УДК 681.3

АЛГОРИТМ ОПРЕДЕЛЕНИЯ ПРИНАДЛЕЖНОСТИ ТОЧКИ МНОГОУГОЛЬНИКУ ОБЩЕГО ВИДА ИЛИ МНОГОГРАННИКУ С ТРЕУГОЛЬНЫМИ ГРАНЯМИ

Н.А. Тюкачев

Кафедра «Программирование и информационные технологии»,
ГОУ ВПО «Воронежский государственный университет»;
nik@cs.vsu.ru

Представлена членом редколлегии профессором С.В. Мищенко

Ключевые слова и фразы: алгоритмы на графах; геометрические алгоритмы.

Аннотация: Рассматривается алгоритм определения принадлежности точки многоугольнику общего вида или многограннику с триангулированной поверхностью, основанный на локальном анализе семейств инцидентных вершин. Алгоритм, в отличие от ранее предлагаемых, не требует глобальной упорядоченности ребер и может быть использован в трехмерном пространстве для оверлейных операций со слоями в геоинформационных системах.

Проверка принадлежности точки многоугольнику или многограннику необходима во многих алгоритмах, например, в булевых операциях над многоугольниками или многогранниками. Для простых многоугольников, определенных замкнутым контуром без самопересечений и без «дыр»; известно несколько алгоритмов [2, 3, 6]. Все они предназначены или для выпуклых многоугольников или для многоугольников, у которых контур представляет собой упорядоченную последовательность ребер. Ни один из этих алгоритмов, кроме триангуляционного, не может быть обобщен на трехмерные многогранники, так как даже на планарном графе, соответствующем тетраэдру (рис. 1), не существует эйлерова цикла, то есть пути, проходящего без повторений через все ребра.

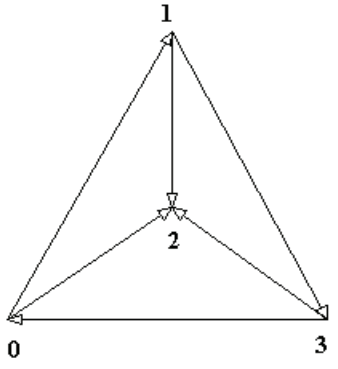


Рис. 1. Планарный граф тетраэдра

Существенно сложнее анализ принадлежности точки для многогранников, поверхность которых представляет собой набор, например, треугольных граней (рис. 2). Такие многогранники используются, например, для описания слоев в геоинформационных системах.

Прежде, чем перейдем к обзору 2D-алгоритмов, напомним несколько определений из геометрии и теории графов. Часто эти определения перекликаются, но, в основном, определения теории графов – топологические. Только при определении планарного графа необходимы координаты вершин для определения пересечения ребер.

Рассмотрим евклидову плоскость D_2 с декартовой системой координат.

Ребром называется направленный отрезок с началом в точке a и концом в точке b, не совпадающей с a, который обозначается как $E(a,b)$. При этом будем говорить, что для точки b ребро E является *входящим*, а для точки a – *выходящим*.

Контуром C называется множество ребер $[E_0, E_1, \ldots, E_{n-1}]$ такое, что $n > 2$ и $\forall i \in \{0, \ldots, n-1\} : E_{i-1} = E(v_{i-1},v_i)$, $E_i = E(v_i, v_{i+1})$. Точка v_i, для которой ребра E_{i-1} и E_i являются соответственно входящим и выходящим, будем называть i-й вершиной контура C, ребра E_{i-1} и E_i – соседними или смежными. Порядок обхода ребер контура $C = [E_0, E_1, \ldots, E_{n-1}]$ с увеличением индекса $i \to i+1$ будем называть *прямым* направлением обхода, а противоположный ему – *обратным*.

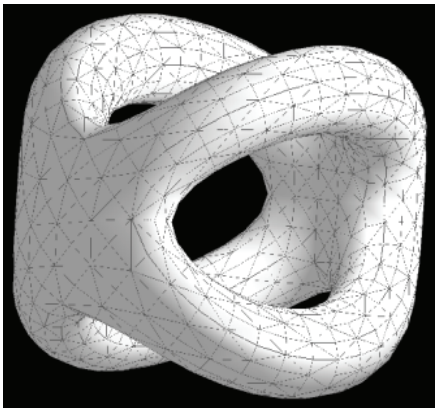


Рис. 2. Многогранник, поверхность которого представляет собой набор треугольных граней

Областью A называется ограниченное полигональное открытое связное множество на евклидовой плоскости, определенное с точностью до множества меры нуль. Под *полигональностью* понимается то, что граница области состоит из замкнутых ломаных.

Рассмотрим границу dA некоторой области A. Если ориентация границы dA согласована с ориентацией A, то граница области соответствует набору контуров на плоскости. Для взаимно-однозначного соответствия к определению области добавляется два ограничения.

Контур C называется *ограничивающим* контуром области A, если выполняются следующие условия:

1) $C \subset dA$;

2) обход C в прямом направлении является положительным обходом, то есть обходом, при котором область А остается слева;

3) $\forall i \in \{0,\ldots, n-1\}: \exists \varepsilon^+ > 0: \forall x \in \varepsilon^+ (v_i): x \in A$;

4) $\forall i \in \{0,\ldots, n-1\}: \exists \varepsilon^- > 0: \forall x \in \varepsilon^- (v_i): x \notin A$.

Общим полигоном или многоугольником называется множество всех ограничивающих контуров некоторой области. *Внешним* контуром области A назовем ограничивающий ее контур, положительный обход которого производится против часовой стрелки. *Внутренним* контуром области A назовем ограничивающий ее контур, положительный обход которого производится по часовой стрелке.

Многоугольник называется *выпуклым*, если определяемое им множество точек является выпуклым.

