УДК 681.3.513

Е.А. Никулин

МОДЕРНИЗАЦИЯ АЛГОРИТМА КОНТРОЛИРУЕМОЙ ТРИАНГУЛЯЦИИ ПОЛИГОНОВ

Нижегородский государственный технический университет им. Р.Е. Алексеева

Совершенствуется итерационный алгоритм разрезания произвольного полигона на множество треугольников с длинами сторон, не превышающими заданного значения.

Ключевые слова: полигон, триангуляция, разрезание, итерация.

Операция разрезания полигонов (многоугольников) широко распространена в различных прикладных программах и системах автоматизированного проектирования. Она используется в следующих целях:

- для разделения полигона на независимые фрагменты, с которыми далее будут выполняться индивидуальные преобразования;
- замены невыпуклого полигона объединением выпуклых фрагментов, с которыми все тестовые и расчетные задачи выполняются проще и быстрее;
- представления произвольных полигонов объединением простейших *выпуклых* фигур – *треугольников* – с целью унификации и максимального упрощения большинства полигональных операций.

Под *триангуляцией* будем понимать операцию разрезания произвольного, возможно, невыпуклого полигона на треугольники. Ее можно выполнить как рекурсивным [1], так и итерационным методом: поочередным отрезанием выступающих треугольников диагоналями до тех пор, пока не останется последний треугольник. Общий недостаток этих методов состоит в неконтролируемой возможности получения разновеликих и чересчур вытянутых треугольников.

В [1] разработан метод триангуляции произвольного полигона $\mathbf{P}=\mathbf{p}_1\mathbf{p}_2\ldots\mathbf{p}_n\mathbf{p}_1$ с *контролируемыми размерами* отрезаемых треугольников, у которых длины сторон не превышают заданного габаритного параметра h. Идея метода, эскизно и без детализации намеченная в [2], заключается в циклическом поиске у полигона *выпуклой* вершины $\mathbf{p}_m$ с *минимальным* внутренним углом (рис. 1)

$$\psi \equiv |\varphi_m| = \min_{i\in[1,n]} \{|\varphi_i|\}, \quad \text{где} \quad \varphi_i = \angle(\mathbf{p}_{i-1} - \mathbf{p}_i\ ,\ \mathbf{p}_{i+1} - \mathbf{p}_i) \tag{1}$$

и последующем отрезании от него треугольников с длинами сторон, не превышающими h.

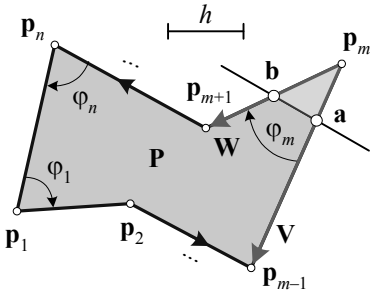

Рис. 1

Модернизация разработанного в [1] алгоритма заключается в замене цикла перебора вершин полигона $\mathbf{P}=\mathbf{p}_1\mathbf{p}_2\ldots\mathbf{p}_n\mathbf{p}_1$ их левым циклическим сдвигом с помощью функции

$$LCShift(\mathbf{P}) = \mathbf{p}_2\mathbf{p}_3\ldots\mathbf{p}_n\mathbf{p}_1\mathbf{p}_2, \tag{2}$$

возвращающей список вершин полигона **P**, ограниченный вершиной $\mathbf{p}_2$. Благодаря этому алгоритм работает с фиксированным индексом $m = 1$. Поиск минимального выпуклого угла теперь основан на n циклических сдвигах $\mathbf{P} = LCShift(\mathbf{P})$ и запоминании конфигурации нового полигона **P** при каждом уменьшении модуля его первого угла $\psi = |\varphi_1|$.

В эффективной контролируемой триангуляции число отрезанных треугольников должно быть минимальным, а их форма близка к правильной с длинами сторон, максимально близкими к h, и внутренними углами, близкими к 60°. Для реализации этих требований будем определять число треугольников, отрезаемых от угла $\mathbf{p}_n\mathbf{p}_1\mathbf{p}_2$, в зависимости от величины ψ (рис. 2): один при $\psi \le 60°$, два при $60° < \psi \le 120°$ либо три при $120° < \psi < 180°$.

