General Rasterization Technique for Parametric Curves

Sergey Chirikov (Institute of Fine Mechanics & Optics, St.Petersburg, Russia) <sergchi@arcadia.spb.ru>
Timour Paltashev (S3 Incorporated, Santa Clara, CA, USA) <tpaltash@S3.com>

Общий Метод Растрирования Параметрических Кривых

Чириков С.В. (Институт Точной Механики и Оптики - технический университет, Санкт-Петербург, Россия), Палташев Т.Т. (S3 Incorporated, USA).

Key words: *Bresenham algorithm, line rasterization, lecal curves, L-curves, conic arcs, conic interpolator, linear interpolator, Bezier curve, B-spline, conic spline.*

This paper describes the algorithms and hardware structures of rasterizers able to generate geometrically defined parametric curves in multidimensional space with the speed comparable with Bresenham algorithms for lines and circles.

New class of geometrically defined curves named as lecal curves (L-curves) has been proposed. This class of curves has a high level flexibility in shape control and can be used to rasterize other type of the curves (Bezier, for example). The hardware implementation of L-curve generator includes only two types of simple computational units – linear and conic interpolators linked into hierarchical computational structure.

Ключевые слова: *алгоритм Брезенхема, растрирование линий, лекальные кривые, L-кривые, коническая дуга, конический интерполятор, линейный интерполятор, кривая Безье, B-сплайн, конический сплайн.*

Рассматриваются алгоритмы и вычислительные структуры генераторов графических примитивов вывода, способных выполнять растрирование геометрически заданных кривых в многомерном пространстве со скоростью сравнимой с быстродействием алгоритмов Брезенхема для векторов и окружностей. Предлагается новый класс геометрически кривых, названный лекальными кривыми (L - кривые). Эти кривые обладают высокой гибкостью управления формой кривой. Определяются условия, при которых лекальная кривая эквивалентна кривой Безье. Показывается, что структура генератора лекальных кривых представляет собой иерархическую вычислительную структуру, состоящую из вычислительных элементов двух видов - линейного и конического интерполяторов, допускающих эффективную реализацию в виде БИС.

1. Общая идея метода растрирования параметрических кривых.

В работах [1,7] предложен метод растрирования конических дуг, вписанных в треугольник общего положения. Метод растрирования заключается в параллельной генерации двух векторов, задающих стороны треугольника.

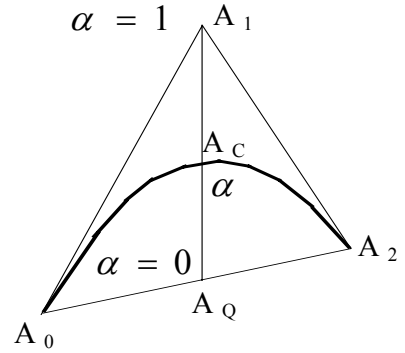

Рис.1. Коническая дуга, вписанная в треугольник.

Структура генератора конических дуг состоит из двух блоков - управляющего блока (реализованного в виде конического интерполятора **CI**) и исполняющего блока (реализованного в виде двух линейных интерполяторов **LI** и сумматора приращений координат). Функциональная структура генератора конических дуг приведена на рис.2.

$$d\vec{A}(t) = \vec{A}_{01}x(t) + \vec{A}_{12}y(t)$$

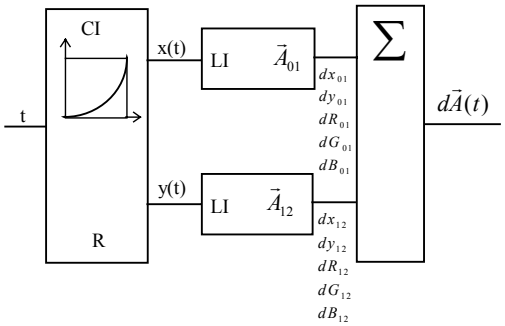

Рис.2. Функциональная схема генератора конических дуг.

