УДК 681.3.513

Е.А. Никулин

ГЕНЕРАТОР АБСОЛЮТНО СЛУЧАЙНЫХ ПОЛИГОНОВ

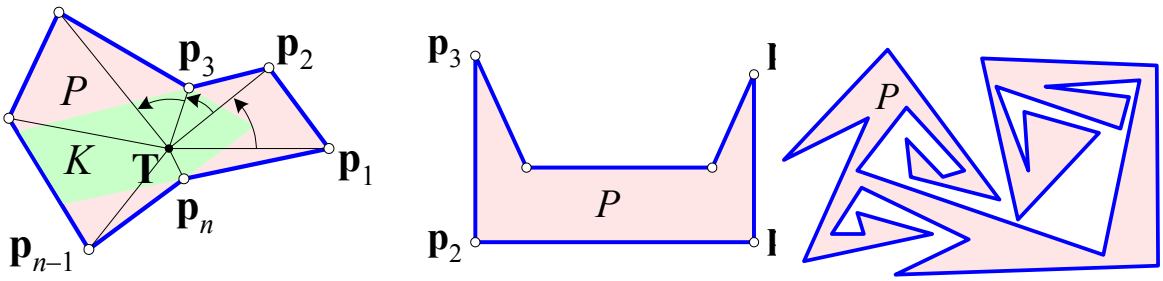
Нижегородский государственный технический университет им. Р.Е. Алексеева

Тема работы: разработка алгоритма построения полигона со случайными длинами и направлениями ребер.
Цель работы: пополнение библиотеки моделей и алгоритмов компьютерной графики.
Метод решения: Имитация броуновского движения по замкнутой цепочке случайных не пересекающихся отрезков.
Оригинальность: Способность алгоритма генерировать лабиринтные полигоны.
Выводы: В работе получен алгоритм построения полигонов абсолютно непредсказуемой формы.

Ключевые слова: отрезок, полигон, вершина, случайное число.

В компьютерной графике для отладки различных тестов и алгоритмов используются разнообразные графические объекты, из которых особенно востребованы полигоны (многоугольники). Данные плоские фигуры определяются координатами вершин в порядке их соединения и могут быть созданы как вручную, так и алгоритмически, как детерминированно, так и случайно. Последний способ генерирования полигонов особенно ценен, поскольку он позволяет автоматически без утомительного ручного ввода вершин быстро получить множество разных форм, пройти в совокупности по всем критичным ветвям тестируемого алгоритма и провести статистическую обработку результатов тестирования.

В [1] рассмотрено несколько алгоритмов генерирования полигонов как выпуклых (функции *conv_poly*, *cover*2 и *rcpoly*), так и невыпуклых (функции *poly_points* и *rpoly*). В последнем алгоритме вершины полигона создаются методом вращения луча на один оборот вокруг заданной, возможно, случайно, точки **T** на случайные углы и откладывания вдоль него отрезков случайной длины. Все случайные числа генерируются в заданных интервалах. За одно обращение функция *rpoly* возвращает случайный так называемый *звездный* полигон $P = \mathbf{p}_1\mathbf{p}_2\ldots\mathbf{p}_n\mathbf{p}_1$, имеющий ядро K, из всех точек которого напрямую видны все вершины (рис. 1, *а*). У каждого выпуклого полигона ядро не только существует, но и совпадает с полигоном, однако, произвольный невыпуклый полигон может и не иметь ядра (рис. 1, *б*), и он не способен сгенерироваться алгоритмом *rpoly*. Совершенно очевидно отсутствие ядер у лабиринтных полигонов (рис. 1, *в*).



а) *б*) *в*)

Рис. 1. Типы невыпуклых полигонов:
а – звездный; *б* – безъядерный; *в* – лабиринтный

В данной работе рассматривается алгоритм построения *абсолютно* случайного полигона методом, имитирующим процесс дискретного броуновского движения на плоскости в

случайных направлениях со случайными длинами шагов. Полилиния такого хаотического блуждания должна удовлетворять двум условиям:

1) ее несмежные отрезки не должны пересекаться либо касаться друг друга;
2) на последнем шаге ломаная линия должна замкнуться в полигон с совпадающими крайними вершинами $\mathbf{p}_1 = \mathbf{p}_{n+1}$.

