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
по дисциплине «Компьютерная графика»

Тема: Реализация алгоритма плавающего горизонта построения трехмерных поверхностей.

Студент: Блохин Дмитрий

Группа: ИУ7-42Б

Москва.
2020 г.

Цель работы: изучение 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 быть неподвижной.

-Выполнить масштабирование для обеспечения размещения исходного изображения целиком в пределах поля вывода.

-Список уравнений поверхностей задается в отдельном модуле.

Ход работы:

Алгоритм плавающего горизонта предназначен для построения трехмерных поверхностей задаваемых уравнением вида $F(x,y,z) = 0$.

Алгоритм плавающего горизонта:

Исходные данные: задать поверхность, задать пределы изменения по координате z, шаг которым мы расставляем секущую плоскость, задать пределы изменения по координате x, шаг изменения абсцисс.

1. Инициализировать начальными значениями массивы горизонтов.
2. Для каждой секущей плоскости z = const выполнить следующие действия:
2.1. Обработать левое боковое ребро (Если очередная точка является первой точкой первой кривой, то запомнить ее P. Если очередная точка является первой точкой не первой кривой, то соединить ее с точкой P и запомнить ее в P).
2.2. Если ордината точки текущей кривой больше максимума или меньше минимума, то точка видима, в противном случае точка невидима.
2.3. Если видимость сегмента кривой изменилась, то найти точку пересечения с горизонтом.
2.4. Если сегмент кривой видим, то изобразить его целиком.
2.5. Если видимость изменилась и текущая точка невидима, то изобразить участок кривой от предыдущей точки до точки пересечения.
2.6. Если видимость сегмента изменилась и текущая точка стала видимой, то изобразить участок кривой от точки пересечения до текущей точки.
2.7. Скорректировать массивы верхнего и нижнего горизонтов.
2.8. Обработать правое боковое ребро.

Реализация:

Используемые подпрограммы:

1. **Подпрограмма заполнения массивов плавающих горизонтов**

subroutine.py - /Users/demonblo/Downloads/Computer-graphics-master/lab10/subroutine.py (3.7.3)

```
from PyQt5.QtCore import Qt


def sign(x):
    if not x:
        return 0
    else:
        return x / abs(x)


def horizon(x1, y1, x2, y2, top, bottom, image):
    x = x1
    y = y1
    dx = x2 - x1
    dy = y2 - y1
    sx = sign(dx)
    sy = sign(dy)
    #print(x1, y1, x2, y2, "\n")
    dx = abs(dx)
    dy = abs(dy)

    # если точка
    if dx == 0 and dy == 0 and 0 <= x < image.width():
        if y >= top[x]:
            top[x] = y
            image.setPixel(x, image.height() - y, Qt.white)

        if y <= bottom[x]:
            bottom[x] = y
            image.setPixel(x, image.height() - y, Qt.white)

        return top, bottom

    #  Нужно ли менять местами x и y
    change = 0
    if dy > dx:
        dx, dy = dy, dx
        change = 1

    y_max_curr = top[x]
    y_min_curr = bottom[x]
    e = 2 * dy - dx

    i = 1
    while i <= dx:
        if 0 <= x < image.width():

            if y >= top[x]:
                if y >= y_max_curr:
                    y_max_curr = y
                image.setPixel(x, image.height() - y, Qt.white)

            if y <= bottom[x]:
                if y <= y_min_curr:
                    y_min_curr = y
                image.setPixel(x, image.height() - y, Qt.white)
```