Многоугольник называется *простым* если: он имеет, по крайней мере, две различных вершины; ни одна пара непоследовательных ребер не разделяет вершину; каждая пара последовательных ребер разделяет только одну вершину.

Многоугольник называется *регулярным*, если он не имеет пересекающихся или «висящих» ребер и совпадающих вершин.

Вершинно-полным называется многоугольник, который не имеет пересекающихся ребер, однако у него могут быть совпадающие вершины и коллинеарные ребра.

Ориентированным называется многоугольник, любые два ребра которого или не пересекаются, или коллинеарны, или пересекаются в точке, являющейся конечной точкой как минимум одного из ребер.

Классы полигонов образуют следующую иерархию: *выпуклые* ⊂ *простые* ⊂ *регулярные* ⊂ *вершинно-полные* ⊂ *ориентированные* ⊂ *общие*.

Классическое определение принадлежности точки многоугольнику тесно связано с ориентированностью контура, то есть с существованием эйлерова цикла и теорией графов.

Пусть V – непустое конечное множество. Через V_2 обозначим множество всех двуэлементных подмножеств из V. *Графом* G называется пара множеств (V, E), где E – произвольное подмножество из V_2. Элементы множеств V и E называют соответственно *вершинами* и *ребрами* графа G.

Ориентированный граф (орграф) $G = (V, E)$ есть пара множеств, где V – множество вершин (узлов); E – множество дуг (ориентированных ребер). Дуга – это упорядоченная пара вершин (v, w), где вершину v называют началом, а w – концом дуги. Можно сказать, что дуга $v \to w$ ведет от вершины v к вершине w, при этом вершина w смежная с вершиной v.

Два ребра, *инцидентные* одной вершине, называются *смежными*; две вершины, *инцидентные* одному ребру, также называются *смежными*.

Планарный граф – граф, который может быть изображен на плоскости без пересечения ребер.

Степень вершины (*валентность* вершины) v – это количество ребер, инцидентных этой вершине v. Обозначается как $\deg(v)$.

Маршрут (или *путь*) в графе G – это чередующаяся последовательность вершин и ребер $v_0, e_1, v_1, e_2, v_2, ..., e_k, v_k$, в которой любые два соседних элемента инцидентны. Если $v_0 = v_k$, то маршрут замкнут, иначе – открыт. Другими словами, маршрутом называется чередующаяся последовательность вершин и ребер $v_0, e_1, v_1, e_2, v_2, ..., v_{t-1}, e_t, v_t$, в которой $e_i = v_{i-1}v_i$ $(1 \le i \le t)$. Такой маршрут кратко называют (v_0, v_t)-маршрутом и говорят, что он соединяет v_0 с v_t; а вершины v_0, v_t – концевые вершины указанного маршрута.

Цепь в графе – маршрут, все ребра которого различны.

Если все вершины (а, следовательно, и ребра) различны, то такая цепь называется *простой*. В цепи $v_0, e_1, v_1, e_2, v_2, ..., e_k, v_k$ вершины v_0 и v_k называются *концами* цепи. Цепь с концами u и v соединяет вершины u и v. Для орграфов цепь называется *путем*.

Цикл в орграфе – это простой путь длиной не менее 1, который начинается и заканчивается в одной и той же вершине. Простой цикл орграфов называется *контуром*.

Эйлеров цикл – это такой цикл, который проходит ровно один раз по каждому ребру неориентированного или ориентированного графа.

Алгоритм определения эйлерова цикла прост: выходим из произвольной точки и помечаем ребра графа. На каждом шаге идем по ребру, удаление которого нарушает связанность графа, если другого пути нет.

Теорема Эйлера. Связный неориентированный граф содержит эйлеров цикл тогда и только тогда, когда число вершин нечетной степени равно 0 или 2.

Согласно определениям через вершину может проходить несколько фрагментов контура многоугольника, и каждый фрагмент добавляет вершине две степени. Следовательно, такие многоугольники удовлетворяют условиям теоремы Эйлера.

На рис. 3 представлен ориентированный граф из 27 узлов, для которого существует эйлеров цикл.

Кратко напомним известные алгоритмы определения принадлежноости точки $Q(x, y)$ многоугольнику.

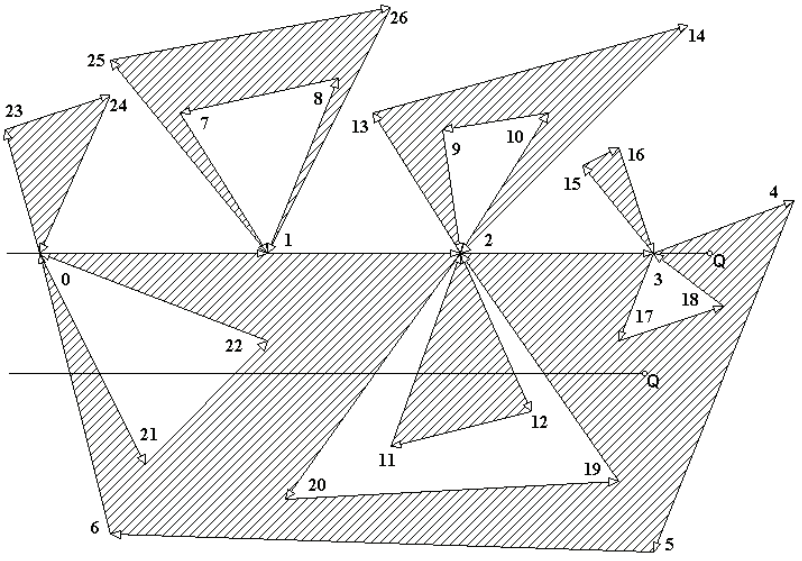


Рис. 3. Пример графа, удовлетворяющего условию Эйлера

1. Алгоритм для выпуклых многоугольников, в частности, для треугольников

При последовательном обходе вершин в одном направлении проверяется, что для всех $i \in [1, \ldots, n]$ точка $Q(x, y)$ и v_{i+2} лежат по одну сторону от ребра $E(v_i, v_{i+1})$. Формализацию алгоритма можно реализовать, введя нормальное ориентированное уравнение прямой [4], проходящей через вершину v_i с координатами (x_i, y_i)