Данная схема допускает очевидное обобщение на случай кривой, вписанной в многоугольник общего положения. Необходимо реализовать управляющую структуру, которая будет генерировать несколько сигналов, одновременно изменяющихся в диапазоне [0,1]. Эти сигналы $\varphi_k(t)$, подаваемые на вход нескольких линейных интерполяторов **LI**, будут инициировать генерацию векторов, причем скорость генерации каждого вектора будет определяться только сигналом $\varphi_k(t)$. Математически, такая кривая будет описываться следующей формулой

$$\vec{A}(t) = \vec{A}_0 + \sum_{k=1}^{m} \varphi_k(t)\vec{A}_{k-1,k}\ ; \qquad (1)$$

$$t, \varphi_k(t) \subseteq [0,1].$$

Для практической реализации генератора кривых необходимо предложить достаточно простую, с точки зрения реализации, вычислительную структуру управляющей схемы, которая, в то же время, могла быть использована для растрирования общепринятых в компьютерной графике кривых, в первую очередь - кривых Безье. Функциональная схема такого генератора кривых будет иметь следующий вид.

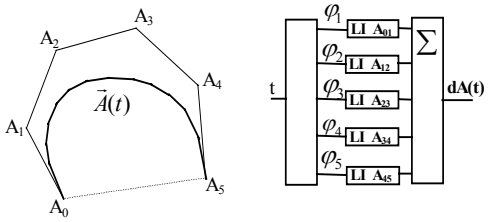

Рис. 3. Функциональная схема генератора кривых, вписанных в многоугольник.

Данная функциональная схема не определяет способ получения управляющих сигналов $\varphi_k(t)$. Строго говоря, можно предложить множество функций, которые можно использовать в качестве управляющих сигналов. Вопрос в алгоритмической реализации этих функций. В работе [1], было показано, что для генерации векторов, управляющие функции должны представлять совой кривые, вписанные в квадрат со стороной R, причем, величина R определялась длиной самого длинного вектора.

$$R = \max\{\|\vec{A}_{01}\|, \|\vec{A}_{12}\|\}, \qquad (2)$$

где $\|\vec{A}\| = \max\{|X|, |Y|, \ldots\}$.

2. Конический интерполятор - базовый элемент генератора лекальных кривых.

Как было показано в [1], для эффективного растрирования пригодны лишь алгебраические кривые второго порядка, т.е., конические дуги. Алгоритм растрирования коник, вписанных в квадрат со стороной R, назван алгоритмом конического интерполятора. Математически работа конического интерполятора описывается следующими функциями

$$x(t) = t - \sigma \cdot sign(1-Z)\sqrt{\sigma(\sigma + 2t(1-t))}; \quad (3)$$

$$y(t) = t + \sigma \cdot sign(1-Z)\sqrt{\sigma(\sigma + 2t(1-t))}, \quad (4)$$

где $x(t), y(t), t \subseteq [0,1]$,

$t = \frac{i}{2R}$; $\quad i \subseteq [0,2R]$ - текущий индекс итерации алгоритма.

$$\sigma = \frac{1}{2} \cdot \frac{1+Z}{1-Z}; \ (Z \neq 1);$$

Этот алгоритм пригоден для аппаратной реализации и по быстродействию сравним с алгоритмом Брезенхема [2] для окружностей. Блок-схема и функциональная схема конического интерполятора показаны на рис. 4. Начальная инициализация алгоритма задается следующими соотношениями:

$$u = 4n + 2m; \qquad (5a)$$

$$v = u - 4R(n+m); \qquad (5b)$$

$$D = (n+m)(1-2R); \qquad (5c)$$

$$k_1 = 4n;\ k_2 = 4m. \qquad (5d)$$

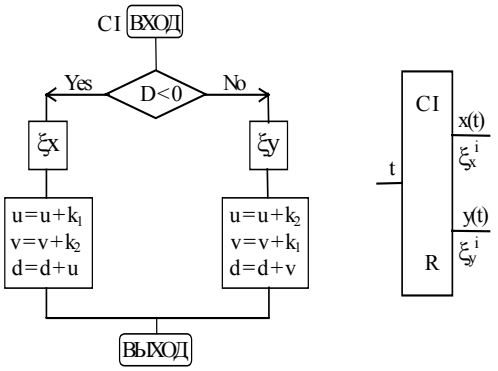

Рис. 4. Блок-схема алгоритма конического интерполятора и его функциональная схема.

Используя конический интерполятор в качестве вычислительного элемента, мы можем построить различные управляющие структуры с однотипной архитектурой. В дальнейшем условимся называть, генерируемые таким образом линии, лекальными кривыми (L-curves).

3. Интерполяционные структуры и лекальные кривые.