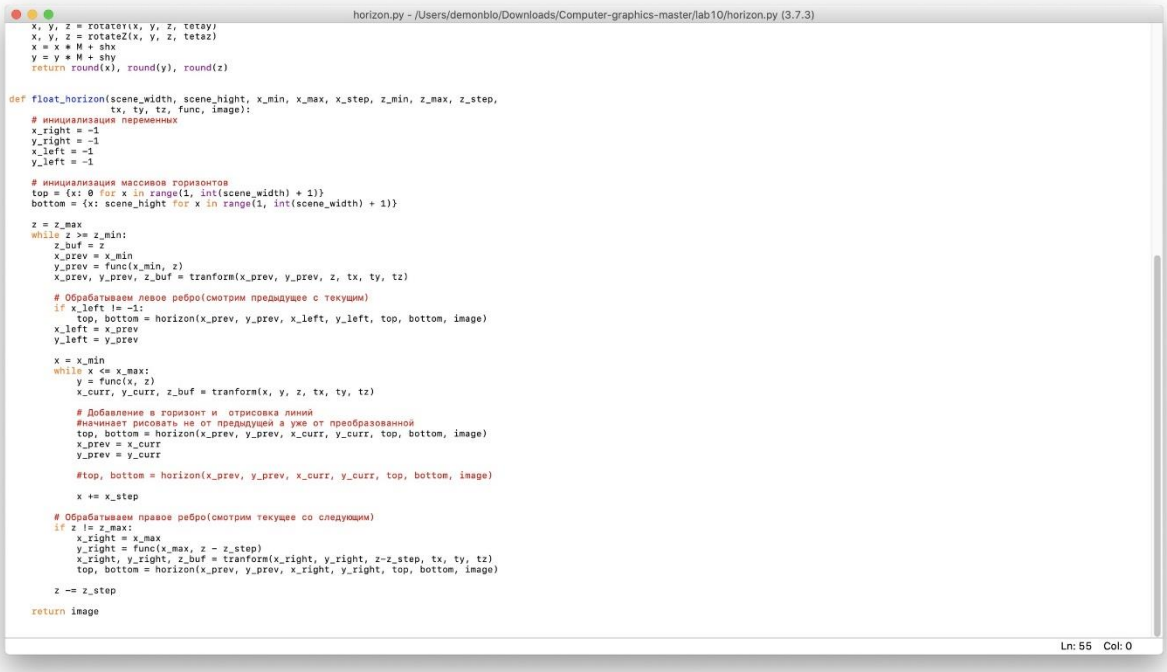
Ln: 19 Col: 16

subroutine.py - /Users/demonblo/Downloads/Computer-graphics-master/lab10/subroutine.py (3.7.3)

```
    #  Нужно ли менять местами x и y
    change = 0
    if dy > dx:
        dx, dy = dy, dx
        change = 1

    y_max_curr = top[x]
    y_min_curr = bottom[x]
    e = 2 * dy - dx

    i = 1
    while i <= dx:
        if 0 <= x < image.width():

            if y >= top[x]:
                if y >= y_max_curr:
                    y_max_curr = y
                image.setPixel(x, image.height() - y, Qt.white)

            if y <= bottom[x]:
                if y <= y_min_curr:
                    y_min_curr = y
                image.setPixel(x, image.height() - y, Qt.white)

        if e >= 0:
            if change:
                top[x] = y_max_curr
                bottom[x] = y_min_curr

                x += sx

                y_max_curr = top[x]
                y_min_curr = bottom[x]

            else:
                y += sy

            e -= 2 * dx
        if e < 0:
            if not change:
                top[x] = y_max_curr
                bottom[x] = y_min_curr

                x += sx

                y_max_curr = top[x]
                y_min_curr = bottom[x]

            else:
                y += sy

            e += 2 * dy

        i += 1

    return top, bottom
```

Ln: 19 Col: 16

Реализация алгоритма плавающего горизонта и алгоритмы вращения и преобразования:

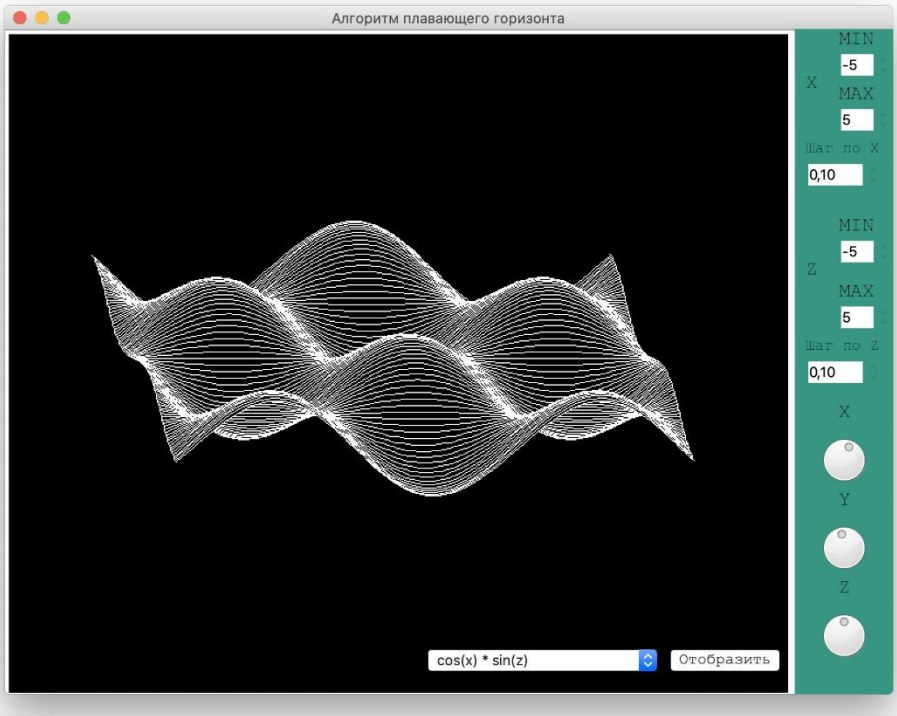


```
    x, y, z = rotateY(x, y, z, tetay)
    x, y, z = rotateZ(x, y, z, tetaz)
    x = x * M + shx
    y = y * M + shy
    return round(x), round(y), round(z)


def float_horizon(scene_width, scene_hight, x_min, x_max, x_step, z_min, z_max, z_step,
                  tx, ty, tz, func, image):
    # инициализация переменных
    x_right = -1
    y_right = -1
    x_left = -1
    y_left = -1

    # инициализация массивов горизонтов
    top = {x: 0 for x in range(1, int(scene_width) + 1)}
    bottom = {x: scene_hight for x in range(1, int(scene_width) + 1)}

    z = z_max
    while z >= z_min:
        z_buf = z
        x_prev = x_min
        y_prev = func(x_min, z)
        x_prev, y_prev, z_buf = tranform(x_prev, y_prev, z, tx, ty, tz)

        # Обрабатываем левое ребро(смотрим предыдущее с текущим)
        if x_left != -1:
            top, bottom = horizon(x_prev, y_prev, x_left, y_left, top, bottom, image)
        x_left = x_prev
        y_left = y_prev

        x = x_min
        while x <= x_max:
            y = func(x, z)
            x_curr, y_curr, z_buf = tranform(x, y, z, tx, ty, tz)

            # Добавление в горизонт и  отрисовка линий
            #начинает рисовать не от предыдущей а уже от преобразованной
            top, bottom = horizon(x_prev, y_prev, x_curr, y_curr, top, bottom, image)
            x_prev = x_curr
            y_prev = y_curr

            #top, bottom = horizon(x_prev, y_prev, x_curr, y_curr, top, bottom, image)

            x += x_step

        # Обрабатываем правое ребро(смотрим текущее со следующим)
        if z != z_max:
            x_right = x_max
            y_right = func(x_max, z - z_step)
            x_right, y_right, z_buf = tranform(x_right, y_right, z-z_step, tx, ty, tz)
            top, bottom = horizon(x_prev, y_prev, x_right, y_right, top, bottom, image)

        z -= z_step

    return image
```