$$f_i = Nx_i(x - x_i) + Ny_i(y - y_i),$$

где Nx_i и Ny_i – проекции единичной внешней нормали. Функция f_i принимает положительные значения для внешних относительно ребра точек: $(x, y) \in \Omega_i^+$, отрицательные – для внутренних точек, нулевые значения – для точек на границе. Внешняя область для выпуклого многоугольника задается пересечением всех внешних областей Ω_i^+

$$\Omega^+ = \bigcap_{i=1}^{n} \Omega_i^+ .$$

2. Тригонометрический алгоритм

Для *простого односвязного* многоугольника алгоритм может быть следующим. Вершины v_i многоугольника последовательно нумеруются от 1 до n. Для каждой пары вершин вычисляется разность углов лучей, проведенных из точки Q. Разности суммируются. Если сумма равна нулю, то делается вывод о том, что заданная точка лежит вне заданного многоугольника.

Этот алгоритм требует большого объема вычислений с использованием обратных тригонометрических функций и, поэтому, непригоден для компьютерных интерактивных графических приложений, в которых многоугольник имеет большое число вершин.

3. Триангуляционный алгоритм

Проводится триангуляция многоугольника [5]. Проверяется принадлежность точки Q хотя бы одному из этих треугольников. Если точка не принадлежит ни одному треугольнику, значит, она не принадлежит и многоугольнику. Алгоритм может быть применен и для трехмерных многогранников, но задача разбиения произвольного многогранника на тетраэдры не тривиальна.

4. Определение по площади для выпуклых многоугольников, в частности, для треугольников

Точка Q соединяется отрезками с вершинами треугольника. Если площадь исходного многоугольника (треугольника) равна сумме площадей образовавшихся треугольников, то считается, что точка принадлежит треугольнику.

5. Лучевой алгоритм

Для многоугольника вычисляется число пересечений луча, например, горизонтального, проведенного из точки Q достаточно далеко за пределы многоугольника. Если это число четное, то точка не принадлежит многоугольнику.

Основные проблемы возникают при прохождении луча Q через вершины: для многоугольника общего вида (см. рис. 3) в вершине может сходиться несколько ребер и ребра могут совпадать с лучом. Алгоритм предлагает в этом случае засчитывать пересечение луча только с концом ориентированного ребра. Важным аспектом является глобальная по всему контуру упорядоченность ребер, обеспечивающая то, что в вершине не могут сходиться концами два последовательных ребра.

Как уже утверждалось ранее, для многогранников глобальная упорядоченность ребер и граней, то есть существование замкнутых циклов по графу ребер и граней, невозможна. Поэтому необходимо использовать иной алгоритм, который будем называть *инцидентным*, опробовать который будем на многоугольниках общего вида.

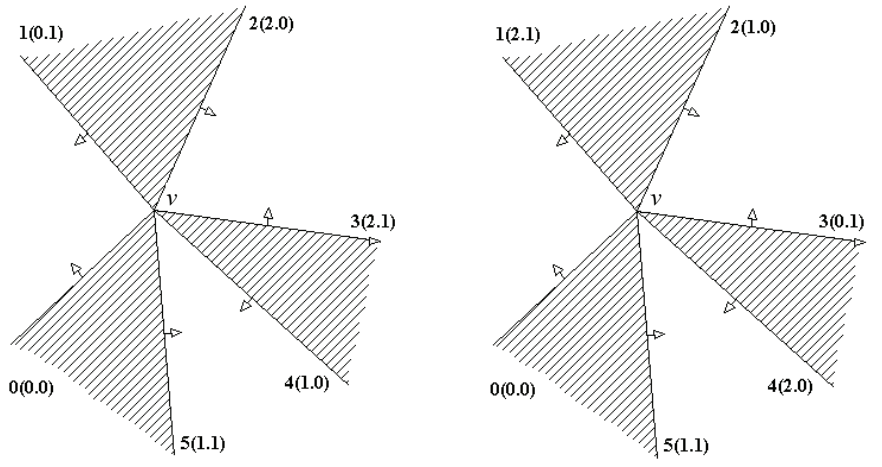
6. Инцидентный лучевой алгоритм для многоугольников

Для алгоритмов принадлежности точки особую сложность представляют вершины, степень которых больше 2. Для дальнейшего анализа введем новое понятие – *семейство инцидентных вершин*. В плоском случае через вершину v может проходить несколько фрагментов контура. Каждый фрагмент может содержать только два ребра E_0 и E_1 с нормалями N_0 и N_1, у которых внешние нормали не меняют ориентацию, то есть скалярное произведение векторных произведений должно быть больше нуля

$$(E_0 \times N_0,\ E_1 \times N_1) > 0\,. \qquad (1)$$

Две инцидентных вершины, одна из которых является началом ребра E_0, а вторая – концом ребра E_1, образуют семейство инцидентных вершин. У вершины v_0, изображенной на рис. 4, *а* и 4, *б*, существует 6 инцидентных вершин, которые могут входить в три семейства.

В случае 4, *а* это семейства [0(0,0), 1(0,1)], [4(1,0), 5(1,1)], [2(2,0), 3(2,1)]. В случае 4, *б* это семейства [0(0,0), 3(0,1)], [2(1,0), 5(1,1)], [4(2,0), 1(2,1)]. Первое число после номера вершины обозначает номер семейства, второе – номер вершины в семействе. Всего может быть шесть различных вариантов семейств. На рис. 4, *а* и 4, *б* представлено только два из них. Недопустимы варианты, в которых происходит смена направления нормали, например, вершины 1 и 5 не могут быть инцидентными в одном семействе.