Мы показали, что работа конического интерполятора описывается параметрическими функциями $x(t), y(t)$, причем, параметр t линейно зависит от текущего количества входных инициирующих импульсов (итераций алгоритма). Из этого следует, что если любой выход CI соединить с входом другого CI, то с выходов этого второго интерполятора также будут сниматься единичные управляющие сигналы, и эти сигналы будут являться суперпозицией входного сигнала как функции параметра t. Приведем вычислительную структуру, составленную из конических интерполяторов.

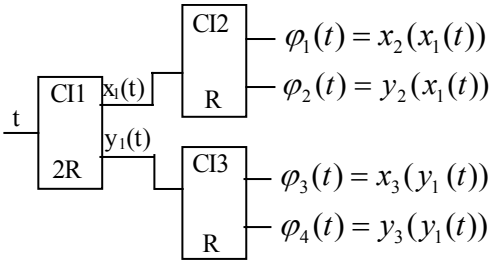

Рис.5. Функциональная схема управляющего модуля генератора лекальных кривых.

Лекальная кривая, построенная на основе такой структуры, математически задается формулой

$$\vec{A}(t) = \vec{A}_0 + \vec{A}_{01}\varphi_1(t) + \vec{A}_{12}\varphi_2(t) + \vec{A}_{23}\varphi_3(t) + \vec{A}_{34}\varphi_4(t). \qquad (6)$$

Функциональная схема генератора лекальных кривых, в этом случае, будет иметь следующий вид:

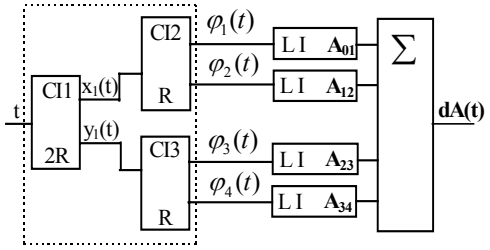

Рис.6. Функциональная схема генератора лекальных кривых.

Лекальная кривая соединяет начальную и конечную точки разомкнутого пятиугольника. В начальной и конечной точках этого многоугольника лекальная кривая имеет касательную параллельную смежному ребру (**A_{01}** и **A_{34}**) многоугольника. Если многоугольник выпуклый, то кривая будет лежать внутри многоугольника.

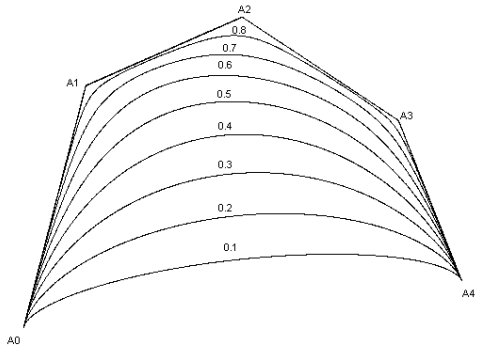

Рис. 7. Пример задания семейства лекальных кривых, вписанных в многоугольник
($\alpha_1 = \alpha_2 = \alpha_3 \subseteq \{0.1, 0.2, \ldots, 0.8\}$).

Свойства лекальной кривой близки к свойствам кривой Безье, однако, в отличие от кривой Безье, поведение которой полностью определяется положением ее вершин, лекальная кривая (рис. 7), кроме вершин, имеет еще три управляющих параметра (эксцентриситеты Z_1,Z_2,Z_3), которые управляют степенью близости линии к задающему многоугольнику. Из приведенного выше рисунка (рис. 7) видно, что если все три эксцентриситета близки к единице (гиперболический режим), лекальная кривая прижимается к ребрам многоугольника. При выборе эксцентриситетов близких к -1 (эллиптический режим), кривая прижимается к ребру, соединяющему начальную и конечную точки многоугольника. В остальных случаях лекальная кривая гибко изменяет свою форму управляемая параметрами Z_1,Z_2,Z_3.

Предложенная методика построения генераторов лекальных кривых из набора конических интерполяторов, позволяет предложить множество вычислительных структур для генерации различных кривых вписанных в многоугольник. Приведем несколько примеров.

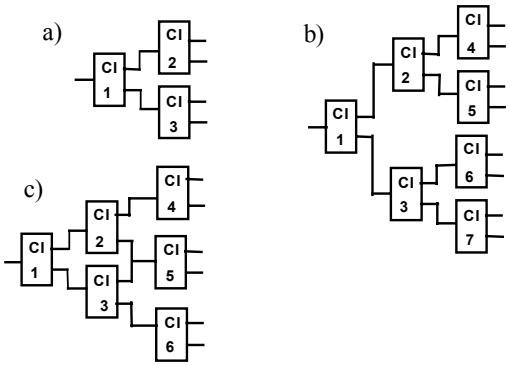

Рис.8. Примеры управляющих структур генератора лекальных кривых.