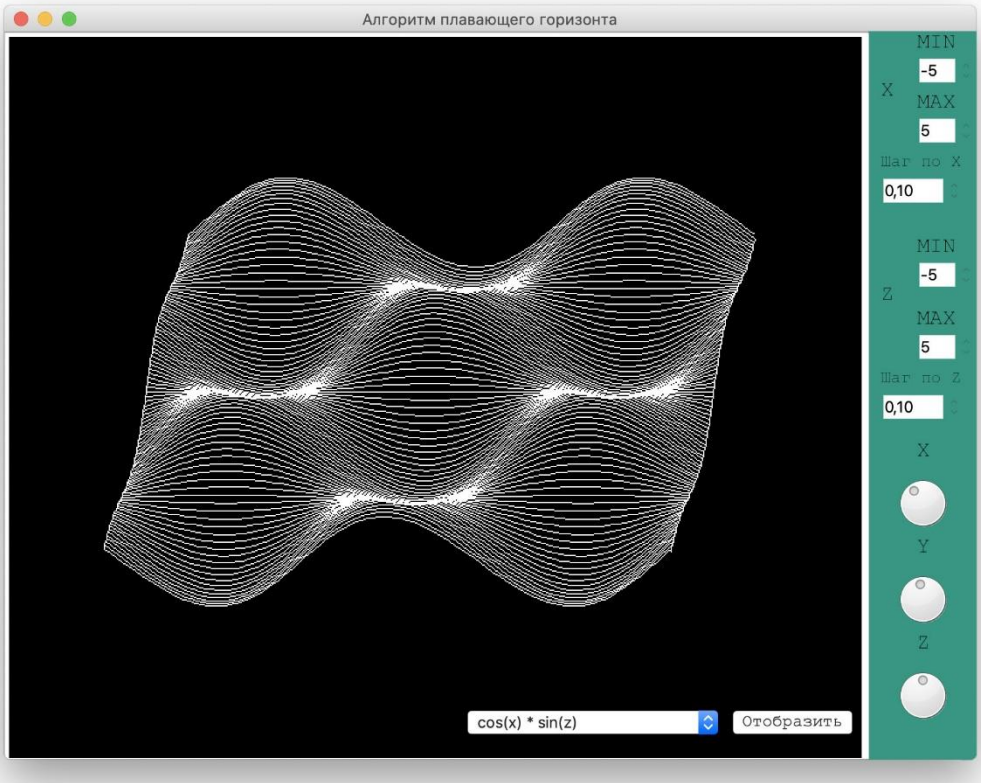

```
M = 48
shx = 600 / 2 + 50
shy = 710 / 2 - 50


def rotateX(x, y, z, teta):
    teta = teta * pi / 180
    buf = y
    y = cos(teta) * y - sin(teta) * z
    z = cos(teta) * z + sin(teta) * buf
    return x, y, z


def rotateY(x, y, z, teta):
    teta = teta * pi / 180
    buf = x
    x = cos(teta) * x - sin(teta) * z
    z = cos(teta) * z + sin(teta) * buf
    return x, y, z


def rotateZ(x, y, z, teta):
    teta = teta * pi / 180
    buf = x
    x = cos(teta) * x - sin(teta) * y
    y = cos(teta) * y + sin(teta) * buf
    return x, y, z


def tranform(x, y, z, tetax, tetay, tetaz):
    x, y, z = rotateX(x, y, z, tetax)
    x, y, z = rotateY(x, y, z, tetay)
    x, y, z = rotateZ(x, y, z, tetaz)
    x = x * M + shx
    y = y * M + shy
    return round(x), round(y), round(z)


def float_horizon(scene_width, scene_hight, x_min, x_max, x_step, z_min, z_max, z_step,
                  tx, ty, tz, func, image):
    # инициализация переменных
    x_right = -1
    y_right = -1
    x_left = -1
    y_left = -1

    # инициализация массивов горизонтов
    top = {x: 0 for x in range(1, int(scene_width) + 1)}
    bottom = {x: scene_hight for x in range(1, int(scene_width) + 1)}

    z = z_max
    while z >= z_min:
        z_buf = z
        x_prev = x_min
        y_prev = func(x_min, z)
        x_prev, y_prev, z_buf = tranform(x_prev, y_prev, z, tx, ty, tz)

        # Обрабатываем левое ребро(смотрим предыдущее с текущим)
        if x_left != -1:
```

Демонстрация работы программы:

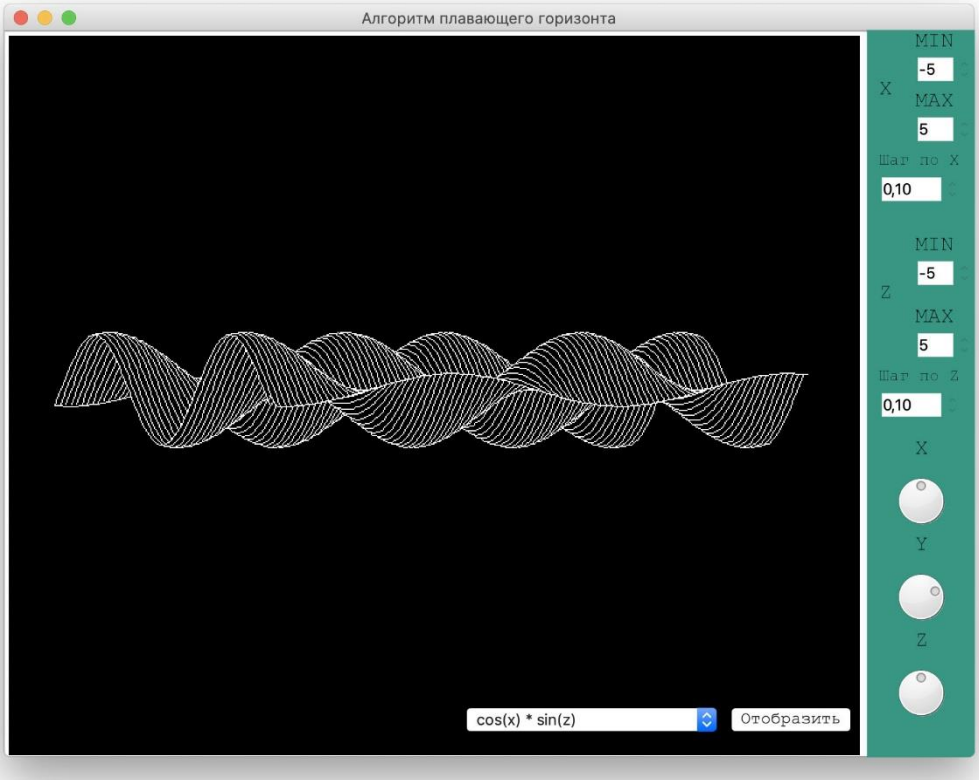
Зададим cos(x) * sin(z)



