РАСТРОВЫЕ АЛГОРИТМЫ

Под дискретной плоскостью будем понимать множество всех точек с целочисленными координатами на обычной двухмерной плоскости. Дискретную плоскость называют также целочисленной решеткой, растровой плоскостью или растром.

Простейшие свойства множеств на целочисленной решетке

Для удобства дальнейшего изложения будем считать, что на плоскости имеется квадратная сетка с шагом 1, причем узлы нашей целочисленной решетки являются центрами соответствующих квадратных ячеек сетки. Другими словами, узлы растра окружены "единичными" квадратными окрестностями "радиуса" 1/2. Инициализации точки растра с координатами (*i, j*) соответствует закраска каким-либо цветом ее квадратной окрестности (рис.1).

Точки на плоскости называются 4-соседями (или "непосредственными" соседями), если у них отличаются только *x*-координаты или только *y*-координаты, причем только на 1.

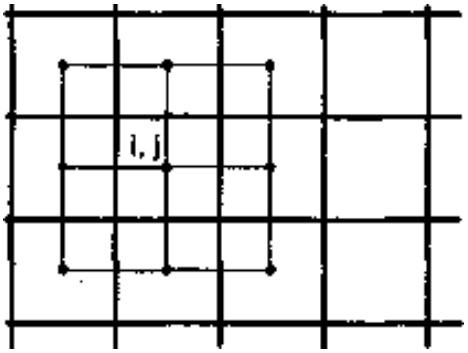

Рис.1

Точки на плоскости называются 8-соседями (или "косвенными" соседями), если у них отличаются *x*-координаты или *y*-координаты, но не более чем на 1.

Заметим, что в соответствии с этими соглашениями всякий непосредственный сосед яв) же время и косвенным соседом.

Всякая точка на плоскости имеет четырех непосредственных соседей и восемь косвенных соседей, откуда и второе название для таких точек. Если точки являются непосредственными соседями, то их квадратные окрестности имеют общую сторону; квадратные окрестности косвенных соседей имеют общую сторону или общую вершину (рис.2).

Введем понятие пути на целочисленной решетке.

4-путем (или сильносвязным путем) на плоскости называется множество точек, $A_1, A_2, ..., A_n$ для которых точки A_i, A_{i+1} являются косвенными соседями для $i = 1, 2, ..., n-1$.

8-путем (или слабосвязным путем) на плоскости называется множество точек $A_1, A_2, ..., A_n$, для которых точки A_i, A_{i+1} являются слабыми соседями для $i = 1, 2, ..., n-1$.

Путь называется замкнутым, если $A_1 = A_n$.

Множество на целочисленной решетке называется сильносвязным (четырех-связным), если любые две точки его можно соединить сильносвязным путем.

Множество на целочисленной решетке называется слабосвязным (восьми-связным), если любые две точки его можно соединить слабосвязным путем.

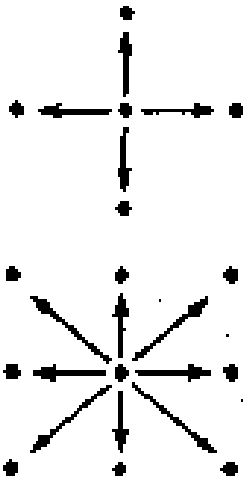

Рис.2

Попробуем определить кривую на дискретной плоскости, т. е. "одномерное" множество на целочисленной решетке.

Простой кривой на плоскости называется множество, у которого все точки, за исключением двух, имеют ровно двух соседей (слабых или сильных, в зависимости от вида кривой). Эти две исключительные точки имеют ровно по одному соседу.

Простой замкнутой кривой на плоскости называется множество, у которого все точки имеют ровно двух соседей (слабых или сильных, в зависимости от вида кривой).

Простая замкнутая слабосвязная кривая разбивает плоскость на два сильносвязных множества. Простая замкнутая сильносвязная кривая разбивает плоскость на два слабосвязных множества.

Растровое представление геометрических объектов

Понятие растрового представления геометрического объекта

В дальнейшем целочисленную решетку будем называть растром.

Предположим, что на обычной двухмерной плоскости имеется некоторый геометрический объект и нужно получить дискретное представление объекта на целочисленной решетке, или, как говорят, растровое представление объекта.