К выходам интерполяционной структуры должны подключаться входы генераторов векторов, а генерируемая лекальная кривая будет вписана в разомкнутый многоугольник, число ребер которого равно числу выходов управляющей интерполяционной структуры. Условимся в дальнейшем называть подобные вычислительные структуры лекальными интерполяторами.

Генерация лекальной кривой, таким образом, выполняется посредством квази-одновременной параллельной генерации набора векторов - ребер многоугольника. Мы говорим о квази-одновременности, поскольку в каждый момент времени, соответствующей одному единичному импульсу на входе лекального интерполятора, снимается сигнал лишь с одного выхода интерполяционной структуры и выполняется приращение вдоль лишь одного вектора. Распределенные вычисления, реализуемые лекальным интерполятором, даже при его аппаратной реализации по сути дела являются вычислениями с разделением времени. Нетрудно заметить, что **CI** второго уровня (CI2,CI3) простаивают 50% времени работы лекального интерполятора**, CI** третьего уровня (CI5, ..., CI7) простаивают 75% времени каждый и так далее, по мере усложнения управляющей структуры. Из этого наблюдения следует, что слишком усложнять управляющую структуру генератора лекальных кривых нецелесообразно.

Оценим время генерации конической кривой предложенным методом. Из иерархической структуры лекального интерполятора видно, что интерполяторы нижнего уровня должны получать на вход вдвое меньшее число входных импульсов, чем управляющий ими **CI**. Однако, очевидно, что конические интерполяторы низшего уровня, управляющие генерацией векторов должны получить на вход, как минимум, столько сигналов, какова длина генерируемого ими вектора. Число входных и выходных сигналов интерполятора жестко связано с размером R квадрата, в который вписана управляющая коника. Найдем эту величину из условия

$$R_0 = \max\{\|\vec{A}_{01}\|, \|\vec{A}_{12}\|, \|\vec{A}_{23}\|, \|\vec{A}_{34}\|, \ldots\}, \qquad (7)$$

где: $\|\vec{A}\| = \max(|A_x|, |A_y|, |A_z|, \ldots)$.

Таким образом, **CI** нижнего уровня имеет $R=R_0$, Конический интерполятор следующего уровня $R=2R_0$, следующий $R=4R_0$, и так далее. Переход на следующий уровень требует удвоения величины R и, следовательно, удвоения времени работы лекального интерполятора. Для схемы a) $R_1=2R_0$. Для схемы b) $R_1=4R_0$.

4. Нерегулярный характер ступенчатого эффекта.

Мы знаем, что вектор, окружность, эллипс при дискретизации выглядят как набор вертикальных и горизонтальных ступенек. Этот эффект называется ступенчатым эффектом, и он, строго говоря, неустраним. Однако, лестничный эффект для этих кривых носит регулярный, симметричный характер и “не режет глаз”. В случае генератора лекальных кривых, ситуация не такова. Каждый вектор делает свой вклад в создание лестничного эффекта, а так как векторы генерируются попеременно, то и лестничный эффект, являясь суммарным эффектом от нескольких векторов, усиливается в число раз, равное количеству векторов и имеет характерный нерегулярный характер. Если не принимать специальных мер, которые мы обсудим ниже, то лекальная кривая будет выглядеть, как лохматая шерстяная нитка. Пример такой ситуации приведен ниже.

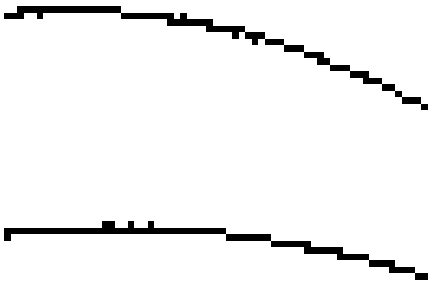

Рис.9. Иллюстрация нерегулярного характера ступенчатого эффекта.

Этот эффект имеет место и в случае конических дуг, однако для лекальных кривых высокого порядка этот эффект становится просто нетерпим. Есть несколько подходов к устранению нерегулярного характера ступенчатого эффекта.

