**МГТУ им. Н.Э. Баумана**

**КОМПЬЮТЕРНАЯ ГРАФИКА**

**Лабораторный практикум №6**

**по теме: « Реализация и исследование алгоритма построчного затравочного заполнения сплошных областей»**

***Студент: Нгуен Фыок Санг***

***Группa: ИУ7И-46Б***

***Работу проверил:***

2020

**Цель работы: реализация и исследование алгоритма построчного затравочного заполнения.**

1. **Алгоритма построчного затравочного заполнения:**
   * ***Идея алгоритма:***

При инициализации алгоритма в стек помещается единственный затравочный пиксел, работа завершается при опустошении стека.

Пока стек не пуст:

* + 1. ***Затравочный пиксел на интервале извлекается из стека, содержащего затравочные пикселы и ему присваивается требуемое значение:***

пиксел (x,y) = требуемое зн-е.

* + 1. ***Интервал с затравочным пикселом заполняется влево и вправо от затравочной точки вдоль сканирующей строки до тех пор, пока не будет найдена граница:***
  1. Сохраняем координаты абсциссы затравочного пиксела:

xt=x.

* 1. Заполняем интервал вправо от затравки:

x=x+1

пока (пиксел (x,y) < > граничное зн-е)

делать:

пиксел (x,y)=требуемое зн-е;

x=x+1;

* 1. В переменной xr запоминаем крайний правый пиксел:

xr=x-1.

* 1. Восстанавливаем координату абсцисс затравки:

x =xt.

* 1. Заполняем интервал слева от затравки:

x=x-1,

пока (пиксел (x,y) < > граничное зн-е) делать:

пиксел (x,y)=требуемое зн-е; x=x-1;

* 1. В переменной xl запоминаем крайний левый пиксел:

xl= x+1.

* 1. Восстанавливаем координату абсциссы затравки:

x=xt.

* + 1. **В диапазоне xl <= x <= xr 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.:**
  1. Осуществляем проверку строки, расположенной выше текущей:

x=xl.

y=y+1

Пока (x<=xr) ищем затравочный пиксел на строке , выше текущей:

f=0 ( промежуточная переменная);

пока (пиксел (x,y)< > гран. зн-е и пиксел(x,y) < > треб. зн-е и x<xr)

делать:

если f=0 , тогда f=1;

x=x+1;

помещаем в стек крайний справа пиксел :

если f=1, тогда

если (пиксел(x,y)< >гран. зн-е и пиксел(x,y)< >треб. зн-е и x=xr), тогда пиксел (x,y) помещается в стек);

в противном случае в стек заносится пиксел(x-1,y);

затем промежуточной переменной присваивается нулевое значение: f=0.

Далее проверка не является ли строка выше текущей границей многоуника, или уже полностью заполненной продолжается в том случае, если интервал был прерван:

xn=x;

пока (пиксел (x,y)=гран. зн-е или пиксел(x,y)=треб. зн-е и x<xr)

делать : x=x+1;

убедимся, что координата абсциссы пиксела увеличилась:

если x=xn, тогда х=х+1.

* 1. Аналогично пункту 3.1 осуществляем проверку строки ниже текущей, изменения лишь касаются координаты y, ей присваивается значение : y=y-1.
     1. **И так до тех пор, пока стек не становится пустым, область заполненной , а работа алгоритма завершенной**
  + ***Отзывы:***
* Размер стека минимизируется за счет хранения только одного затравочного пиксела для любого непрерывного интервала на сканирующей строке.
* Алгоритм применим к гранично-определенным областям
  + ***Код программы:***

def fill(win):

pixmap = QPixmap()

painter = QPainter()

painter.begin(win.image)

boder = boder\_color.rgb()

fill = fill\_color.rgb()

**# инициализируем стек**

stack = []

p = get\_start\_point(win)

stack.append(p)

**# пока стек не пуст**

while stack:

**# извлечение пикселя (х,у) из стека**

pixel = stack.pop()

x = pixel.x()

y = pixel.y()

**# присваиваем ему новое значение**

win.image.setPixel(x, y, fill)

**# сохраняем x-координату затравочного пиксела**

xt = x

**# заполняем интервал слева от затравки**

x = x - 1

while win.image.pixel(x, y) != boder:

win.image.setPixel(x, y, fill)

x = x - 1

**# сохраняем крайний слева пиксел**

xl = x + 1

x = xt

**# заполняем интервал справа от затравки**

x = x + 1

while win.image.pixel(x, y) != boder:

win.image.setPixel(x, y, fill)

x = x + 1

**# сохраняем крайний справа пиксел**

xr = x - 1

**# ищем затравку на строке выше**

y = y + 1

x = xl

f = 0

while x <= xr:

f = 0

while win.image.pixel(x, y) != boder and win.image.pixel(x, y) != fill and x <= xr:

if f == 0:

f = 1

x = x + 1

**# помещаем в стек крайний справа пиксел**

if f == 1:

if x == xr and win.image.pixel(x, y) != fill and win.image.pixel(x, y) != boder:

stack.append(QPointF(x, y))

else:

stack.append(QPointF(x - 1, y))

f = 0

**# продолжим проверку, если интервал был прерван**

xt = x

while (win.image.pixel(x, y) == boder or win.image.pixel(x, y) == fill) and x < xr:

x = x + 1

**# удостоверимся, что координата пиксела увеличена**

if x == xt:

x = x + 1

**# ищем затравку на строке ниже**

y = y - 2

x = xl

while x <= xr:

f = 0

while win.image.pixel(x, y) != boder and win.image.pixel(x, y) != fill and x <= xr:

if f == 0:

f = 1

x = x + 1

if f == 1:

if x == xr and win.image.pixel(x, y) != fill and win.image.pixel(x, y) != boder:

stack.append(QPointF(x, y))

else:

stack.append(QPointF(x - 1, y))

f = 0

xt = x

while (win.image.pixel(x, y) == boder or win.image.pixel(x, y) == fill) and x < xr:

x = x + 1

if x == xt:

x = x + 1

**# с задержкой**

if win.delay.isChecked():

delay()

pixmap.convertFromImage(win.image)

win.scene.addPixmap(pixmap)

**# без задержки**

if not win.delay.isChecked():

pixmap.convertFromImage(win.image)

win.scene.addPixmap(pixmap)