Задача оценки взаимного расположения двух отрезков *ab* и *cd* по координатам их концевых точек **a**, **b**, **c** и **d** изучается в [1]. Вводится скалярная функция векторных аргументов

$$f(\mathbf{a},\mathbf{b},\mathbf{p}) = \left|\mathbf{p}-\mathbf{a} \quad \mathbf{b}-\mathbf{a}\right|, \qquad (1)$$

возвращающая определитель квадратной матрицы, составленной из двух столбцовых векторов **p** - **a** и **b** - **a**. Если точка **p** лежит на прямой, проходящей через точки **a** и **b**, то определитель матрицы с линейно-зависимыми столбцами равен нулю. Для всех точек **p**, лежащих справа от направляющего вектора прямой **b** - **a**, функция (1) возвращает положительные числа, а для всех левых точек — отрицательные. Следующий тест взаимного расположения отрезков *ab* и *cd:*

$$segm\_test(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = \left(f(\mathbf{a},\mathbf{b},\mathbf{c})\cdot f(\mathbf{a},\mathbf{b},\mathbf{d}) \le 0\right) \wedge \left(f(\mathbf{c},\mathbf{d},\mathbf{a})\cdot f(\mathbf{c},\mathbf{d},\mathbf{b}) \le 0\right) \qquad (2)$$

возвращает решение системы двух неравенств, проверяющих расположение концов каждого отрезка не по одну сторону от прямой, несущей другой отрезок. Функция *segm_test* возвращает 1 при пересечении либо касании отрезков и 0 – в противном случае.

Абсолютно случайный полигон как выпуклый, так и невыпуклый, как с ядром, так и без ядра, с произвольными углами между ребрами и направлением обхода строится методом имитации процесса шагания по плоскости в случайных направлениях без пересечения пройденного пути и возвратом в исходную точку. Блок-схема функции *arpoly*(**T**,*a*,*b*,*m*,*M*), реализующей этот процесс, приведена на рис. 2.