а) *б*)

Рис. 4. Три семейства вершины *v* с шестью инцидентными вершинами

Лучевой алгоритм определения принадлежности в плоском случае сводится к следующим шагам:

– находим те вершины, которые попадают на луч;

– для каждой такой вершины строим семейство инцидентных вершин;

– для каждого семейства определяем, принадлежит ли вершина интервалу проекций инцидентных вершин на ось *Y*, если принадлежит, то число пересечений увеличиваем на 1;

– для всех ребер без вершин считаем число пересечений.

Таким образом, задача определения принадлежности свелась от двухмерной к одномерной: определения принадлежности точки интервалу.

7. Инцидентный лучевой алгоритм для многогранников

Для многогранников в трехмерном пространстве на вершину *v* может опираться любое количество граней, ребер и вершин. Для поверхности, изображенной на рис. 5, инцидентные грани образуют два семейства инцидентных вершин: [0(0,0), 1(0,1), 2(0,2), 3(0,3)] и [4(1,0), 5(1,1), 6(1,2)].

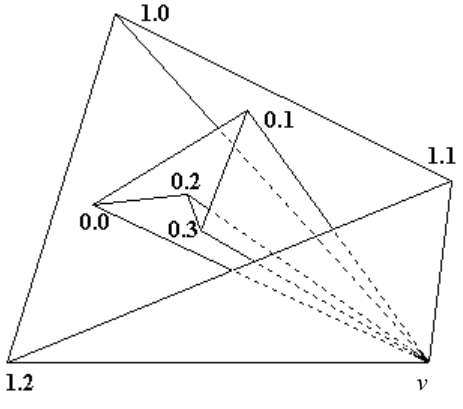


Рис. 5. Два семейства инцидентных граней вершины *v*

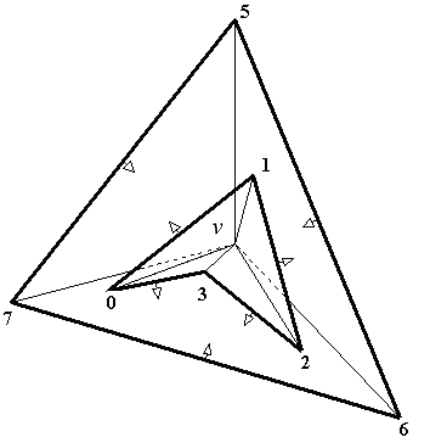


Рис. 6. Проекция двух семейств инцидентных граней вершины *v* на плоскость *YZ*

На рис. 6 изображена проекция двух семейств инцидентных граней вершины *v* на плоскость *YZ* перпендикулярную лучу *X*. В первое семейство входят вершины 0, 1, 2, 3, во второе – 5, 6, 7. Критерием отбора в семейство является выполнение условия: любые две инцидентные вершины должны принадлежать грани, у которой третья вершина *v*.

Определение принадлежности луча *X* конусу семейств инцидентных граней сводится к задаче определения принадлежности проекции вершины *v* на плоскость *YZ* проекции многоугольника инцидентных вершин семейства на ту же плоскость.

Лучевой алгоритм определения принадлежности в трехмерном случае сводится к следующим шагам:

– для всех граней без вершин и ребер считаем число пересечений;

– находим те вершины, которые попадают на луч;

– для каждой такой вершины строим семейства инцидентных вершин;

– для каждого семейства определяем, принадлежит ли проекция вершины *v* на плоскость *YZ* проекции многоугольника инцидентных вершин семейства на ту же плоскость, если принадлежит, то число пересечений увеличиваем на 1;

– находим те ребра, которые попадают на луч;

– для каждого такого ребра строим семейства инцидентных вершин;

– для каждого семейства определяем, принадлежит ли проекция вершины *v* на плоскость *YZ* проекции многоугольника инцидентных вершин семейства на ту же плоскость, если принадлежит, то число пересечений увеличиваем на 1.

Таким образом, трехмерный алгоритм определения принадлежности точки свелся к двухмерному алгоритму.

8. ER-модель

Для описания многоугольника или триангулированной поверхности будем использовать модель «сущность-связь» (Entity-Relationship model или ER-модель). Основной концепцией ER-модели является [1]:

– тип сущности (entity type), который представляет группу объектов реального мира, обладающих одинаковыми свойствами;

– тип связи (relationship type) является набором ассоциаций между одним (или несколькими) типами сущностей, участвующими в этой связи. Каждому типу связи присваивается имя, которое должно описывать его назначение.

Рассмотрим мгогоугольники общего вида, которые могут быть определены тремя сущностями: многоугольники (polygons), ребра (edges) и вершины (vertex). В общем случае эти сущности связаны между собой тремя отношениями «многие ко многим» *:* (многоугольники – ребра, многоугольники – вершины, ребра – вершины) и тремя отношениями «один ко многим» (многоугольник – инцидентные многоугольники, ребро – инцидентные ребра, вершина – инцидентные вершины). Но для алгоритма принадлежности достаточно оставить пять связей «один ко многим» (рис. 7).

Обращаем внимание на то, что для реализации алгоритма для каждой вершины потребуется знать семейства ее соседей, то есть оставить связь «вершина – инцидентные вершины».

Структура сущности Vertex помимо координат *x*, *y* содержит поле Family – двумерный массив семейств номеров инцидентных вершин и поле Visit – признак посещения вершины, используемый при реализации алгоритма принадлежности. Этот признак запрещает обрабатывать вершину дважды.

Структура ребер Edge содержит массив IdVertex номеров вершин, на который она опирается, и массив IdPolygon номеров многоугольников, которому ребро принадлежит. Кроме этого у ребра есть поле Norm – вектор внешней нормали.

