МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное агентство по образованию
«Пермский национальный исследовательский политехнический университет»
Кафедра микропроцессорных средств автоматизации

Отчёт по лабораторной работе №3
По дисциплине компьютерная графика
На тему: «Алгоритмы удаления невидимых линий»

Работу выполнили
студенты гр. ИСУП-18-2м
_____ А. С. Морозов
_____ В. О. Раскошинский

Проверил доцент кафедры МСА
_____ Л. А. Мыльников

Пермь, 2020 г.

Описание алгоритма

Наиболее часто алгоритм плавающего горизонта применяется для удаления невидимых линий при отображении функций, описывающих поверхность в виде $f(x, y, z) = 0$. Подобные функции возникают во многих приложениях в математике, технике, естественных науках и других дисциплинах.

Главная идея данного метода заключается в сведении трехмерной задачи к двумерной путем пересечения исходной поверхности последовательностью параллельных секущих плоскостей, имеющих постоянные значения координат x, y или z.

Параллельные плоскости определяются постоянными значениями z. Функция $f(x, y, z) = 0$ сводится к последовательности кривых, лежащих в каждой из этих параллельных плоскостей, например к последовательности $y = f(x, z)$ или $x = g(y, z)$, где z постоянно на каждой из заданных параллельных плоскостей.

Алгоритм следующий: если на текущей плоскости при некотором заданном значении x соответствующее значение y на кривой больше максимума или меньше минимума по y для всех предыдущих кривых при этом значении x, то текущая кривая видима. В противном случае она невидима.

Реализация алгоритма

Файл `subroutine.py` содержит функцию `horizon`, которая реализует алгоритм плавающего горизонта, представлена в листинге 1.

```
1  def horizon(x1, y1, x2, y2, top, bottom, image):
2      x = x1
3      y = y1
4      dx = x2 - x1
5      dy = y2 - y1
6      sx = sign(dx)
7      sy = sign(dy)
8      dx = abs(dx)
9      dy = abs(dy)
10
11     if dx == 0 and dy == 0 and 0 <= x < image.width():
12         if y >= top[x]:
13             top[x] = y
14             image.setPixel(x, image.height() - y, Qt.white)
15
16         if y <= bottom[x]:
17             bottom[x] = y
18             image.setPixel(x, image.height() - y, Qt.white)
19
20         return top, bottom
21
22     change = 0
23     if dy > dx:
24         dx, dy = dy, dx
25         change = 1
26
```

```
27      y_max_curr = top[x]
28      y_min_curr = bottom[x]
29      e = 2 * dy - dx
30
31      i = 1
32      while i <= dx:
33          if 0 <= x < image.width():
34
35              if y >= top[x]:
36                  if y >= y_max_curr:
37                      y_max_curr = y
38                  image.setPixel(x, image.height() - y, Qt.white)
39
40              if y <= bottom[x]:
41                  if y <= y_min_curr:
42                      y_min_curr = y
43                  image.setPixel(x, image.height() - y, Qt.white)
44
45          if e >= 0:
46              if change:
47                  top[x] = y_max_curr
48                  bottom[x] = y_min_curr
49
50                  x += sx
51
52                  y_max_curr = top[x]
53                  y_min_curr = bottom[x]
54
55              else:
56                  y += sy
57
58              e -= 2 * dx
59          if e < 0:
60              if not change:
61                  top[x] = y_max_curr
62                  bottom[x] = y_min_curr
63
64                  x += sx
65
66                  y_max_curr = top[x]
67                  y_min_curr = bottom[x]
68
69              else:
70                  y += sy
71
72              e += 2 * dy
73
74          i += 1
75
76      return top, bottom
```

Листинг 1: Функция `horizon` – реализация алгоритма плавающего горизонта

Данная функция импортируется в файл `horizon.py` и используется совместно с функциями поворота координат `rotateX`, `rotateY`, `rotateZ` и `tranform` в функции `float_horizon`. Исходный код функции `float_horizon` представлен в листинге 2.

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

    # initialization of horizon arrays

    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)

        # process the left edge (look at the previous with the current)
        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)

            # Adding to the horizon and drawing lines
            # start to draw not from the previous one
            # but already from the transformed
            top, bottom = horizon(x_prev, y_prev, x_curr, y_curr, top, bottom, image)
            x_prev = x_curr
            y_prev = y_curr

            x += x_step

        # process the right edge (look at the current one with the following)
        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
```

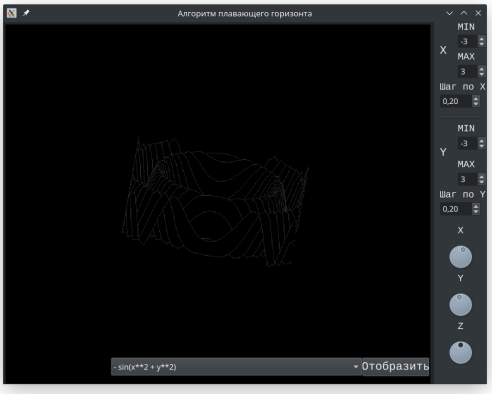
Листинг 2: Функция `float_horizon` – использование алгоритма

В то же время функция `float_horizon` импортируется в файл `lab.py` и используется в функции `draw`. Исходный код функции `draw` представлен в листинге 3. Функция `draw` используется в классе `Window` для визуализации результатов при изменении входных значений, полученных из пользовательского интерфейса, который описан в файле `window.ui`.