Очевидный способ - измельчение растровой сетки. Вычисления следует проводить с субпиксельной точностью. Собственно, введение субпиксельности неизбежно в любом случае. Каждый вектор при дискретизации дает погрешность, при суммировании приращений векторов, вы суммируем и их погрешности. Отсюда следует, что величина R должно быть увеличена в число раз, равное количеству векторов. Таким образом, для генератора конических дуг эта величина равна 2, для лекальных генераторов построенных по схемам a) и b) (рис. 7) коэффициенты, соответственно, должны быть равны 4 и 8. Отметим, что эта оценка обеспечивает лишь требование, чтобы уклонение пикселей от теоретической кривой не превышало единицы растра.

Практически, генерация кривой с субпиксельной точностью означает увеличение длины вектора в целое число раз, а подсветка пикселя должна выполняться не при каждой итерации, а лишь при каждой второй/четвертой/восьмой ... , в зависимости от выбранного коэффициента расщепления единицы растра. На приведенном ниже рисунке коэффициент расщепления равен 4.

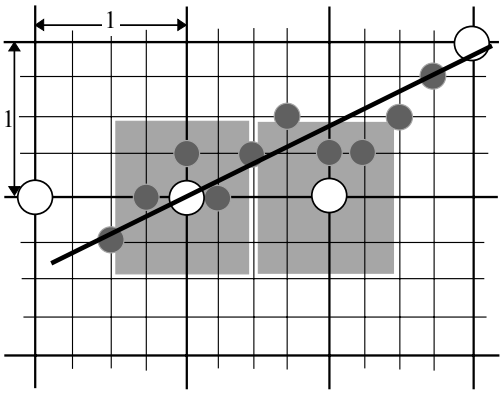

Рис.10. Растрирование кривой с субпиксельной точностью.

В пределах единицы растра можно проинтегрировать все субпиксельные приращения координат и найти усредненное значение координаты пикселя. Такое усреднение также ослабляет нерегулярный характер ступенчатого эффекта. Данный подход, в целом, работоспособен, но приводит к резкому замедлению генерации лекальных кривых. Его следует применять в сочетании с другими методами.

Отметим, что нерегулярность сильнее проявляется, если линейные интерполяторы реализуются на основе алгоритма Брезенхема для векторов и менее заметна, если линейные интерполяторы реализуют ЦДА алгоритм. Причина такого различия заключается в том, что в первом случае координаты векторов сначала округляются до целого значения, а затем суммируются, при этом погрешности округления также суммируются. В случае ЦДА алгоритма сначала суммируются текущие координаты векторов (в формате с фиксированной точкой) и лишь затем выполняется округление до целого значения, при этом погрешность округления - однократна.

Другой способ основан на наблюдении, что нерегулярность имеет место в тех случаях, когда ребра управляющего многоугольника направлены в противоположные стороны. Один вектор увеличивает координату, а на следующей итерации другой вектор ее уменьшает. В результате цепочка смежных пикселей колеблется вдоль теоретического положения линии и создает эффект “шерстяной нити”. Выше было показано, что при расщеплении кривых пополам, они укорачиваются и спрямляются, а, следовательно, направления их ребер сближаются, что приводит к ослаблению, причем существенному, нерегулярности ступенчатого эффекта.

Этот подход, строго говоря, не требует никаких изменений в вычислительной структуре. Он даже ослабляет требования к субпиксельности, однако, при этом он перекладывает часть вычислительных затрат на инициализацию алгоритма. Расщепление кривой на последовательность сопряженных сегментов всегда возможно, однако эта задача должна выполняться в прикладной программе, которая инициализирует аппаратно реализованный генератор кривых.

5. Генерация кривых Безье с помощью генератора лекальных кривых.

Если настроить все конические интерполяторы на гиперболический режим ($Z = 1$),

$$x(t) = 2t - t^2; \qquad y(t) = t^2; \qquad (8)$$

то выходной управляющий сигнал лекального интерполятора будет задаваться следующими формулами

$$\varphi_1(t) = 4t - 6t^2 + 4t^3 - t^4; \quad (9a)$$
$$\varphi_2(t) = 4t^2 - 4t^3 + t^4; \quad (9b)$$
$$\varphi_3(t) = 2t^2 - t^4; \quad (9c)$$
$$\varphi_4(t) = t^4, \quad (9d)$$

а лекальная кривая
$\vec{A}(t) = \vec{A}_0 + \vec{A}_{01}\varphi_1(t) + \vec{A}_{12}\varphi_2(t) + \vec{A}_{23}\varphi_3(t) + \vec{A}_{34}\varphi_4(t)$ будет представлять собой некоторый степенной полином.

