Заполнение сплошных, областей

Вопрос о заполнении внутренности сплошной области занимает важное место среди задач растровой графики. Большинство задач о заполнении двухмерной фигуры может быть отнесено к одному из двух типов: заполнение внутренности многоугольника, заданного своими вершинами или ребрами, и заполнение внутренности области, ограниченной замкнутым контуром, представленным своей растровой разверткой.

Тест принадлежности точки многоугольнику

Будем понимать под многоугольником фигуру, ограниченную на плоскости простой (несамопересекающейся) замкнутой ломаной. Сама ломаная задается набором своих вершин $A_i(x_i, y_i)$, $i = 1, 2, ..., n$, причем соседние точки в этом списке являются смежными вершинами ломаной. Задача состоит в том, чтобы получить растровую развертку многоугольника, т. е. инициировать его внутренние точки.

Начнем с обсуждения задачи о локализации точки относительно многоугольника. Решение этой задачи даст возможность эффективно определять, является ли точка внутренней или внешней по отношению к нему.

Знаменитая теорема Жордана утверждает, в частности, что простая (т. е. не имеющая самопересечений) замкнутая плоская ломаная разбивает плоскость на две связные компоненты – ограниченную, которая является внутренностью многоугольника, и неограниченную, которая является внешней по отношению к многоугольнику.

Алгоритм должен уметь различать внутренние и внешние точки плоскости. Обозначим ребра многоугольника через E_i:

$E_i : [A_i, A_{i+1}], i = 1, 2, ..., n$.

Пусть $P(x, y)$ – некоторая точка плоскости, не лежащая на ломаной, и нужно определить, принадлежит она этому многоугольнику или нет.

Проведем через точку *P* горизонтальную полупрямую с правым концом в точке *P*. Так как ломаная ограничена, то всегда легко найти на этой полупрямой достаточно удаленную точку *Q*, которая заведомо не принадлежит многоугольнику. Если отрезок *QP* не имеет пересечений с границей многоугольника, то точки *Q* и *P* лежат в одной компоненте связности и, следовательно, точка P – внешняя.

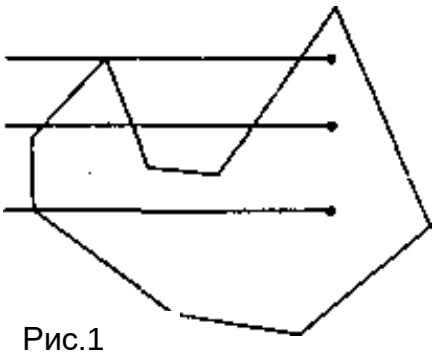

Рис.1

Рассмотрим случай, когда отрезок *QP* пересекает ломаную (рис.1). Будем двигаться от точки Q в направлении к точке P. Миновав первое пересечение отрезка и границы, мы попадем внутрь многоугольника. Миновав следующее пересечение отрезка и границы, мы окажемся снаружи многоугольника, и так далее. Легко видеть, что если мы встретим на своем пути четное число пересечений, то точка *P* будет внешней точкой многоугольника. Если же число пересечений окажется нечетным, то точка *P* будет внутренней точкой многоугольника. Важно только удостовериться, что пересечения отрезка с границей были существенными, т. е. отрезок действительно пересек ломаную, а не просто касался ее в одной из вершин.

Для выявления существенных пересечений можно воспользоваться следующим правилом. Пересечения отрезка с горизонтальными ребрами игнорируются. При подсчете числа пересечений негоризонтальных ребер ломаной с отрезком *PQ* пересечение игнорируется, если точкой пересечения является верхняя вершина ребра, и засчитывается в любом другом случае. С учетом этого соглашения касание отрезка *PQ* с ломаной в точках максимума игнорируется, а в точках минимума считается дважды. Тем самым несущественные пересечения не изменяют четности общего числа пересечений. Если же существенное пересечение имеет место в вершине ломаной (верхнее ребро пересекает отрезок *PQ* в нижней вершине, а нижнее ребро – в верхней вершине, и по нашему соглашению принимается во внимание лишь первое пересечение), то число подсчитанных пересечений

увеличивается на единицу.

