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 № 6

Тема:Реализация и исследование алгоритма построчного затравочного заполнения сплошных областей

Студент: Блохин Д.М.

Группа: ИУ7-42Б

Оценка (баллы): _____

Преподаватель: Куров А.В.

Москва.
2020 г.

Цель работы: Реализация и исследование алгоритма построчного затравочного заполнения.

Задание:

Необходимо обеспечить ввод произвольной многоугольной области, содержащей произвольное количество отверстий. Ввод (вершин многоугольника) производить с помощью мыши, при этом для удобства пользователя должны отображаться ребра, соединяющие вводимые вершины. Предусмотреть ввод горизонтальных и вертикальных ребер. Должен быть предусмотрен ввод затравочной точки.

-Пользователь должен иметь возможность задания цвета заполнения.

-Работа программы должна предусматривать два режима – с задержкой и без задержки.

-Режим с задержкой должен позволить проследить выполняемую последовательность действий.

-Обеспечить замер времени выполнения алгоритма (без задержки, с выводом на экран только окончательного результата).

-Продемонстрировать возможность заполнения с помощью затравочного алгоритма произвольной области, ограниченной замкнутой кривой линией.

Ход работы:

Алгоритм построчного затравочного заполнения:

1. Ввод исходных данных (информация о границах заполняемой области, затравочный пиксель, цвет границы, цвет заполнения)
2. Занесение затравочного пикселя в стек.
3. Затравочный пиксел на интервале извлекается из стека, содержащего затравочные пикселы.
4. Интервал с затравочным пикселом заполняется влево и вправо от затравки вдоль сканирующей строки до тех пор, пока не будет найдена граница.
5. В переменных $X_{лев}$ и $X_{прав}$ запоминаются крайний левый и крайний правый пикселы интервала.
6. В диапазоне $X_{лев} <= x <= X_{прав}$ проверяются строки, расположенные непосредственно над и под текущей строкой. Определяется, есть ли на них еще не заполненные пикселы. Если такие пикселы есть (т. е. не все пикселы граничные, или уже заполненные), то в указанном диапазоне

крайний правый пиксел в каждом интервале отмечается как затравочный и помещается в стек.

Реализация:

```
def fill_default(self):
    stack = []
    stack.append(self.seed)

    while len(stack) > 0:
        x, y = stack.pop()

        x_temp = x
        put_pix(self, x, y, self.fill_color)

        x += 1
        while get_pix(self, x, y) != self.bd_color:
            put_pix(self, x, y, self.fill_color)
            x += 1
        x_right = x - 1

        x = x_temp

        x -= 1
        while get_pix(self, x, y) != self.bd_color:
            put_pix(self, x, y, self.fill_color)
            x -= 1
        x_left = x + 1

        for i in range(1, -2, -2):
            x = x_left
            y += i
            while x <= x_right:
                flag = False

                while get_pix(self, x, y) != self.bd_color and get_pix(self, x, y) != self.fill_color and x < x_right:
                    if not flag:
                        flag = True
                    x += 1

                if flag:
                    if get_pix(self, x, y) != self.bd_color and get_pix(self, x, y) != self.fill_color and x == x_right:
                        stack.append((x, y))
                    else:
                        stack.append((x - 1, y))

                x_enter = x
                while (get_pix(self, x, y) == self.bd_color or get_pix(self, x, y) == self.fill_color) and x < x_right:
                    x += 1

                if x == x_enter:
                    x += 1
            y -= i
```