Это означает, что заданной геометрической фигуре (линии или области) следует поставить в соответствие множество на целочисленной плоскости, которое в некотором смысле является приближением исходной фигуры.

Такое представление неоднозначно, так как, что такое приближение, можно понимать по-разному и существуют разные способы приближения непрерывного объекта.

Для "заполненной" геометрической фигуры растровым приближением можно считать множество точек дискретной плоскости, принадлежащих этой фигуре. Однако такое определение теряет смысл, если мы хотим получить растровое представление отрезков прямых или дуг кривых.

Например, растровым приближением круга на плоскости может служить множество точек, попавших внутрь этого круга, а растровым приближением прямой – множество точек, являющихся центрами ячеек решетки, с которыми прямая пересекается.

Можно предложить и другие способы растрового представления этих объектов.

Важным является вопрос реализации того или иного представления, т. е. об алгоритмах растровой развертки и об их эффективности.

Уточним задачу. Будем считать, что в нашем распоряжении имеется элементарная операция – инициализация точки с целочисленными координатами.

В каждом конкретном случае нужно указать алгоритм выбора последовательности точек, которые нужно инициировать на растре, чтобы получить соответствующее растровое представление объекта в целом.

Возникает естественный вопрос: зачем нужно знать, как работают алгоритмы генерации отрезка прямой, окружности или эллипса, алгоритмы

закраски многоугольника или, в более общем случае, области плоскости, если все они реализованы в виде стандартных процедур в любом пакете типа Turbo Pascal или Turbo C. Дело в том, что монитор не единственное растровое устройство, а при работе с такими устройствами, как принтер, плоттер, мышь, необходимо уметь программно реализовывать растровую генерацию соответствующих геометрических объектов.

Необходимость "залезть внутрь" алгоритма генерации возникает и тогда, когда атрибуты инициируемого пиксела (цвет, наличие или отсутствие) зависят от каких-либо условий. Например, от положения пиксела на прямой или внутри закрашиваемого многоугольника. Умение построить соответствующий алгоритм часто позволяет изменить структуру существующего алгоритма с целью ускорения его работы. Такие ситуации типичны при разработке алгоритмов удаления невидимых линий и поверхностей (например, для растровой реализации одной из версий метода плавающего горизонта или метода буфера глубины).

Растровая развертка отрезка. Алгоритм Брезенхема

Процесс последовательной инициализации множества пикселов экрана, изображающего отрезок прямой линии, называется растровой разверткой отрезка, а само это множество – растровым представлением отрезка. Из сказанного ясно, что важно знать не только, как устроено это множество, но и владеть способом его генерации, т. е. располагать алгоритмом, позволяющим последовательно строить точки этого множества.

Более точно, если нам известны целочисленные координаты концов отрезка, мы должны знать, какие точки надлежит последовательно инициировать на растре, чтобы получить полное растровое представление отрезка. Эта задача решается неоднозначно. Ее решение зависит от того, какого типа растровый образ мы хотим получить. Даже если ограничить класс растровых представлений отрезка простыми растровыми кривыми в смысле определений предыдущего пункта, то таких представлений может

быть два – восьмисвязное и четырехсвязное. Возможны и другие растровые модели отрезка в зависимости от того, какими свойствами мы хотели бы наделить полученный образ. Не углубляясь в обсуждение этого вопроса, предложим сначала простое, "наивное", решение задачи.

Чтобы выбрать промежуточные пикселы (точки растра), которые являются в некотором смысле наименее удаленными от идеального отрезка, можно, например, инициализировать последовательно все точки растра, окрестности которых пересекаются с этим отрезком (рис.3).

Задача. Убедитесь, что в результате получится четырехсвязная растровая развертка отрезка. Предложите алгоритм последовательной генерации точек этой кривой.

Пусть концы M_1 и M_2 отрезка имеют координаты (x_1, y_1) и (x_2, y_2). Тогда отрезок определяется уравнением $y = y_1 + k(x - x_1)$,

где $k = \dfrac{y_2 - y_1}{x_2 - x_1}, x_1 \le x \le x_2$.

Для простоты предположим, что угловой коэффициент k неотрицателен и не превосходит 1: $k \le 1$.