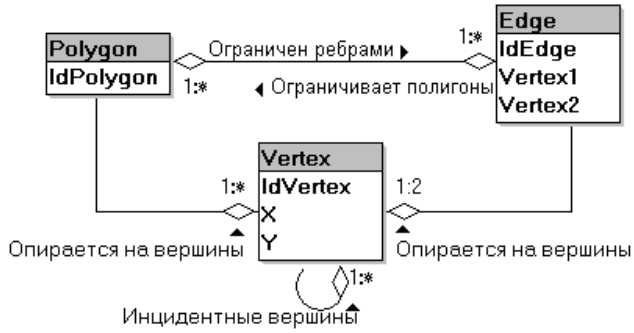


Рис. 7. ER-модель для многоугольников общего вида

Наконец, структура многоугольников Polygon содержит массив IdVertex номеров вершин, на который она опирается, и массив IdEdge номеров ребер, составляющих контур многоугольника.

Для многогранников в ER-модель добавляется еще одна сущность тела Body (рис. 8).

Структура сущности Vertex помимо координат *X*, *Y*, *Z*, массивов номеров граней IdFace и тел IdBody, содержит поле Family – двумерный массив семейств номеров инцидентных вершин и поле Visit – признак посещения вершины, используемый при реализации алгоритма принадлежности. Этот признак запрещает обрабатывать вершину дважды.

Структура ребер Edge содержит номера вершин Vertex1 и Vertex2, на которые ребро опирается, массив IdBody номеров тел, которому ребро принадлежит.

Структура треугольных граней Face содержит массивы IdVertex номеров вершин, на который она опирается, массив IdEdge номеров ребер, составляющих контур грани и массив IdBody номеров тел, которому ребро принадлежит. Кроме этого у грани есть поле Norm – вектор внешней нормали.

Наконец, структура Body содержит массив номеров граней IdFace и вершин IdVertex.

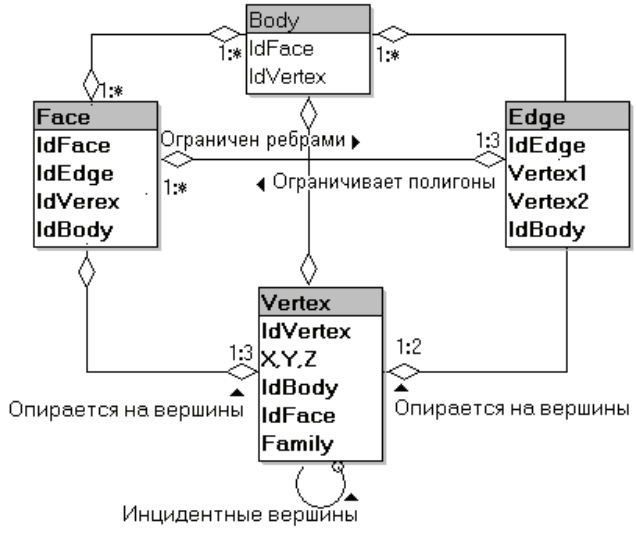


Рис. 8. ER-модель для многогранников

9. Детализация алгоритма для многоугольников

Введем бит принадлежности точки внутренней области Parity, принимающий значение 0 и 1 и меняющий значение при входе в область и при выходе из нее.

Алгоритм состоит из двух шагов:

1. Находим те вершины, которые попадают на луч.

1.1. Для каждой такой вершины строим семейство инцидентных вершин.

1.2. Для каждого семейства определяем, принадлежит ли вершина интервалу проекций инцидентных вершин на ось *Y*, если принадлежит, то меняем бит принадлежности.

2. Для всех ребер без вершин считаем число пересечений.

Второй шаг проблем не вызывает. Рассмотрим более подробно шаг 1.1 плоского алгоритма. На рис. 3 представлен ориентированный граф из 27 узлов, для которого существует эйлеров цикл: 0, 1, 8, 7, 1, 25, 26, 1, 2, 12, 11, 2, 10, 9, 2, 20, 19, 2, 13, 14, 2, 3, 17, 18, 3, 15, 16, 3, 4, 5, 6, 0, 23, 24, 0, 21, 22, 0. Таких циклов существует достаточно много. Так, например, для вершины с номером 1 существует три фрагмента путей по вершинам: (0, 1, 8; 7, 1, 25; 26, 1, 2), но возможен путь по вершинам (7, 1, 25; 26, 1, 8; 0, 1, 2). Для вершины с номером 2 существует пять фрагментов путей: (1, 2, 12; 11, 2, 10; 9, 2, 20; 19, 2, 13; 14, 2, 3).

Впрочем, построение эйлерова цикла не обязательно для построения семейств инцидентных вершин.

1. Можно, например, упорядочить инцидентные ребра вершины по углу и каждую последовательную пару соответствующих инцидентных вершин считать семейством. Такой подход требует тригонометрических вычислений с неизбежной потери точности и замедляющих работу алгоритма. Поэтому мы его рассматривать не будем.

2. Второй локальный способ сводится к последовательной сборке инцидентных вершин в пары, удовлетворяющие условию (1).

3. Еще один локальный способ формирования семейств будет предложен ниже при обсуждении алгоритмов для многогранников.

Тем не менее, вернемся к проблеме глобальной упорядоченности ребер, то есть к алгоритму нахождения эйлерова цикла. Для реализации алгоритма нахождения эйлерова цикла потребуется рабочий стек Stack и еще один стек для эйлерова пути Path. Напомним алгоритм определения эйлерова пути.

1. Очистить признаки посещения узлов.

2. Начать движение из любого узла, например, с номером $v = 0$, поместив этот номер в стек Stack.

3. Если стек Stack не пуст, то для текущего узла найти ребро, соединяющее с инцидентных узлом, по которому еще не ходили. Иначе происходит выход из алгоритма.

4. Если такое ребро найдено то, поместить в стек Stack инцидентный узел, пометить то, что ребро пройдено, и перейти на шаг 3. Иначе выполнить шаг 5.