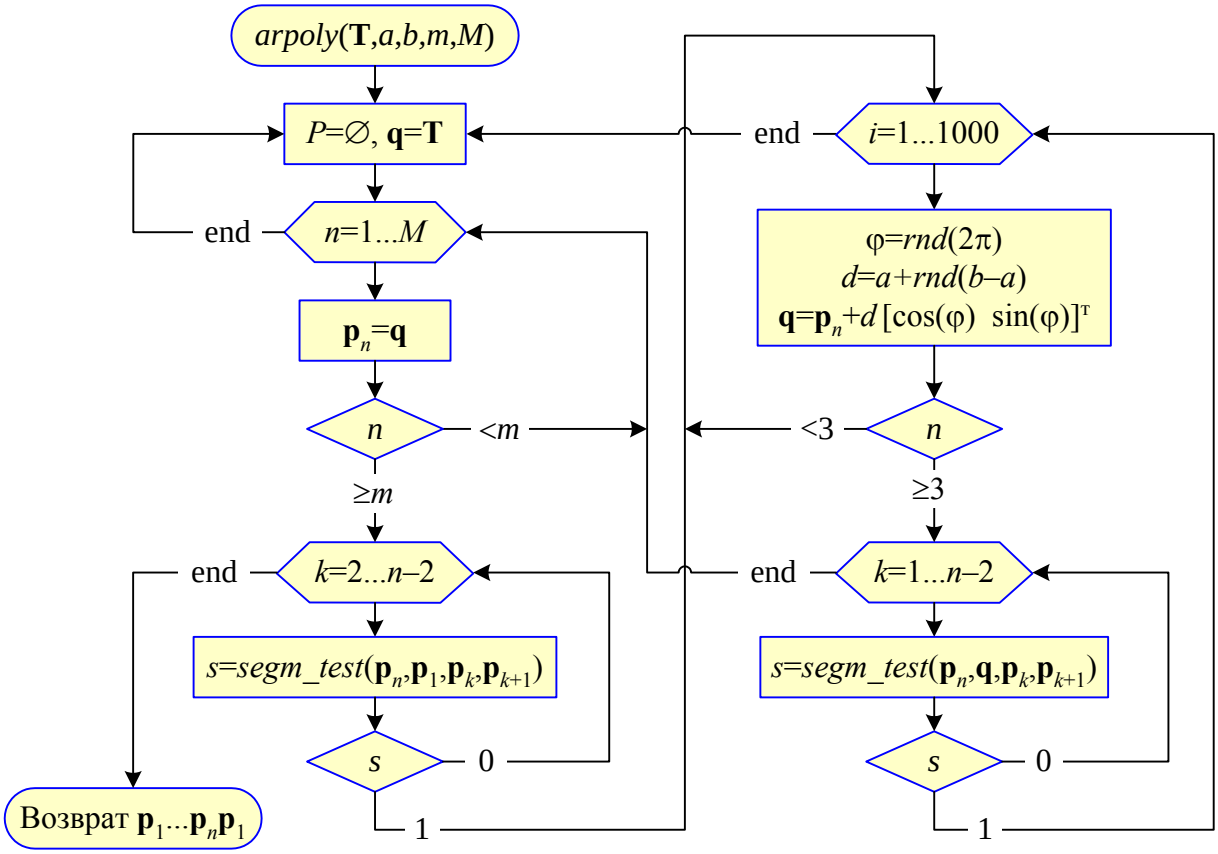


Рис. 2. Блок-схема алгоритма

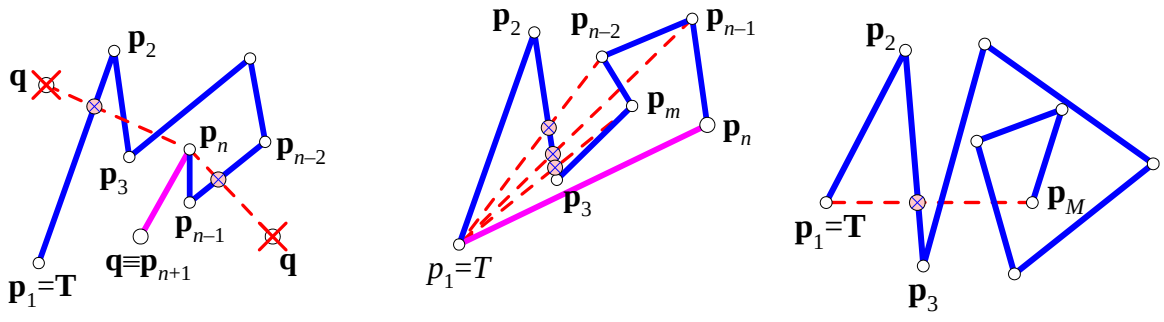
Первый векторный аргумент **T** задает расположение начальной вершины полигона $\mathbf{p}_1$ (может быть задана детерминированно либо случайно), следующая пара аргументов a и b определяет диапазон изменения длины шага $d \in (a, b)$, четвертый аргумент $m \geq 3$ задает минимальный номер вершины $\mathbf{p}_m$, с которой начинается проверка возможности замыкания полилинии ребром $\mathbf{p}_m\mathbf{p}_1$, а пятый аргумент $M \geq m$ ограничивает максимальное число шагов хаотического блуждания для предотвращения его зацикливания в случае невозможности замыкания полилинии из глубин построенного лабиринта.

Главные действия алгоритма происходят в цикле $n=\overline{1,M}$:

- поиск вершины $\mathbf{p}_{n+1}$ начинаем с откладывания от точки $\mathbf{p}_n$ отрезка случайной длины $d = a + rnd(b - a)$ под случайным углом $\varphi = rnd(2\pi)$. На конце получаем пробную точку

$$\mathbf{q} = \mathbf{p}_n + d\left[\cos(\varphi) \quad \sin(\varphi)\right]^{\mathrm{T}};$$