Высказанные выше соображения приводят к следующему алгоритму:

Заполнение многоугольников

Для заполнения внутренности многоугольника можно было бы воспользоваться описанным выше тестом принадлежности, перебирая последовательно точки растра и инициируя те из них, которые лежат внутри многоугольника. Такой алгоритм прост и понятен, однако в смысле временных затрат очень неэффективен. Его можно несколько улучшить, поместив многоугольник внутрь минимального объемлющего прямоугольника со сторонами, параллельными осям координат, и анализировать лишь те точки, которые попали внутрь прямоугольника.

Ломаная, ограничивающая многоугольник, разбивает всякую горизонтальную прямую на чередующиеся интервалы, лежащие внутри или снаружи многоугольника. Для определения концов этих интервалов можно поступить следующим образом.

Зафиксируем горизонтальную прямую L, на которой предполагается определить местоположение интервалов, находящихся внутри многоугольника, ищем точки пересечения с этой прямой, если они есть, каждого ребра многоугольника. Для проверки наличия пересечения предварительно убедимся в том, что концы ребра расположены по разные стороны от прямой.

При поиске точек пересечения будем пользоваться правилами, примененными в тесте принадлежности. Тогда кратные пересечения с контуром многоугольника в его вершинах будут правильно учтены. Далее упорядочим полученные точки слева направо и сгруппируем их попарно. Эти пары и будут являться концами интервалов, лежащих внутри многоугольника и

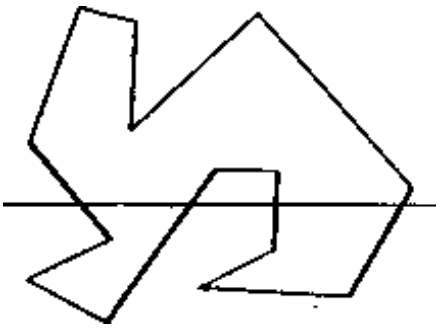

Рис.2

подлежащих закраске (рис.2).

Изложенная схема заполнения многоугольника носит название заполнения в порядке сканирования строк, а сам алгоритм относится к типу алгоритмов построчного сканирования.

Для ускорения работы алгоритма можно предварительно упорядочить ребра в порядке возрастания наибольшей из ординат концов. При перемещении сканирующей прямой сверху вниз проверке на пересечение подвергаются лишь те ребра, у которых значение максимальной ординаты больше ординаты сканирующей прямой. При этом из списка исключаются ребра, значение минимальной ординаты которых больше ординаты сканирующей прямой. Таким образом, поддерживается информация о множестве "активных" ребер, пересекающихся со сканирующей прямой (рис.3).

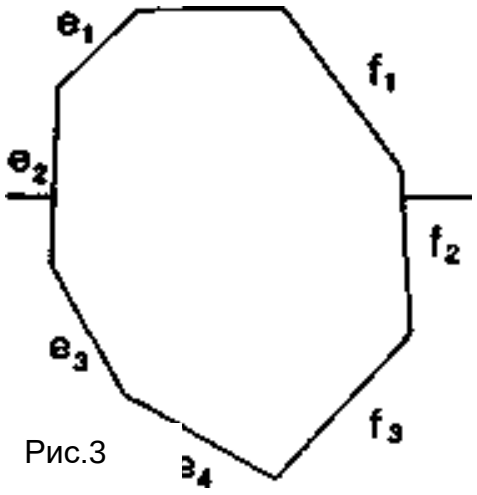

Рис.3

Приведенный алгоритм пригоден для заполнения произвольных многоугольников. Если же многоугольник является выпуклым, то алгоритм можно упростить и повысить его эффективность. Заметим, что границу выпуклого многоугольника можно разбить на две ломаные – "левую" и "правую" и, возможно, два ребра – "верхнее" и "нижнее" так что каждая из боковых ломаных имеет ровно одно пересечение с каждой сканирующей прямой (рис. 9). Используя алгоритм Брезенхема и одновременно генерируя растровое представление для ребер левой и правой ломаных границы, получим левый и правый пикселы границы многоугольника на каждой сканирующей горизонтальной прямой. Последовательно заполняя интервалы между этими пикселами для каждого значения ординаты от верхней строки развертки к нижней, получим растровую развертку выпуклого многоугольника.

Алгоритмы заполнения области с затравкой

В алгоритмах заполнения области с затравкой используется иной подход. В них предполагается, что граница области задана на растровой плоскости и указана одна из внутренних точек области, которая называется затравочной. Требуется заполнить определенным цветом связную компоненту области, содержащую затравочный пиксел. Под связностью понимается 4– или 8–связность на дискретной плоскости (в зависимости от постановки задачи). Можно представить, что в затравочной точке находится источник, заливающий всю область определенным цветом. Поэтому этот процесс иногда называют заливкой области.