5. Если ребро не найдено, то взять номер узла из стека Stack, то есть вернуться назад, поместить номер узла в стек пути Path и перейти на шаг 3.

Результат работы алгоритма накапливается в стеке Path:

Path = (0, 8, 7, 1, 25, 26, 1, 1, 12, 11, 2, 10, 9, 2, 20, 19, 2, 13, 14, 2, 2, 17, 18, 3, 15, 16, 3, 3, 4, 5, 6, 0, 23, 24, 0, 21, 22, 0).

Однако, для ветвящихся контуров типа представленного на рис. 3 стек Path не совсем точно отражает путь: во-первых, сильно ветвящиеся вершины, например 1 или 2 или 3, несколько раз подряд попадают в стек; во-вторых, рядом оказываются вершины не соединенные ребрами. Для корректировки пути введем еще два стека: PathTemp – вспомогательный стек – и PathDouble – для повторяющихся вершин. Алгоритм состоит из двух проходов. На первом проходе собираем повто-

ряющиеся вершины в стек PathDouble, на втором – вставляем из стека PathDouble вершины, между которыми не существует ребра. Во время первого прохода стек Path разрушается, поэтому приходится создавать вспомогательный стек PathTemp.

Первый проход состоит из следующих шагов.

1. Взять номер вершины из стека Path в переменную Old.

2. Если стек Path не пуст, взять номер вершины в переменную *v*. Иначе выход.

3. Если номера вершин *v* и Old совпадают, то положить *v* в стек PathDouble.

4. Иначе положить *v* в стек PathTemp.

5. Назначить Old ← *v* и перейти к шагу 2.

Второй проход работает со стеком PathTemp и состоит из следующих шагов.

1. Взять номер вершины из стека PathTemp в переменную Old.

2. Положить в стек Path.

3. Если стек PathTemp не пуст, взять номер вершины в переменную *v*. Иначе выход.

4. Найти ребро, у которого первая вершина Old, а вторая – *v*.

5. Если такое ребро найдено, то положить *v* в стек Path. Old ← *v*. Перейти к шагу 3.

6. Иначе взять из стека PathDouble номер вершины Old, положить его в стек Path и перейти к шагу 4.

В результате работы алгоритма стек Path содержит один из возможных эйлеровых путей, который, например, для графа, изображенного на рис. 3, содержит следующие вершины: Path = (0, 1, 8, 7, 1, 25, 26, 1, 2, 12, 11, 2, 10, 9, 2, 20, 19, 2, 13, 14, 2, 3, 17, 18, 3, 15, 16, 3, 4, 5, 6, 0, 23, 24, 0, 21, 22, 0).

Получение семейств инцидентных вершин из пути не составляет труда: необходимо для каждой вершины правого и левого соседа поместить в новое семейство.

На шаге 1.2 алгоритма для каждого семейства определяем, принадлежит ли проекция вершины P_y на ось Y открытому интервалу проекций инцидентных вершин (v_y, w_y) одного семейства на ось Y, если принадлежит, то число пересечений увеличиваем на 1. Возможны пять вариантов принятия решения (рис. 9–11):

1) $P_y \notin (v_y, w_y)$ – принимается решение: нет смены бита принадлежности Parity;

2) $P_y \in (v_y, w_y)$ – принимается решение: смена бита принадлежности Parity;

3) одна инцидентная вершина находится перед вершиной на луче: отложенное решение. В переменную sign запоминаем знак проекции на ось Y другой вершины;

4) одна смежная вершина находится за вершиной на луче: принимается решение с учетом знака sign отложенного решения (рис. 10, 11);

5) обе инцидентные вершины находится на луче – нет смены бита принадлежности.

Бит принадлежности меняется или в случае 2, или, может быть, в случае 4. Отметим, что для корректного анализа отложенных решений необходима упорядоченность вершин на луче справа налево.

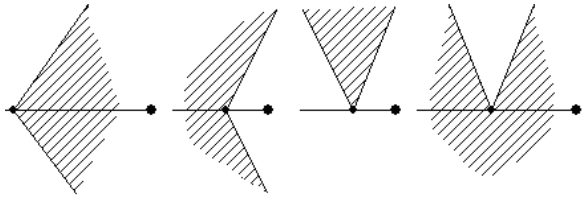


Рис. 9. Принимается решение: есть смена бита или нет

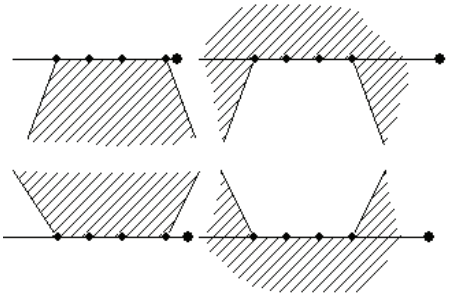


Рис. 10. Непринятые отложенные решения

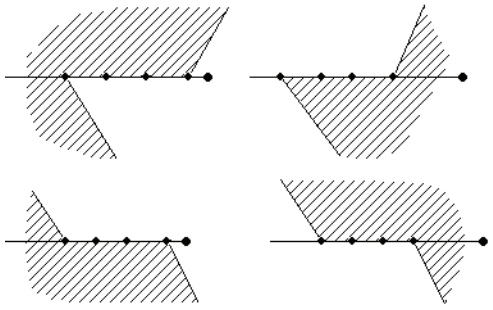


Рис. 11. Принятые отложенные решения

10. Детализация алгоритма для многогранников

Алгоритм состоит из трех шагов.

1. Находим вершины, попадающие на луч.

1.1. Для каждой такой вершины строим семейства инцидентных вершин.

1.2. Для каждого семейства определяем, принадлежит ли проекция вершины v на плоскость YZ проекции многоугольника инцидентных вершин семейства на ту же плоскость, если принадлежит, то меняется бит принадлежности. Задача сводится к плоской.