```
def draw(win):
    win.scene.clear()
    win.image.fill(black)
    tx = win.dial_x.value()
    ty = win.dial_y.value()
    tz = win.dial_z.value()

    if win.funcs.currentText() == "- sin(x**2 + y**2)":
        f = f1

    if win.funcs.currentText() == "(sin(sqrt(x**2 + y**2)))/(sqrt(x**2 + y**2))":
        f = f2

    if win.funcs.currentText() == "exp((- x**2 - y**2) * x)":
        f = f3

    if win.funcs.currentText() == "sqrt(x**2+y**2)+3*(cos(sqrt(x**2+y**2)))+5":
        f = f4

    win.image = float_horizon(win.scene.width(),
                              win.scene.height(),
                              win.x_min.value(),
                              win.x_max.value(),
                              win.dx.value(),
                              win.z_min.value(),
                              win.z_max.value(),
                              win.dz.value(),
                              tx, ty, tz, f,
                              win.image)

    pix = QPixmap()
    pix.convertFromImage(win.image)
    win.scene.addPixmap(pix)
```

Листинг 3: Функция `draw` – визуализация результатов

Графический интерфейс пользователя представлен на рисунках 1, 2 и 3. Весь код и результаты его работы представлены в открытом репозитории.



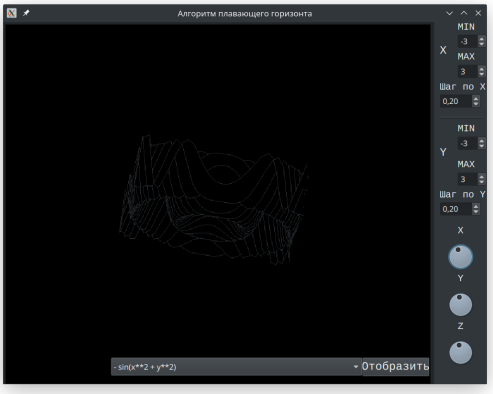


Рис. 1: $z = -sin\left(x^2 + y^2\right)$

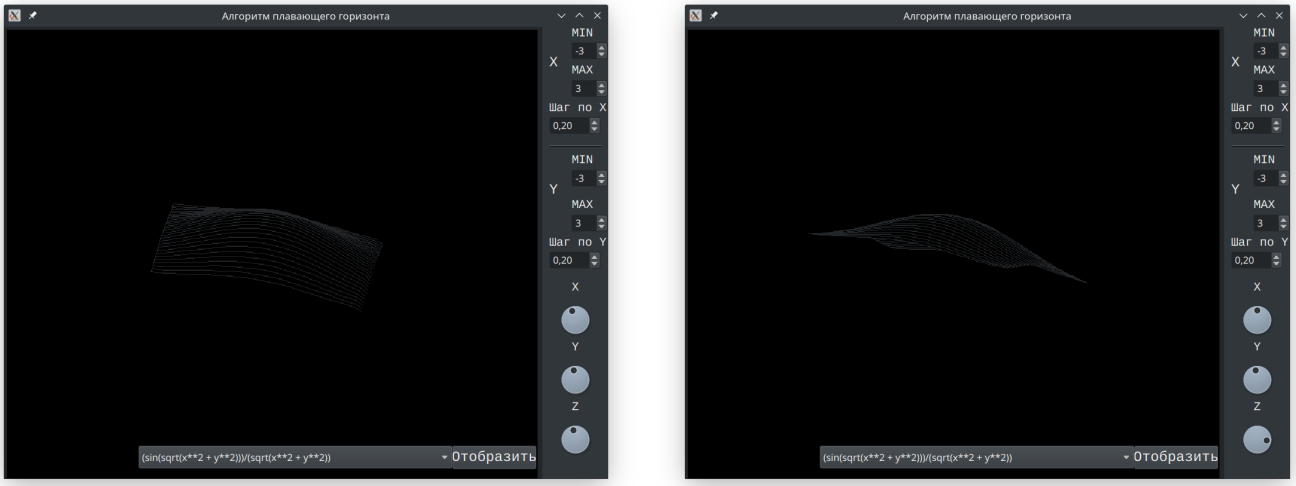


Рис. 2: $z = \frac{sin\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}}$

Рис. 3: $z = \sqrt{x^2 + y^2} + 3 \cdot cos\left(\sqrt{x^2 + y^2}\right) + 5$