- при $n \in \{1, 2\}$ запоминаем точки $\mathbf{p}_{n+1} = \mathbf{q}$, в результате чего создается двухсегментная полилиния $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$;
- при $n \geq 3$ проверяем пересечение отрезка $\mathbf{p}_n\mathbf{q}$ с ранее построенными *несмежными* ребрами $\mathbf{p}_1\mathbf{p}_2 \div \mathbf{p}_{n-2}\mathbf{p}_{n-1}$ (рис. 3, *а*). При первом обнаружении значения $segm\_test(\mathbf{p}_n,\mathbf{q},\mathbf{p}_k,\mathbf{p}_{k+1}) = 1$ повторяем генерацию новой пробной точки **q** до тех пор, пока она не станет подходящей для следующей вершины полигона. Во избежание зацикливания ограничим число пробных повторов достаточно большим числом, например, 1000;
- при $n \geq m$ проверяем полилинию $\mathbf{p}_1\mathbf{p}_2\ldots\mathbf{p}_n$ на возможность замыкания ребром $\mathbf{p}_n\mathbf{p}_1$, не пересекающимся ни с одним из *несмежных* ребер $\mathbf{p}_2\mathbf{p}_3 \div \mathbf{p}_{n-2}\mathbf{p}_{n-1}$ (рис. 3, *б*). При возвращении тестом (2) значения $segm\_test(\mathbf{p}_n,\mathbf{p}_1,\mathbf{p}_k,\mathbf{p}_{k+1}) = 0$ для всех $k = 2 \ldots n - 2$ добавляем к полилинии начальную вершину и возвращаем замкнутый полигон $P = \mathbf{p}_1\mathbf{p}_2\ldots\mathbf{p}_n\mathbf{p}_1$;
- если за все M шагов случайного блуждания полилиния $\mathbf{p}_1\mathbf{p}_2\ldots\mathbf{p}_M$ не может быть замкнута без самопересечений (рис. 3, *в*), то заново повторим процесс шагания от точки **T** до успешного получения $n \in [m, M]$ -стороннего замкнутого полигона $P = \mathbf{p}_1\mathbf{p}_2\ldots,\mathbf{p}_n\mathbf{p}_1$, не забывая предварительно инициализировать пустой список $P = \varnothing$. Такой же повтор следует делать после 1000 неудачных попыток генерации пробных точек.



а) *б*) *в*)

Рис. 3. Типы невыпуклых полигонов

На рис. 4 показан ряд полигонов, сгенерированных алгоритмом $arpoly(\mathbf{T},2,5,m,M)$ с разными значениями m и M. При задании m =3 всегда получаются случайные треугольники.

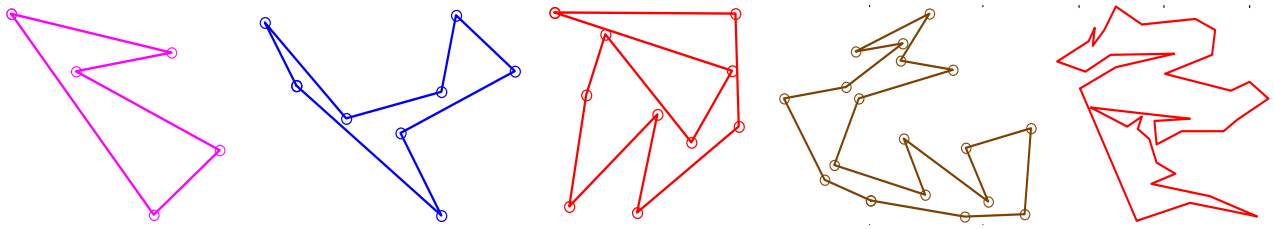


Рис. 4. Результаты работы алгоритма

1. **Никулин, Е.А.** Компьютерная геометрия и алгоритмы машинной графики: учеб. пособие для вузов / Е.А. Никулин. – СПб.: БХВ-Петербург, 2005. – 560 с.

Дата поступления
в редакцию 06.05.2013

E.A. Nikulin

ABSOLUTELY RANDOM POLYGONS GENERATOR

Nizhny Novgorod State Technical University n.a. R.E. Alexeev

Subject: Development of an algorithm to construct polygons with random edge lengths and orientations.
Purpose: Stocking of computer graphics model and algorithm library.
Methodology: Imitation of Brownian movement along a closed chain of random non-intersecting segments.
Originality/value: Ability of the algorithm to generate maze polygons.
Findings: The work had outcome with an algorithm which constructs polygons of absolutely unpredictable configuration.

Key words: segment, polygon, vertex, random number.