Ниже представлена процедура генерации растровой развертки отрезка.

```
Dx:=1; Dy:=abs((y2-y1)/(x2-x1));

x:=x1; y:=y1;
For i:=0 to L-1 do
begin
     PutPixel(x,round(y));
     x:=x+Dx; y:=y+Dy;

end.
```

Схема работы этого алгоритма проста: получаем очередную точку на отрезке и инициируем пиксел, ближайший к этой точке.

Условие $k \le 1$ нужно, чтобы в процессе построения растровой развертки не было пропущено ни одной точки; шаг по оси x единичный, а по оси y – меньше 1. Если угловой коэффициент прямой больше 1, нужно поменять

ролями переменные *x* и *y*.

Заметим, что целочисленная абсцисса точки изменяется на каждом шаге на 1, в то время как целочисленная ордината претерпевает изменение лишь в случае, когда в результате накопления приращений *Dy* вещественная ордината точки окажется в окрестности радиуса 0,5 соседнего уровня по оси ординат. Учтем эти наблюдения и несколько изменим алгоритм по форме, оставляя основную схему без изменения.

```
x:=x1; y:=y1; n:=x2-x1;

m:=y2-y1; d:=m/n; e:=0;
For i:=1 to n do
   begin
      {шаг по оси абсцисс и вычисление отклонения}
      x:=x+1; e:=e+d;
      {если отклонение по оси ординат от текущего
      значения y превосходит 1/2, то нужно увеличить
      y на 1 и скорректировать отклонение e от нового
      значения y}
      if e>0.5 then
         begin
             y:=y+1; e:=e-1;
          end;
      PutPixel(x,y,7)

   end;
```

Нетрудно заметить, что в результате работы алгоритма получится восьмисвязное представление отрезка, так как переход к следующей точке развертки осуществляется на одну из восьми соседних клеток.

Попытаемся теперь сформулировать соображения, которые позволяют описать как восьмисвязное, так и четырехсвязное растровые представления отрезка на плоскости.

Проанализировав расположение отрезка относительно квадратных окрестностей узлов решетки, можно предложить следующие правила генерации четырехсвязной и восьмисвязной развертки отрезка:

1) четырехсвязная развертка отрезка включает те и только те точки

решетки, квадратные окрестности которых пересекаются с отрезком;

2) восьмисвязная развертка отрезка включает те, и только те точки решетки, боковые стороны квадратных окрестностей которых пересекаются с отрезком (рис.4).

Пользуясь правилом 1), выпишем алгоритм генерации четырехсвязной развертки. Имеем

```
x:=x1; y:=y1; n:=x2-x1;

m:=y2-y1; d:=m/n; e:=d/2;
For i:=1 to n+m do
begin
     {шаг по оси абсцисс и вычисление отклонения}
     x:=x+1; e:=e+d;
     {если отклонение по оси ординат от текущего
     значения y превосходит 1/2, то нужно увеличить
     y на 1 и скорректировать отклонение e от нового
     значения y}
     if e>0.5 then
       begin
          y:=y+1; e:=e-1;
       end
         else
            begin
            x:=x+1; e:=e+d;
            end;
     PutPixel(x,y,'white');
end;
```

В этой схеме переменная *e* выражает разницу между ординатой текущей точки на прямой и ординатой точки пересечения прямой с правой границей квадратной окрестности текущей точки растровой развертки.

Если $e \le 0{,}5$, то отрезок пересекается с боковой стороной квадратной окрестности точки, лежащей справа от текущей точки, и нужно сместиться вправо на 1.

Если $e > 0{,}5$, то отрезок пересекается с нижней, границей квадратной окрестности, лежащей выше точки, и нужно сместиться на 1 вверх. Эти

соображения показывают также, что в результате работы последнего алгоритма получится четырехсвязная развертка (рис.5).