Поскольку кривая Безье $\vec{P}_n(t)$ также может быть представлена в форме степенного полинома

$$\vec{P}_n(t) = \sum_{k=0}^{n} \vec{P}_k C_n^k (1-t)^{n-k} t^k = \qquad , \qquad (10)$$

то, приравнивая коэффициенты при одинаковых степенях полиномов $\vec{A}(t)$ и $\vec{P}_4(t)$, получаем линейную систему четырех уравнений. Решая эту систему методом исключения, находим связь между вершинами кривой Безье и эквивалентной ей лекальной кривой

$$\vec{P}_0 = \vec{A}_0; \qquad \vec{P}_1 = \vec{A}_1; \qquad (11a)$$
$$\vec{P}_2 = \frac{1}{3}(\vec{A}_1 + \vec{A}_2 + \vec{A}_3); \qquad (11b)$$
$$\vec{P}_3 = \vec{A}_3. \qquad (11c)$$

Аналогичные соотношения можно получить и для кривой Безье третьего порядка

$$\vec{P}_0 = \vec{A}_0; \qquad (12a)$$
$$\vec{P}_1 = \frac{1}{4}(\vec{A}_0 + 3\vec{A}_1); \qquad (12b)$$
$$\vec{P}_2 = \frac{1}{4}(\vec{A}_4 + 3\vec{A}_3); \ \vec{P}_3 = \vec{A}_4. \qquad (12c)$$

Пересчет вершин кривой Безье в вершины эквивалентной ей лекальной кривой удобно представить матричным преобразованием **A**=G**P**

$$\begin{pmatrix} \vec{A}_{01} \\ \vec{A}_{12} \\ \vec{A}_{23} \\ \vec{A}_{34} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \vec{P}_{01} \\ \vec{P}_{12} \\ \vec{P}_{23} \\ \vec{P}_{34} \end{pmatrix};$$

$$\begin{pmatrix} \vec{A}_{01} \\ \vec{A}_{12} \\ \vec{A}_{23} \\ \vec{A}_{34} \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 3 & 0 & 0 \\ 2 & 2 & -1 \\ -1 & 2 & 2 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} \vec{P}_{01} \\ \vec{P}_{12} \\ \vec{P}_{23} \end{pmatrix}.$$

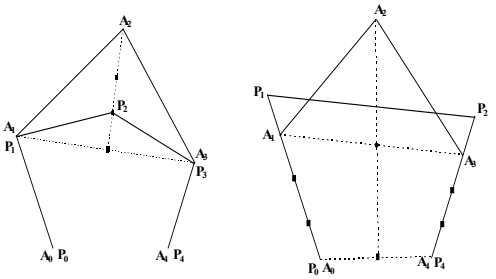

Рис.11. Пересчет управляющих точек кривой Безье (4-го и 3-го порядков) в управляющие точки эквивалентной ей лекальной кривой

Приведенные матрицы соответствуют лекальному генератору с тремя коническими интерполяторами. Следующая схема, состоящая из семи конических интерполяторов и восьми линейных интерполяторов способна растрировать кривые Безье до восьмого порядка включительно. Методика построения матриц пересчета вершин кривой Безье в вершины эквивалентной ей лекальной кривой та же, что и для предшествующей вычислительной структуры. Приведем матрицы пересчета $G_{8,n}$ для кривых Безье порядка n=8,7,6,5.

$$G_{8,8} = \frac{1}{3}\begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 18 & -52 & 53 & -24 & 4 & 0 \\ 0 & 18 & -129 & 312 & -318 & 144 & -24 & 0 \\ 0 & 32 & -192 & 424 & -416 & 186 & -31 & 0 \\ 0 & -31 & 186 & -416 & 424 & -192 & 32 & 0 \\ 0 & -24 & 144 & -318 & 312 & -129 & 18 & 0 \\ 0 & 4 & -24 & 53 & -52 & 18 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix};$$

$$G_{8,7} = \frac{1}{24}\begin{pmatrix} 21 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 60 & -66 & 4 & 39 & -24 & 4 \\ 18 & -150 & 291 & -24 & -234 & 144 & -24 \\ 32 & -192 & 312 & 32 & -318 & 186 & -31 \\ -31 & 186 & -318 & 32 & 312 & -192 & 32 \\ -24 & 144 & -234 & -24 & 291 & -150 & 18 \\ 4 & -24 & 39 & 4 & -66 & 60 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 21 \end{pmatrix};$$