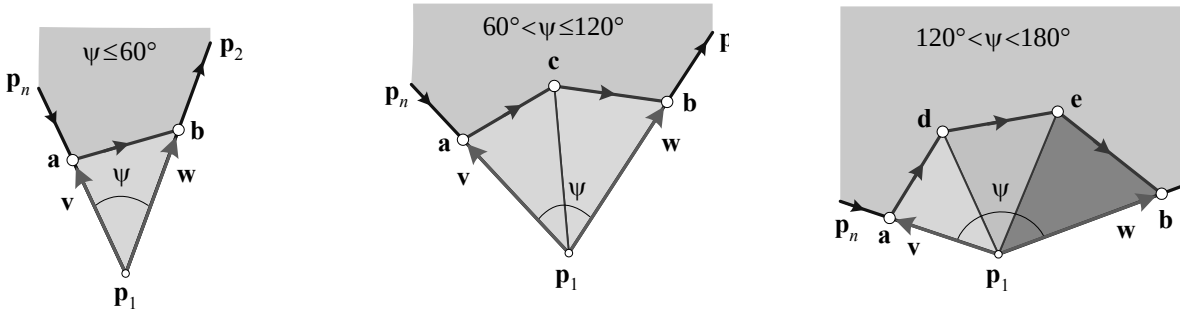

Рис. 2

Выпустим из точки $\mathbf{p}_1$ в направлении вершин $\mathbf{p}_n$ и $\mathbf{p}_1$ векторы **v** и **w** с длинами $|\mathbf{v}| \le h$ и $|\mathbf{w}| \le h$. Расположим на концах векторов точки **a** и **b**, а между ними *равномерно* распределим точки **c**, **d** и **e**. Равномерность достигается помощью параметрической функции

$$\mathbf{u}(\mathbf{v},\mathbf{w},\lambda,\psi) = r(\lambda)\bar{\mathbf{v}}\mathbf{R}(-\lambda\psi), \tag{3}$$

где $\mathbf{R}(\alpha)$ — матрица вращения на угол α, а $\bar{\mathbf{v}} = \mathbf{v}/|\mathbf{v}|$ означает нормировку вектора **v**. Функция (3) возвращает вектор, длина которого $r(\lambda) = (1 - \lambda)\,|\mathbf{v}| + \lambda\,|\mathbf{w}|$ и угол $\gamma = -\lambda\psi$ отклонения от вектора **v** являются линейными функциями параметра λ. При изменении последнего в интервале $\lambda \in [0,1]$ вектор **u** переходит из положения **v** в положение **w**, равномерно увеличивая угол от 0 до ψ и длину от $|\mathbf{v}|$ до $|\mathbf{w}|$. Сопоставим глобальному направлению обхода вершин полигона **P** индикатор $\mathbf{D} \in \{1, -1\}$, возвращаемый функцией *dir_test*(**P**) из [1]. Тогда вершины отрезаемых треугольников будут расположены в следующих точках:

$$\mathbf{a} = \mathbf{p}_1 + \mathbf{v},\ \ \mathbf{b} = \mathbf{p}_1 + \mathbf{w}, \tag{4а}$$

$$\mathbf{c} = \mathbf{p}_1 + \mathbf{u}(\mathbf{v},\mathbf{w},0.5,D\psi), \tag{4б}$$

$$\mathbf{d} = \mathbf{p}_1 + \mathbf{u}(\mathbf{v},\mathbf{w},1/3,D\psi), \quad \mathbf{e} = \mathbf{p}_1 + \mathbf{u}(\mathbf{v},\mathbf{w},2/3,D\psi). \tag{4в}$$

Отрезанные треугольники добавляются в список треугольников **LT**, предварительно инициализированный пустым, а оставшаяся часть полигона поступает на следующую итерацию разрезания. Для недопущения излишнего дробления и уменьшения общего числа треугольников целесообразно по достижении расстояния $|\mathbf{a} - \mathbf{b}| \le h$ отрезать всегда один треугольник $\mathbf{p}_1\mathbf{bap}_1$ независимо от величины угла ψ. Так как при этом $|\mathbf{a} - \mathbf{p}_1| \le h$ и $|\mathbf{b} - \mathbf{p}_1| \le h$, то он удовлетворяет габаритному ограничению и не нуждается в разрезании на два или три.