2. Находим ребра, попадающие на луч.

2.1. Для каждого такого ребра строим семейства инцидентных вершин.

2.2. Для каждого семейства определяем, принадлежит ли проекция вершины v на плоскость YZ проекции многоугольника инцидентных вершин семейства на ту же плоскость, если принадлежит, то меняется бит принадлежности.

3. Для всех открытых граней без вершин и ребер, пересекающихся с лучом, меняется бит принадлежности.

Обсудим более подробно шаг 1.1. Будем считать, что проекция на плоскость YZ, перпендикулярную лучу, каждого семейства может быть односвязным невыпуклым многоугольником без дырок. На рис. 12 изображено четыре семейства инцидентных вершин для вершины v_0, находящейся в начале системы координат YZ.

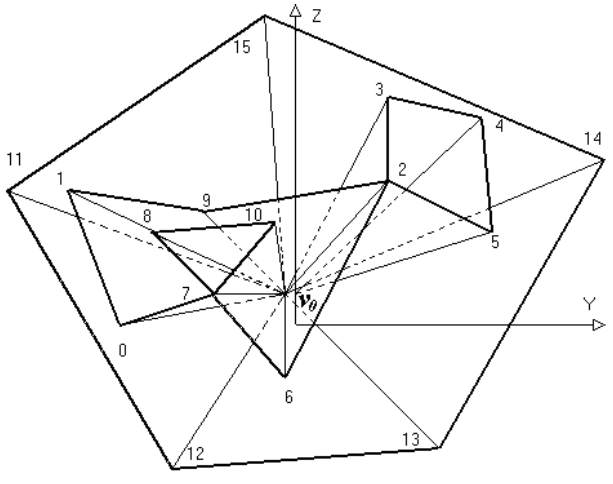


Рис. 12. Проекция на плоскость *YZ* инцидентных граней, ребер и вершин

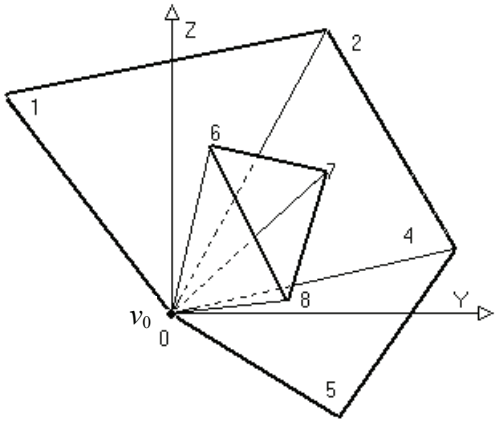


Рис. 13. Инцидентное ребро (v_0, 0) совпало с лучом

Проекция вершины и инцидентных граней содержит, в данном случае, четыре семейства инцидентных вершин: [0, 1, 9, 2, 6, 7], [2, 3, 4, 5], [7, 10, 8], [11, 12, 13, 14, 15]. Вершина v_0 принадлежит только многоугольнику с вершинами 0, 1, 9, 2, 6, 7. Следовательно, бит четности меняется только один раз. Алгоритм сбора семейств проще, чем в плоском случае, так как: во-первых, любые две соседние инцидентные вершины принадлежат треугольной грани, третьей вершиной которой является исследуемая вершина v_0; во-вторых, любое ребро может входить в четное число граней поверхности. В результате получаем семейство контуров многоугольников общего вида.

Алгоритм сбора вершин в семейства инцидентности состоит из следующих восьми шагов.

1. Для вершины v_0 помечаем все инцидентные вершины как непосещенные.

2. Находим непосещенную инцидентную вершину v. Иначе выход из алгоритма.

3. Добавляем пустое семейство инцидентности.

4. Добавляем v в текущее семейство.

5. Помечаем v.

6. Ищем грань, у которой одна вершина v_0, вторая – v, а третья непосещенная вершина v_{new} не входила в список текущего семейства.

7. Если грань найдена, то $v \leftarrow v_{\text{new}}$ и переход на шаг 4.

8. Иначе переход на шаг 2.

Частный случай возникает, если инцидентное ребро совпало с лучом, то есть одна из инцидентных вершин попала на луч (рис. 13).

В этом случае вершина v_0 находится на контуре одного из семейств инцидентных вершин и приходится принимать отложенное решение, которое анализируется так же, как и для многоугольников.

Заслуживает более подробного обсуждения и шаг 2.1, когда луч попадает на ребро, например, вершин 9 и 10 (рис. 14).

В этом случае инцидентными вершинами для вершины v_0 считаются инцидентные вершины концов ребра и достаточно построить только одно семейство инцидентных вершин, начинающееся в третьей вершине одной из инцидентных ребру граней.

Еще один частный случай возникает, если одна из инцидентных вершин, например 0, попадает на ребро (рис. 15).

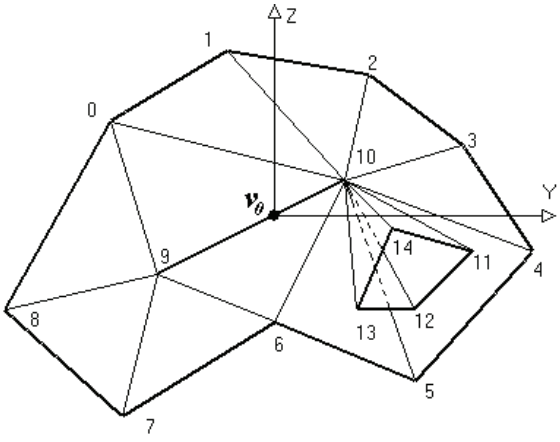


Рис. 14. Луч пересекает ребро (9, 10)