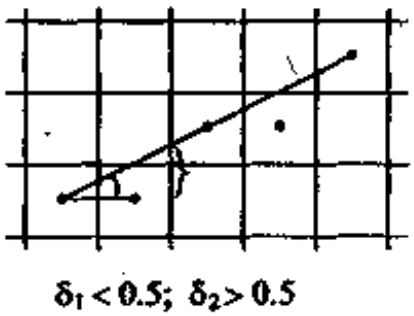

Рис.5

Известно, что операции с вещественными числами осуществляются гораздо медленнее соответствующих операций с целыми числами. Поэтому для повышения эффективности работы алгоритма желательно освободиться от операций, использующих вещественную арифметику. С этой целью в алгоритме для получения восьмисвязной развертки отрезка, умножая переменные *e* и *d* на целое число 2*n*, мы избавляемся от дробных чисел. Изменив "масштаб" переменных и внеся корректировки в неравенства, используемые в операциях сравнения, получим "целочисленную" версию алгоритма:

```
x:=x1; y:=y1;
n:=x2-x1; m:=y2-y1;
d:=2*m; dy:=2*n; e:=0; {1--> dy=2*n}
for i:=1 to n do
begin
  {шаг по оси абсцисс и вычисление отклонения}
  x:=x+1; e:=e+d;
  {если отклонение по оси ординат от текущего
  значения y превосходит 1/2, то нужно увеличить
  y на 1 и скорректировать отклонение e от нового
  значения y}
  if e>n then
        begin
             y:=y+1; e:=e-dy;
        end;
     PutPixel(x,y,'white');

end;
```

Полученный алгоритм носит название целочисленного алгоритма Брезенхема для восьмисвязной развертки отрезка в первом квадранте.

Используя аналогичные преобразования, запишем процедуру генерации четырехсвязной развертки. Имеем:

```
x:=x1; y:=y1;

n:=x2-x1; m:=y2-y1;
d:=2*m; dy:=2*n; e:=m;
for i:=1 to n+m do
begin
     {шаг по оси абсцисс и вычисление отклонения}
     x:=x+1; e:=e+d;
     {если отклонение по оси ординат от текущего
     значения y превосходит 1/2, то нужно увеличить
     y на 1 и скорректировать отклонение e от нового
     значения y}
     if e>n then
               begin
                    y:=y+1; e:=e-dy;
               end
          else
               begin
                    x:=x+1; e:=e+dx;
               end;
      PutPixel(x,y);

end;
```

Полученный алгоритм называется целочисленным алгоритмом Брезенхема для четырехсвязиой развертки отрезка в первом квадранте.

Общий алгоритм Брезенхема для восьмисвязной развертки отрезка

Для получения общего алгоритма растровой развертки нужно избавиться от ограничений, которые мы до сих пор накладывали на расположение отрезка на плоскости, а именно от требования $0 \le k \le 1$ на угловой коэффициент *k*.

Учитывая ориентацию отрезка относительно положительных направлений осей координат и меняя ролями переменные x и y в случае $|k| > 0$, получим окончательные общие версии алгоритма.

Общий алгоритм Брезенхема для восьмисвязной развертки отрезка:

```
PROCEDURE line_8(x1,y1,x2,y2:integer);

VAR
x,y,s1,s2,dx,dy,e,z,i:integer;
change:boolean;
BEGIN
x:=x1; y:=y1; dx:=abs(x2-x1); dy:=abs(y2-y1);
s1:=sign(x2-x1); s2:=sign(y2-y1);
      if dy>dx then
      begin
           z:=dx; dx:=dy; dy:=z;
           change:=true;
      end
      else
          change:=false;
      e:=2*dy-dx;
      for i:=1 to dx do
          begin
                PutPixel(x,y,color);
                while e>=0 do
                    begin
                    if change then x:=x+s1
                       else y:=y+s2; e:=e-2*dx;
                    end;
                if change then y:=y+s2
                       else x:=x+s1;
                e:=e+2*dy;
          end;
          putPixel(x,y,color)

END;
```

Функцию $sign(x)$ надо предварительно определить.

Общий алгоритм Брезенхема для четырехсвязной развертки отрезка:

```
PROCEDURE line_4(x1,y1,x2,y2:integer);

VAR
x,y,sx,sy,dx,dy,e,z,i:integer;
change:boolean;
BEGIN
x:=x1; y:=y1; dx:=abs(x2-x1); dy:=abs(y2-y1);
sx:=sign(x2-x1); sy:=sign(y2-y1);e:=2*dy-dx;
      if dy<dx then change:=false
```

```
                    else
                      begin
                        z:=dx; dx:=dy; dy:=z;
                        change:=true;
                      end;
                  for i:=1 to dx+dy do
                      begin
                        if e<dx then
                          begin
                           if change then y:=y+sy
                            else x:=x+sx;
                                 e:=e+2*dy;
                          end
                          else
                             if change then x:=x+sx
                               else y:=y+sy;
                                    e:=e-2*dx;
                      end;
                    putPixel(x,y,color);

END;
```