В зависимости от формы и размеров полигона возможны ситуации (рис. 3), когда отрезок **ab** или полилинии **acb**, **adeb** пересекают другие ребра полигона, либо вычисленные по (4) точки **c**, **d** или **e** оказываются за его пределами. В результате новый полигон перестает быть частью старого. Он становится самопересекающимся и может инвертировать направление обхода. Понятно, что дальнейшая триангуляция такого полигона невозможна.

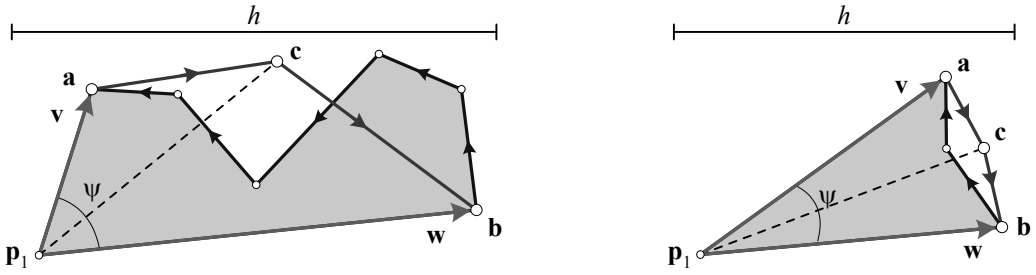

Рис. 3

Предлагается следующее решение этой проблемы. Перед отрезанием треугольников сохраним копии полигона **Q** = **P** и списка отрезанных треугольников **LQ** = **LT**, а после отрезания проверим новый полигон **P** на самопересечение или самокасание тестом *self_test*, определенным в [1]. Он возвращает число 1, если полигон имеет хотя бы одну пару пересекающихся либо касающихся *несмежных* ребер. При *self_test*(**P**) = 1 либо *dir_test*(**P**) ≠ *D* восстановим сохраненные в **Q** и **LQ** данные, уменьшим в (3) длины векторов **v** и **w**, после чего будем отрезать треугольники с новыми вершинами (4) до тех пор, пока полигон **P** не перестанет самопересекаться или изменять направление обхода.

На рис. 4 построена блок-схема алгоритма *contri_poly*(**P**, *h*) контролируемой триангуляции полигона **P** с габаритным параметром *h*. В алгоритме используется несколько вспомогательных функций, определенных в [1]:

- *size*(**P**) — вычисляет размер полигона **P**, возвращая число его сторон *n*;
- *min_poly*(**P**) — удаляет из полигона **P** кратные и коллинеарные вершины;
- *ang*(**V**,**W**) — возвращает минимальный *алгебраический* угол между векторами **V** и **W**.

Работа алгоритма начинается с инициализации пустого списка **LT** = ∅, предназначенного для хранения отрезанных треугольников, и вычисления индикатора направления обхода *D* = *dir_test*(**P**), необходимого в (4), для отбора выпуклых углов по условию $D\varphi_1 < 0$ и обнаружения инверсии направления полигона **P** после недопустимого отрезания (рис. 3).

Дальнейший процесс выполняется циклически до отрезания последнего треугольника. После каждого отрезания полигон изменяет свою конфигурацию, число вершин *n* и их список **P** = $\mathbf{p}_1\mathbf{p}_2\ldots\mathbf{p}_n\mathbf{p}_1$.

Рассмотрим действия, выполняемые в одном цикле алгоритма.

Шаг 1. Зафиксируем число *n* = *size*(**P**) сторон минимального полигона **P** = min_*poly*(**P**), сохраним его копию **Q** = **P** и установим начальное значение угла $\psi = \pi$.