$$G_{8,6} = \frac{1}{4}\begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & -6 & 4 & -1 & 0 \\ -1 & -4 & 26 & -24 & 6 & 0 \\ 0 & -8 & 32 & -28 & 7 & 0 \\ 0 & 7 & -28 & 32 & -8 & 0 \\ 0 & 6 & -24 & 26 & -4 & -1 \\ 0 & -1 & 4 & -6 & 4 & 2 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix};$$

$$G_{8,5} = \frac{1}{24}\begin{pmatrix} 15 & 0 & 0 & 0 & 0 \\ 14 & 4 & -6 & 4 & -1 \\ -9 & 36 & 6 & -24 & 6 \\ -8 & 32 & 12 & -28 & 7 \\ 7 & -28 & 12 & 32 & -8 \\ 6 & -24 & 6 & 36 & -9 \\ -1 & 4 & -6 & 4 & 14 \\ 0 & 0 & 0 & 0 & 15 \end{pmatrix}.$$

Отметим, что многоугольники, задающие кривые Безье и эквивалентные им лекальные им кривые,

различны. Быстродействие генератора лекальных кривых, как было показано выше, определяется вычислительной структурой (уровнем ее организации) и величиной R, которая вычисляется над набором сторон разомкнутого многоугольника <$\mathbf{A_k}$>. Рассчитаем R_A над набором ребер <$\mathbf{A_k}$>, и R_P над набором <$\mathbf{P_k}$>.

$$R_A = \max\{\|\vec{A}_k\|\};\ R_P = \max\{\|\vec{P}_k\|\}; \quad (13)$$

где: $\|\vec{A}_k\| = \max\{|X_k - X_{k-1}|, |Y_k - Y_{k-1}|, |Z_k - Z_{k-1}|, ...\}$;

$\|\vec{P}_k\|$ - определяется аналогично $\|\vec{A}_k\|$.

Нетрудно заметить, что $R_A > R_P$, то есть, генерация кривой Безье будет выполняться медленнее, чем генерация лекальной кривой, вписанной в тот же многоугольник <$\mathbf{P_k}$>. Найдя отношение R_A / R_P, можно оценить коэффициент замедления генерации кривой Безье по сравнению с генерацией лекальной кривой, вписанной в один и тот же многоугольник.

Пусть <$\mathbf{P_k}$> - ребра кривой Безье, тогда $\vec{A}_k = \sum_i g_{ki}\vec{P}_i$ - ребра эквивалентной ей лекальной кривой. Получаем оценку

$$\frac{R_A}{R_P} \le \max_k\{\sum_i |g_{ki}|\}. \quad (14)$$

Матрицы g_{ki} приведены выше. На основе этих результатов составлена следующая таблица

Порядок кривой Безье	2	3	4	5	6	7
R_A / R_P	1	1.25	3	3.6	18.7	48.9

Из этой таблицы видно, что генерация кривых Безье выше 5-го порядка с помощью генератора лекальных кривых нецелесообразна. Таким образом, хотя генератор лекальных кривых является достаточно быстродействующим устройством, его возможности в плане генерации кривых Безье высокого порядка ограничены. Отметим, что уменьшение быстродействия никак не влияет на устойчивость работы генератора, быстро или медленно, но кривая будет сгенерирована все равно.

6. Представление равномерного B-сплайна набором кривых Безье.

Описание кривых сплайнами имеет много качеств, привлекательных для использования в системах интерактивного геометрического проектирования. Разработано множество алгоритмов и программ для представления кривых B-сплайнами. Нашей целью является показать, что растрирование сплайна может быть успешно выполнено с помощью лекального интерполятора.

Из теории сплайнов известно, что B-сплайн можно представить набором сопряженных сегментов, задаваемых следующим уравнением

$$\vec{Z}_m^{[J]}(t) = \sum_{k=0}^{m} \vec{Z}_{m-k+J} B_{k,m}(t), \quad (15)$$

где: $\vec{Z}_m^{[J]}(t)$ - J- тый сегмент сплайн функции, являющийся степенным полиномом степени m. J- индекс сегмента сплайн функции. $t \in [0,1]$ - скалярный параметр. В данном разделе мы рассмотрим только равномерные сплайны, для которых

$$B_{k,m}(t) = \frac{1}{m!}\sum_{i=0}^{k}(-1)^i C_{m+1}^i (t+k-i)^m, \quad (16)$$

где: $C_n^k = \frac{n!}{k!(n-k)!}$ - биномиальный коэффициент.