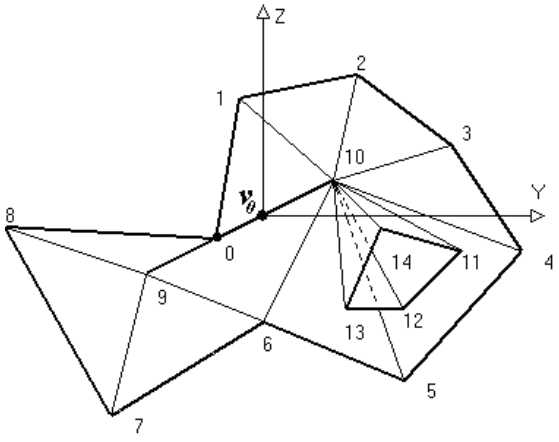


Рис. 15. Одна из инцидентных вершин с номером 0 попадает на ребро, вершина не принадлежит контуру

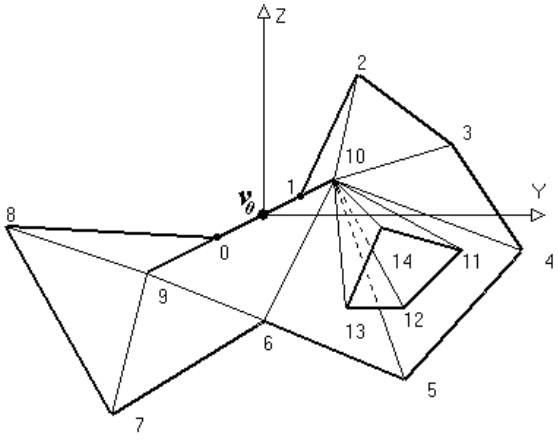


Рис. 16. Одна из инцидентных вершин с номером 0 попадает на ребро, и вершина принадлежит контуру

Если же луч принадлежит грани (0, 1, 10), и вершина v_0 принадлежит контуру семейства, приходится принимать отложенное решение о смене бита, которое анализируется так же, как и для многоугольников.

Время выполнения шагов 1.1 и 2.1 порядка $O(N_v*N_s)$, где N_v – число вершин, N_s – число инцидентных вершин. Число инцидентных вершин в среднем равно шести, поэтому время выполнения этого шага $O(N_v)$.

Время выполнения второго шага $O(N_v)$. Выполнение первого шага можно избежать, если заполнять массивы соседей вершины при реорганизации многоугольника.

Инцидентный лучевой алгоритм не требует упорядоченности ребер по контуру и, следовательно, возможно его применение к задаче определения точки произвольного многогранника в трехмерном пространстве. При анализе попадания вершины на луч задача сводится к определению принадлежности точки многоугольнику.

Список литературы

1. Конноли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика.: Пер. с англ. / Т. Конноли, К. Бегг. – 3-е изд. – М. : Вильямс, 2003. – 1440 с.

2. Ласло, М. Вычислительная геометрия и компьютерная графика на С++ / М. Ласло. – М. : БИНОМ, 1997. – 304 с.

3. Препарата Ф. Вычислительная геометрия: введение / Ф. Препарата, М. Шеймос. – М. : Мир, 1989. – 478 с.

4. Рвачев, В.Л. Методы алгебры логики в математической физике / В.Л. Рвачев. – Киев : Наукова думка, 1974. – 259 с.

5. Скворцов, А.В. Алгоритмы построения и анализа триангуляции / А.В. Скворцов, Н.С. Мирза. – Томск : Изд-во Том. ун-та, 2006. – 167 с.

6. Шикин, Е.В. Компьютерная графика. Полигональные модели / Е.В. Шикин, А.В. Боресков. – М. : ДИАЛОГ-МИФИ, 2005. – 223 с.

Algorithm of Point Recognition of General Polygon or Triangular Face Polygon

N.A. Tyuhachev

Department "Programming and Information Technology", VSU;
nik@cs.vsu.ru

Key words and phrases: geometric algorithms; graph algorithms.

Abstract: The paper presents the algorithm for point recognition of general polygon or triangular face polygon; it is based on the local analysis of incidental vertex sets. The algorithm unlike those offered before doesn't require global order of edges and can be used in three-dimensional space for over-layer operations with layers in geo-information systems.

Algorithmus der Bestimmung der Zugehörigkeit des Punktes zum Vieleck des gemeinsamen Gestaltes oder zum Polyeder mit den dreieckigen Kanten

Zusammenfassung: Es wird den auf der Lokalanalyse der Familien der Inzidentspitzen gestützte Algorithmus der Bestimmung der Zugehörigkeit des Punktes zum Vieleck des gemeinsamen Gestaltes oder zum Polyeder mit der triangulierten Oberfläche vorgeschlagen. Der Algorithmus, zum Unterschied von früher vorschlagenden, fordert keine Ordnungsmäßigkeit der Kanten und kann im dreidimensionalen Raum für die Überlagerungsoperationen mit den Schichten in den Geoinformationssystemen benutzt werden.

Algorithme de la définition de l'appartenance du point au polygone du genre général ou bien au polyèdre aux bornes triangles

Résumé: Est propsé l'algirithme de la définition de l'appartenance du point au polygone du genre général ou bien au polyèdre avec la surface triangulaire basé sur une analyse locale des familles des sommets incidents. En différence de ceux proposés auparavant l'algorithme ne demande pas de l'ordre des arêtes et peut être utilisé dans une surface à trois dimensions pour les opérations «overlay» avec les couches dans les systèmes géoinformatiques.

Автор: ***Тюкачев Николай Аркадиевич*** – кандидат физико-математических наук, доцент кафедры «Программирование и информационные технологии», ГОУ ВПО «ВГУ».

Рецензент: ***Артемов Михаил Анатольевич*** – доктор физико-математических наук, профессор, заведующий кафедрой «Программное обеспечение и администрирование информационных систем», ГОУ ВПО «ВГУ».