Шаг 2. В цикле $i=\overline{1, n}$ вычисляем внутренний угол $\varphi_1 = ang(\mathbf{q}_n - \mathbf{q}_1, \mathbf{q}_2 - \mathbf{q}_1)$ при первой вершине полигона **Q** = $\mathbf{q}_1\ldots\mathbf{q}_n\mathbf{q}_1$. Если он выпукл ($D\varphi_1 < 0$) и минимален ($|\varphi_1| < \psi$), то запоминаем текущий минимум $\psi = |\varphi_1|$ и полигон **P** = **Q**. Далее выполняется циклический сдвиг полигона **Q** = *LCShift*(**Q**).

Теперь разместим на сторонах угла $\mathbf{p}_n\mathbf{p}_1\mathbf{p}_2$ точки **a** и **b**, отстоящие от вершины $\mathbf{p}_1$ не дальше, чем на расстояние *h*. Проще всего отложить от $\mathbf{p}_1$ отрезки длиной *h*, но тогда в конце отрезания ребра, длина которого не кратна *h*, может остаться очень короткий отрезок, что, в свою очередь, даст мелкий треугольник, длины сторон угла могут быть меньше *h*.

Процедура *равномерного* разбиения ребер полигона состоит из начальных установок некоторых величин и их циклических настроек.

Шаг 3. Установочная фаза включает:

- вычисление векторов сторон полигона $\mathbf{p}_1\mathbf{p}_n$ и $\mathbf{p}_1\mathbf{p}_2$:
$$\mathbf{V} = \mathbf{p}_n - \mathbf{p}_1, \quad \mathbf{W} = \mathbf{p}_2 - \mathbf{p}_1;$$
- расчет целых кратностей длин сторон к числу h:
$$k_{\mathbf{v}} = \left\lceil \frac{|\mathbf{V}|}{h} \right\rceil, \quad k_{\mathbf{w}} = \left\lceil \frac{|\mathbf{W}|}{h} \right\rceil;$$
- расчет целой кратности угла ψ к углу 60°:
$$k = \left\lceil \frac{3\psi}{\pi} \right\rceil;$$
- сохранение копий полигона и списка треугольников:
$$\mathbf{Q} = \mathbf{P}, \quad \mathbf{LQ} = \mathbf{LT}.$$