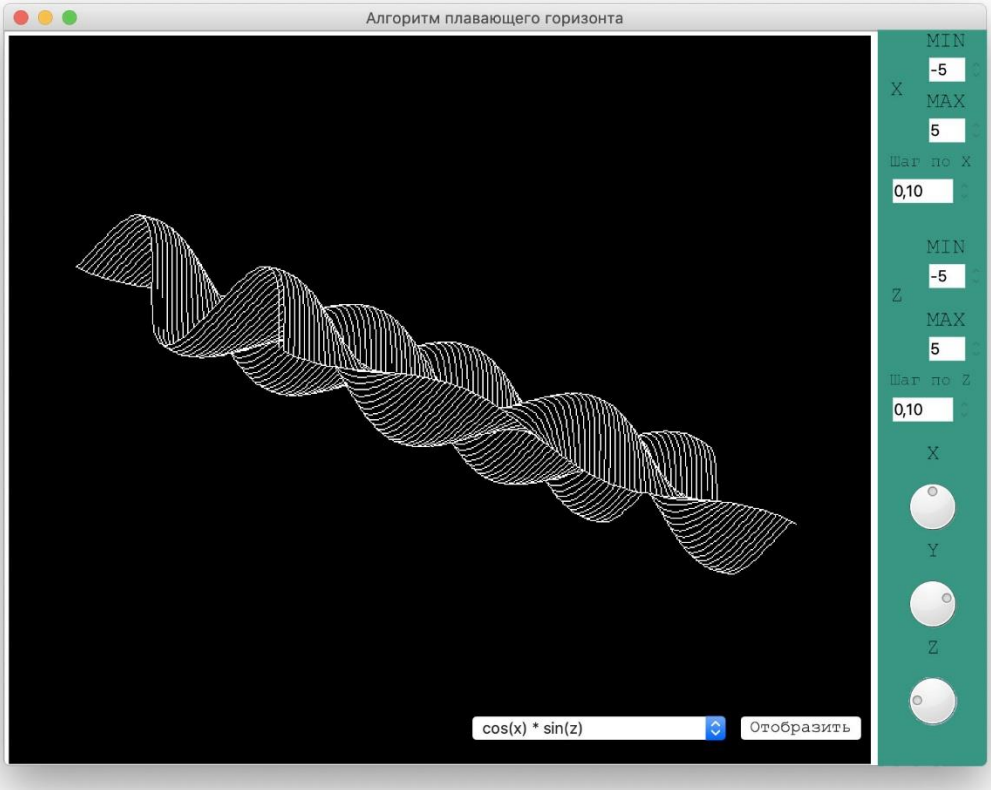
Вращение по OX



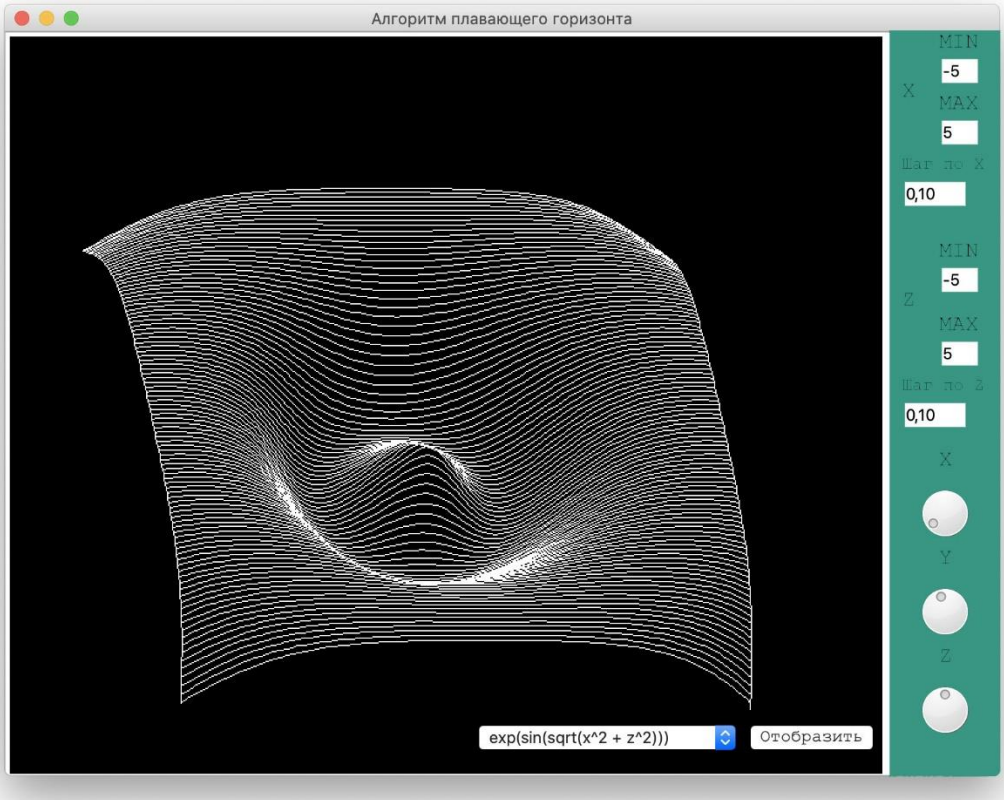
Вращение по OY



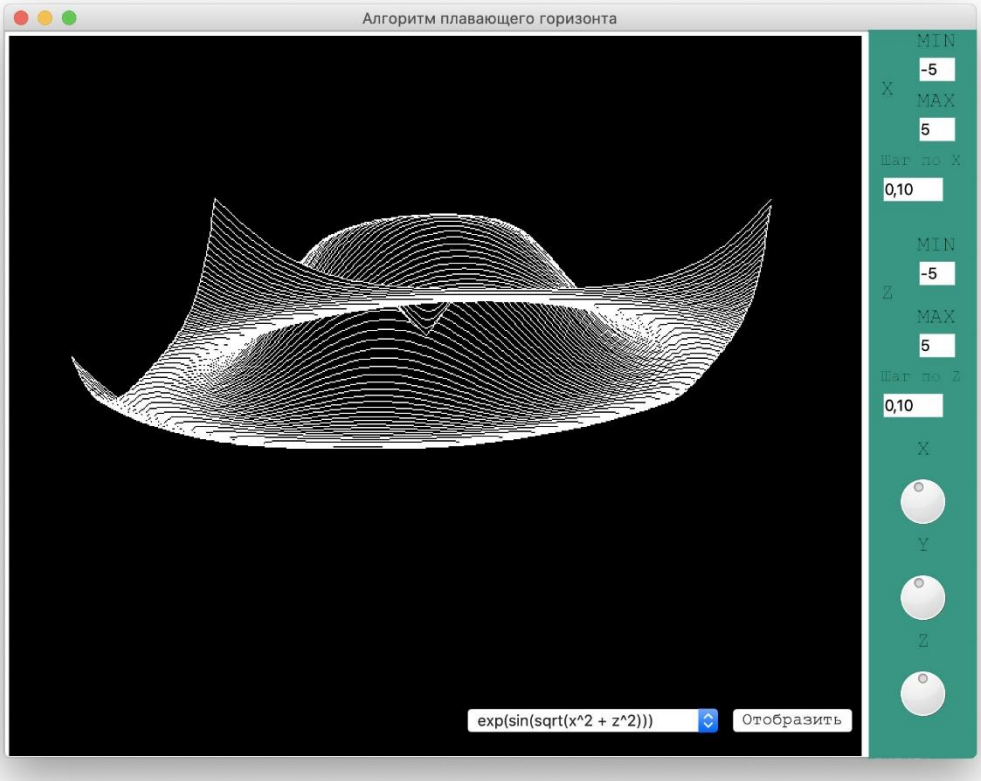
