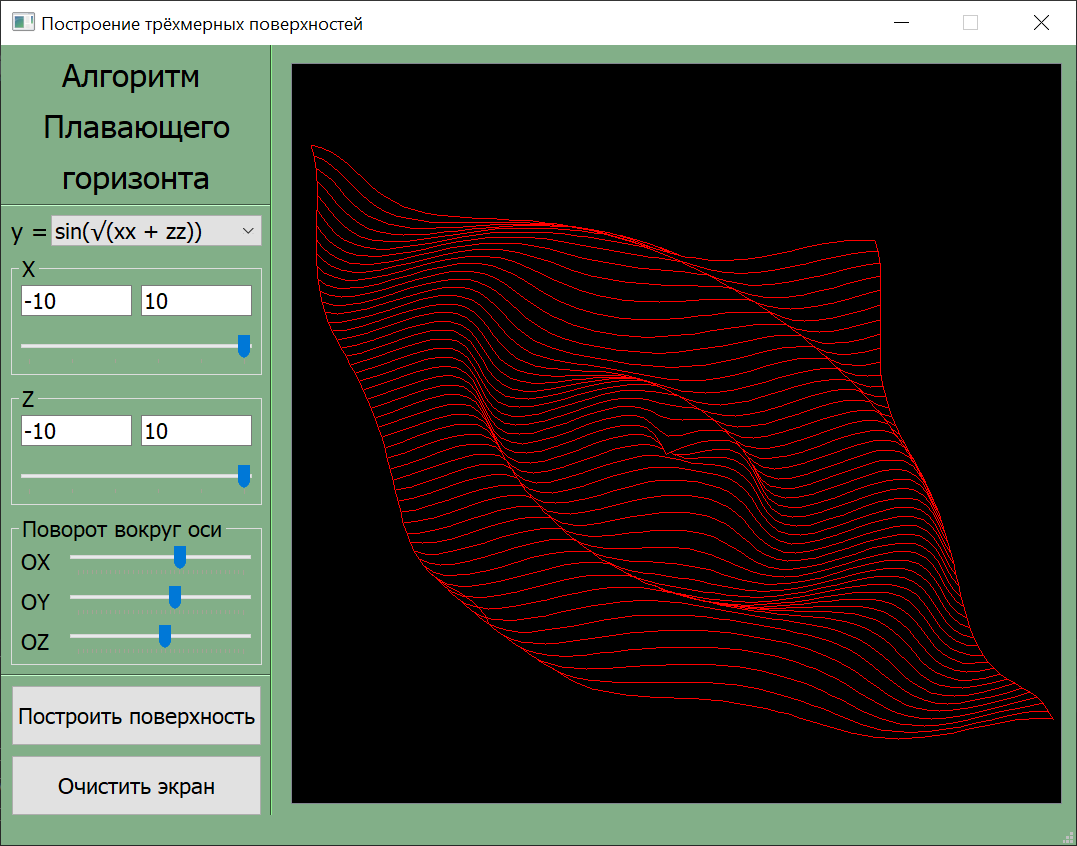
**Визуальные характеристики**

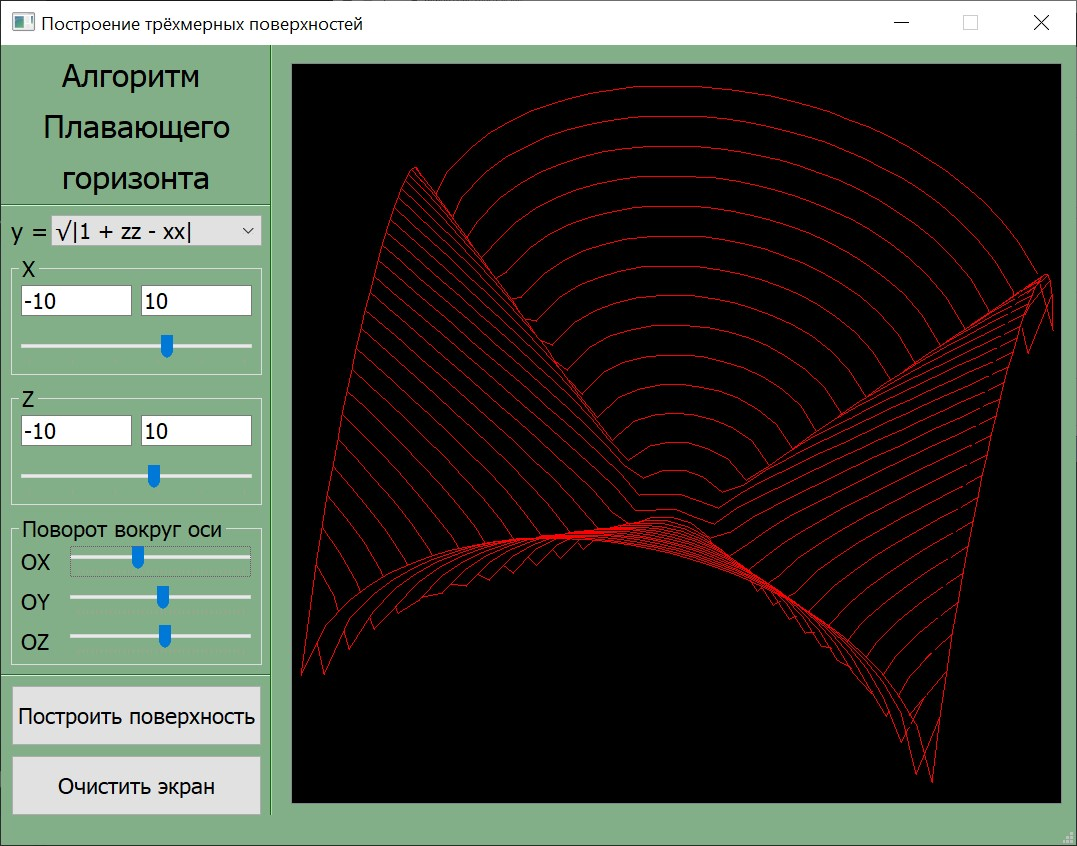
Ниже приведены примеры работы программы в различных ситуациях:











**Заключение**

В ходе лабораторной работы был изучен и реализован алгоритм построения трёхмерных поверхностей – Плавающий горизонт