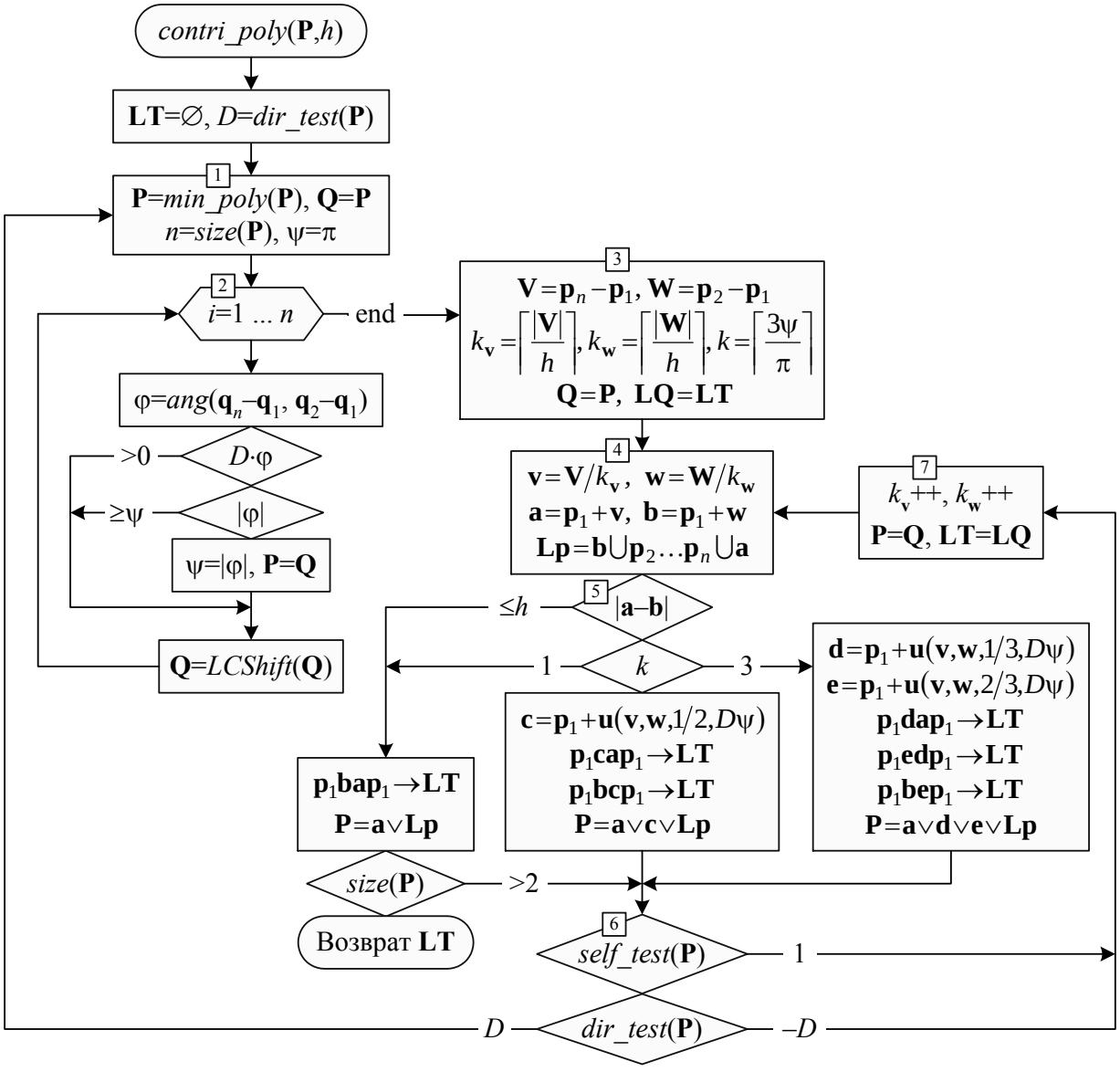

Рис. 4

Шаг 4. Циклическая фаза разбиения ребер $\mathbf{p}_1\mathbf{p}_n$ и $\mathbf{p}_1\mathbf{p}_2$ включает:

- вычисление векторов с длинами, не превышающими значения h:
$$\mathbf{v} = \frac{\mathbf{V}}{k_{\mathbf{v}}}, \quad \mathbf{w} = \frac{\mathbf{W}}{k_{\mathbf{w}}};$$
- вычисление по (4а) вершин треугольников, лежащих на смежных с $\mathbf{p}_1$ ребрах:

$$\mathbf{a} = \mathbf{p}_1 + \mathbf{v},\ \ \mathbf{b} = \mathbf{p}_1 + \mathbf{w};$$

- формирование списка *некратных* вершин полилинии, проходящей между точками **b** и **a** с учетом возможного совпадения точек $\mathbf{b} = \mathbf{p}_2$ и $\mathbf{a} = \mathbf{p}_n$:

$$\mathbf{Lp} = \mathbf{b} \cup \mathbf{p}_2 \ldots \mathbf{p}_n \cup \mathbf{a}. \qquad (5а)$$

В программировании логические объединения можно заменить *конкатенациями* (соединениями) с проверками совпадений точек, возникающих при $k_{\mathbf{w}} = 1$ или $k_{\mathbf{v}} = 1$:

$$\begin{cases} \mathbf{Lp} = \mathbf{p}_2\mathbf{p}_3 \ldots \mathbf{p}_n; \\ \mathbf{Lp} = \text{if}\,(k_{\mathbf{w}} = 1,\ \mathbf{Lp},\ \mathbf{b} \vee \mathbf{Lp}); \\ \mathbf{Lp} = \text{if}\,(k_{\mathbf{v}} = 1,\ \mathbf{Lp},\ \mathbf{Lp} \vee \mathbf{a}). \end{cases} \qquad (5б)$$