Вращение по OZ



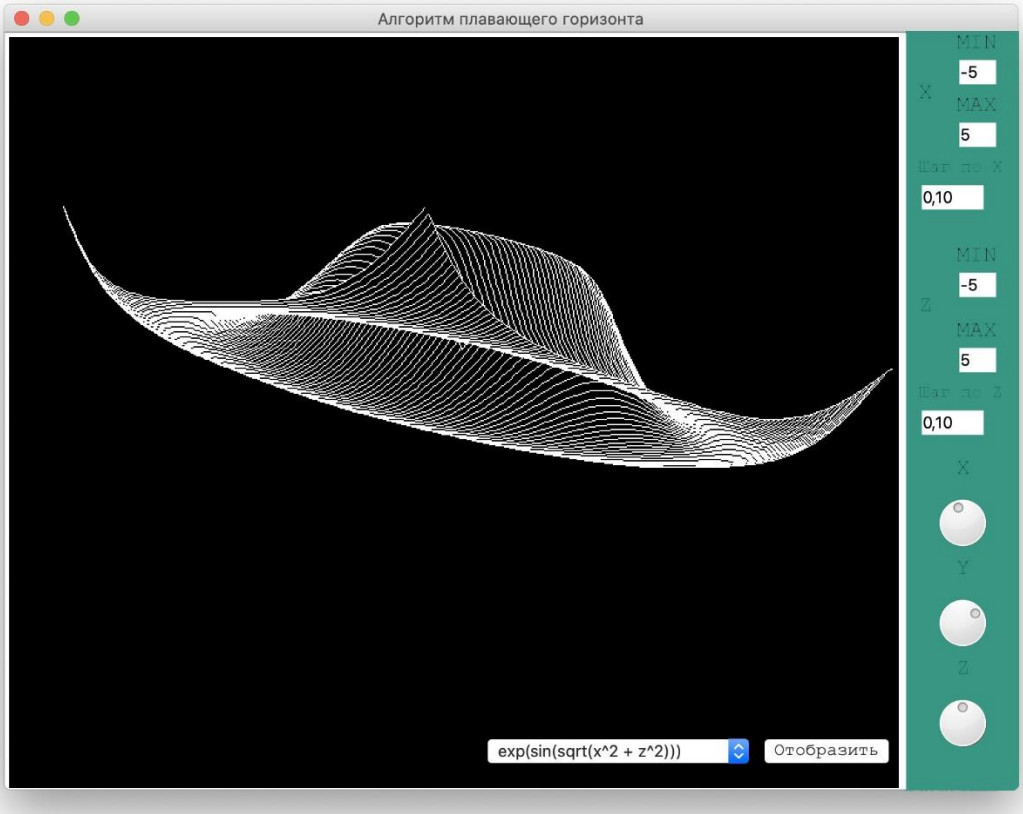
Зададим exp(sin(sqrt(x^2 + z^2)))



Вращение по OX



Вращение по OY



Вращение по OZ