Опишем алгоритм заполнения области с затравкой, использующий стек. Под стеком понимается линейный массив (список) элементов, причем вставки и удаления элементов можно производить лишь с одного конца, т. е. если элемент был вставлен в список последним, он должен быть обработан и удален из списка первым.

Будем предполагать, что нужно получить заполнения 4-компоненты, ограниченной заданным контуром.

Начнем с того, что поместим затравочной пиксел в стек. Далее, пока стек не пуст, будем извлекать из него очередной пиксел и, закрасив его, анализировать пикселы, соседние с ним. Если среди них есть пикселы, не принадлежащие границе и не окрашенные нужным цветом, то поместим их в стек. После этого вернемся к операции извлечения пиксела из стека. По окончании работы алгоритма все внутренние пикселы области будут закрашены.

Формально алгоритм можно описать так:

Приведенный выше алгоритм заполнения области весьма неэффективен, так как предполагает неоднократную обработку одних и тех же пикселов и неконтролируемый рост величины стека. Ниже приводится более эффективный алгоритм заполнения области.

Построчный алгоритм заполнения с затравкой.

Применим идею построчного сканирования для решения задачи заполнения. Заметим, что на каждой строке множество пикселов, подлежащих закраске, состоит из интервалов, принадлежащих внутренности области. Эти интервалы отделены друг от друга интервалами из пикселов, принадлежащих границе или внешности области. Кроме того, если набор пикселов образует связный интервал, принадлежащий внутренней части области, то пикселы на и под этим интервалом либо являются граничными, либо принадлежат внутренней части области (рис.4). Последние могут служить в качестве затравочных пикселов для строк, лежащих выше и ниже рассматриваемой строки. Принимая во внимание сказанное, можно предложить следующую схему заполнения области.

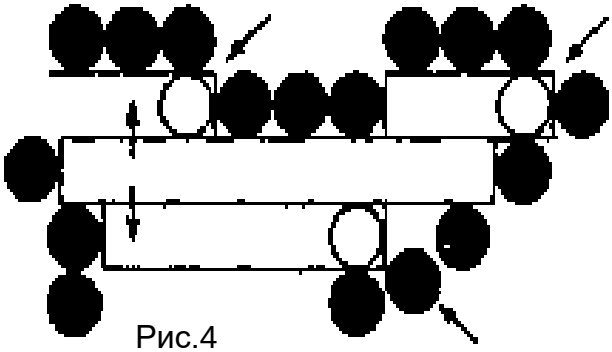
Рис.4

Инициализируем стек, помещая в него затравочный пиксел. Пока стек не пуст:

- извлекаем пиксел (x,y) из стека;
- заполняем максимально возможный интервал, в котором находится пиксел, вправо и влево вплоть до достижения граничных пикселов; запоминаем крайнюю левую Lx и крайнюю правую Rx абсциссы заполненного интервала; в соседних строках над и под интервалом (Lx, Rx) находим незаполненные к настоящему моменту внутренние пикселы области, которые, как мы уже заметили, объединены в интервалы, а в каждом из этих интервалов находим крайние правые пикселы. Каждый из этих пикселов вносим в стек в качестве затравочного.

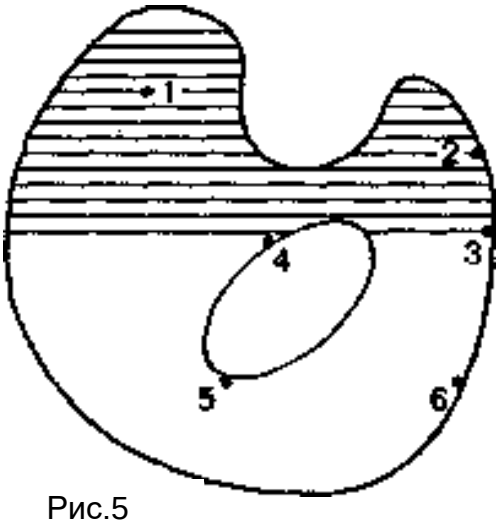

Рис.5

Алгоритм правильно заполняет любую область, в том числе и такую, в которой присутствуют отверстия (рис.5).