Шаг 5. В зависимости от длины отрезка **ab** и коэффициента *k*, вычисленного на шаге 3, возможны три варианта отрезания треугольников и формирования списка вершин нового полигона **P** (см. рис. 2):

- при $|\mathbf{a} - \mathbf{b}| < h$ либо $k = 1$, т. е. $\psi \le 60°$, добавляем в список **LT** *один* треугольник $\mathbf{p}_1\mathbf{bap}_1$, а новый полигон **P** формируем конкатенацией точки **a** и списка **Lp**, полученного в (4). После добавления в **LT** последнего отрезанного треугольника полигон вырождается в двунаправленный отрезок $\mathbf{P} = \mathbf{p}_1\mathbf{p}_2\mathbf{p}_1$. Поэтому по условию *size*(**P**) < 3 алгоритм заканчивает работу и возвращает список **LT** с отрезанными треугольниками;
- при $k = 2$, т. е. $60° < \psi \le 120°$, вычисляем по (4б) точку **c**, добавляем в **LT** *два* треугольника $\mathbf{p}_1\mathbf{cap}_1$ и $\mathbf{p}_1\mathbf{bcp}_1$, а полигон **P** формируем конкатенацией точек **a**, **c** и списка **Lp**;
- при $k = 3$, т. е. $120° < \psi < 180°$, вычисляем по (4в) точки **d** и **e**, добавляем в список **LT** *три* треугольника $\mathbf{p}_1\mathbf{dap}_1$, $\mathbf{p}_1\mathbf{edp}_1$ и $\mathbf{p}_1\mathbf{bep}_1$, а полигон **P** формируем конкатенацией точек **a**, **d**, **e** и списка **Lp**.

Шаг 6. Проверяем самопересечение нового полигона **P**:

$$s = self\_test(\mathbf{P}).$$

При $s = 0$ и $dir\_test(\mathbf{P}) = D$ переходим на шаг 1.

Шаг 7. В противном случае:

- инкрементируем коэффициенты разбиения ребер:

$$k_{\mathbf{v}}{+}{+}, \quad k_{\mathbf{w}}{+}{+};$$

- восстанавливаем копии полигона и списка треугольников:

$$\mathbf{P} = \mathbf{Q}, \quad \mathbf{LT} = \mathbf{LQ};$$

- переходим на шаг 4.

По окончании все возвращенные в списке **LT** треугольники имеют такое же направление обхода, что и у исходного полигона **P**. Это обеспечивается порядком следования вершин треугольников, добавляемых в **LT** на шаге 5.

Проиллюстрируем работу алгоритма контролируемой триангуляции двумя сериями примеров. В первой серии на рис. 5 один и тот же полигон разрезан при разных значениях *h* на 714, 166, 49 и 10 треугольников. Порядок отрезания треугольников от наименьшего выпуклого угла полигона можно проследить на двух последних рисунках.

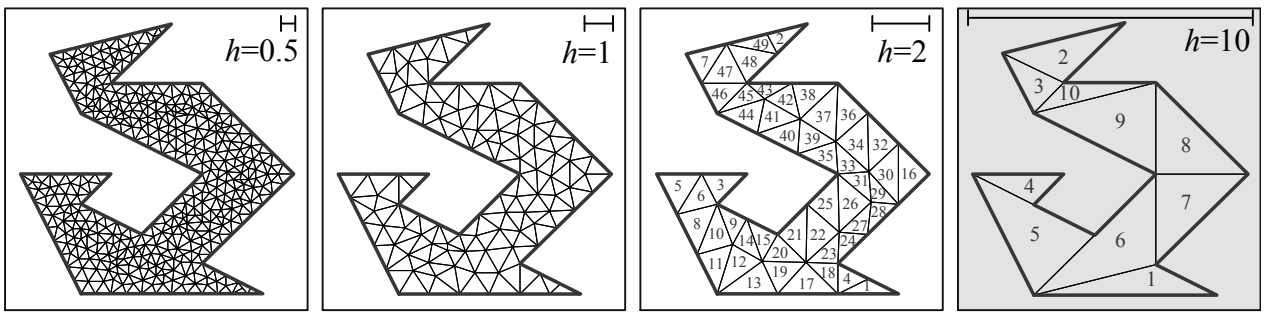

Рис. 5

Отметим, что при задании параметра *h* заведомо бóльшим габаритов полигона размеры отрезаемых треугольников ограничиваются лишь длинами его ребер.