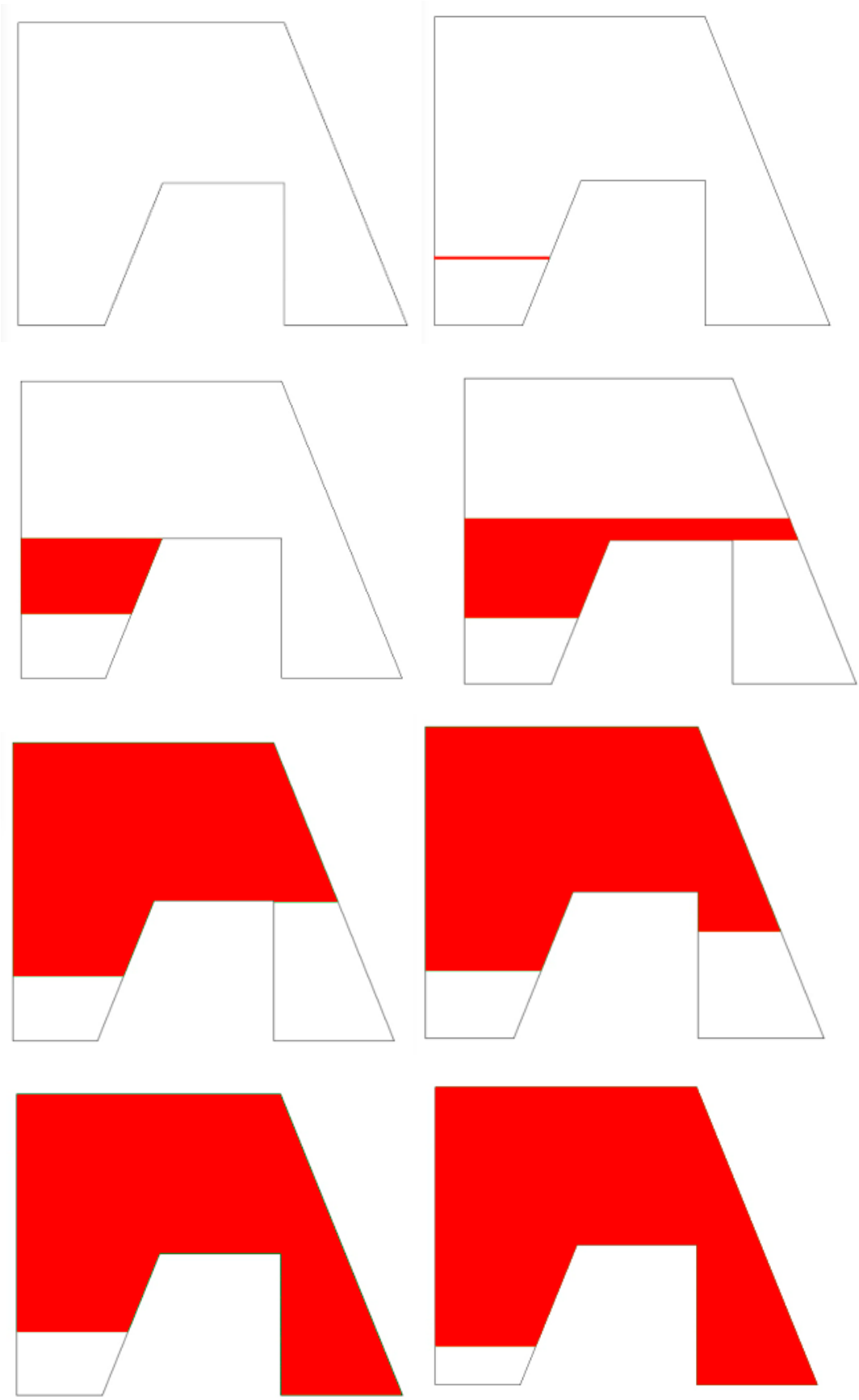
Используемые функции:

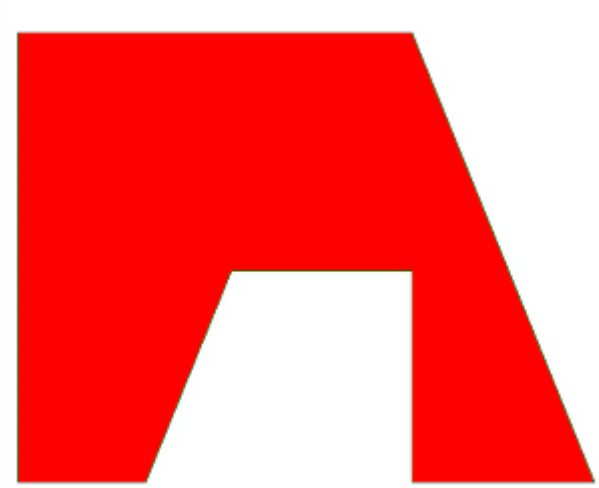
```python
# Закрашивает пиксель цветом
def put_pix(self, x, y, color):
    self.image.setPixel(x, y, color.rgb())


# Возвращает цвет пикселя
def get_pix(self, x, y):
    return self.image.pixelColor(x, y)
```

```python
# Рисует границы вокруг полотна указанным цветом
def put_borders(self, color):
    for i in range(self.can_w):
        put_pix(self, i, 0, color)
        put_pix(self, i, self.can_h - 1, color)

    for i in range(self.can_h):
        put_pix(self, 0, i, color)
        put_pix(self, self.can_w - 1, i, color)
```

Интерфейс и примеры работы

Заполнение:





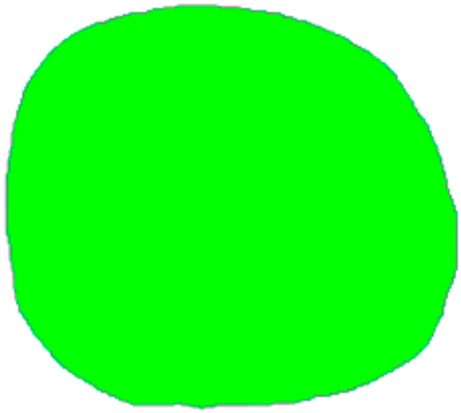
Время выполнения без задержки для данного многоугольника:

Время выполнения:

0.584 сек

Демонстрация работы с отверстиями:

Демонстрация возможности заполнения с помощью затравочного алгоритма произвольной области, ограниченной замкнутой кривой линией:



Подведем итоги:

Временная эффективность алгоритма:

1. Цвет пиксела анализируется 2 раза на строчках, расположенных на границах сверху и снизу, и у первого затравочного пиксела, в остальных случаях – 3 раза.
2. Цвет каждого пиксела изменяется **только один раз**
3. Обрабатываются только те пиксели, которые находятся внутри области закраски и пиксели, расположенные на границе.