Таким образом, функцию *contri_poly* можно использовать для разрезания полигона диагоналями на максимально крупные треугольники, что иллюстрируется на рис. 5 решением со значением габаритного параметра $h = 10$.

Во второй серии примеров триангуляции подвергались два случайных полигона, сгенерированных алгоритмом *rpoly* из [1], и один регулярный полигон с внутренним отверстием (рис. 6). Последняя конфигурация требует дополнительной подготовки данных: нужно соединить внешний и внутренний контуры по диагонали двумя ребрами для получения одного замкнутого контура. Во избежание самокасания полигона эти ребра следует отодвинуть друг от друга на малое расстояние путем соответствующего изменения координат соединяемых вершин внешнего и внутреннего контуров. Направление обхода внутреннего контура должно быть противоположным направлению обхода внешнего. Если у полигона несколько внутренних отверстий, то все они должны быть соединены дополнительными ребрами в один *D*-ориентированный контур, где *D* – индикатор направления обхода внешнего контура полигона.

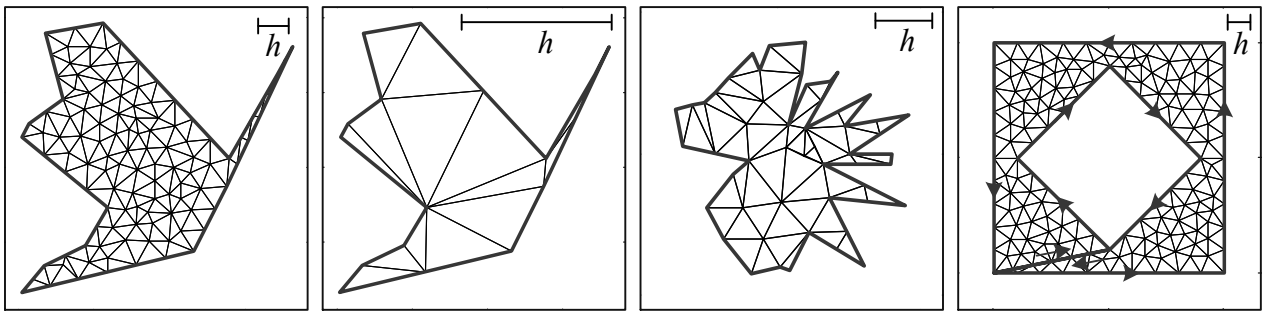

Рис. 6

Выводы

Результатом проведенного исследования является алгоритм разрезания произвольного полигона, автоматизирующий процесс изготовления широкой номенклатуры географических карт. Эта способность разработанного алгоритма решать задачу триангуляции полигона диагоналями, несомненно, повышает его универсальность и полезность.

Библиографический список

1. **Никулин, Е.А.** Компьютерная геометрия и алгоритмы машинной графики: учеб. пособие для вузов / Е.А. Никулин. – СПб.: БХВ-Петербург, 2005. − 560 с.
2. Математика и САПР: [пер. с франц.]; В 2-х кн. Кн. 1 / П. Шенен [и др.] – М.: Мир, 1988. − 204 с.

Дата поступления
в редакцию 28.04.2011

E.A Nikulin

MODERNIZATION OF THE ALGORITHM OF MANAGED POLYGON TRIANGULATION

This article describes an improvement of iterational algorithm of cutting an arbitrary polygon in the set of triangles with a limited upper-bound length of their edges.

Key words: polygon, triangulation, cutting, iteration.