Представим сегмент $\vec{Z}_m^{[J]}(t)$ в виде степенного полинома

$$\vec{Z}_m^{[J]}(t) = \frac{1}{m!}\sum_{k=0}^{m} C_m^k t^k \sum_{i=0}^{m} \vec{Z}_{i+J} \sum_{j=0}^{m-i} (-1)^j C_{m+1}^j (m-i-j)^{m-k}. \quad (17)$$

Кривая Безье также может быть представлена в форме степенного полинома

$$\vec{P}_m(t) = \sum_{k=0}^{m} \vec{P}_k (1-t)^{m-k} t^k = \sum_{k=0}^{m} (-1)^k C_m^k t^k \sum_{i=0}^{k} (-1)^i C_k^i \vec{P}_i. \quad (18)$$

Выражения $\vec{Z}_m^{[J]}(t)$ и $\vec{P}_m^{[J]}(t)$ описывают одну и ту же функцию, откуда следует их эквивалентность. Приравнивая коэффициенты при одинаковых степенях t, находим связь между управляющими точками сплайна и управляющими точками кривой Безье, представляющей J - тый сегмент сплайна.

$$\sum_{i=0}^{k} (-1)^i C_k^i \vec{P}_i^{[J]} = \frac{(-1)^k}{m!}\sum_{i=0}^{m} \vec{Z}_{i+J} \sum_{j=0}^{m-i} (-1)^j C_{m+1}^j (m-i-j)^{m-k}. \quad (19)$$

Из комбинаторной математики известна справедливость следующих соотношений

$$b_k = \sum_{i=0}^{k}(-1)^i C_k^i a_i;\ \ a_k = \sum_{i=0}^{k}(-1)^i C_k^i b_i; \quad (20)$$

Из этих соотношений находим

$$\vec{P}_k^{[J]} = \frac{1}{m!}\sum_{l=0}^{k} C_k^l \sum_{i=0}^{m} \vec{Z}_{i+J} \sum_{j=0}^{m-i} (-1)^j C_{m+1}^j (n-j-j)^{m-i}; \quad (21)$$

$k \in \{0,...,m\}$.

Таким образом, мы нашли формулу для пересчета управляющих точек сплайна в управляющие точки его сегментов, представленных в форме Безье. Данная формула является математическим решением, но

она не подходит для быстрых вычислений. Представим решение в матричной форме

$$(\vec{P}_0^{[J]},...,\vec{P}_m^{[J]})^T = \frac{1}{m!} G_m \cdot (\vec{Z}_J,...,\vec{Z}_{m+J})^T, \qquad (22)$$

где: "T" - операция транспонирования вектора. G_m - матрица размерности $m \times m$.

$$g_{k,i}^{[m]} = \sum_{i=0}^{k} C_k^i \sum_{j=0}^{m-i} (-1)^j C_{m+1}^j (m-i-j)^{m-i}; \qquad (23)$$

$k,i \in [0,m]$.

Принимая во внимание соотношение для биномиальных коэффициентов $C_m^k = C_{m-1}^k + C_{m-1}^{k-1}$, представим $g_{k,i}^{[m]}$ в рекуррентной форме

$$g_{k,i}^{[m]} = g_{k-1}^{[m]} + g_{k-1,i-1}^{[m-1]} - g_{k-1,i}^{[m-1]}, \qquad (24)$$

где: $k \in [1,m]; \quad g_{0,0}^{[m]} = 1;$

$$g_{0.i}^{[m]} = \sum_{j=0}^{m-1} g_{j,i}^{[m-1]}; \qquad m! = \sum_{i=0}^{m} g_{0,i}^{[m]}. \qquad (25)$$

Теперь мы можем предложить простой алгоритм (С-код) для вычисления матрицы G_m любого порядка.

```
void Gm(int M)
{
   static long g[13][13], g1[13][13];  // 13 - максимальный
                        // порядок для 32 разрядных целых
   int i=0, j=0, F=0;
   g[0][0]  = g1[0][0] = g1[1][1] =1;
   g1[0][1] = g1[1][0] = 0;

   for(int m=2; m <= M;  m++)
   {
         for(F=i=0; i < m; i++) {
            for(F=j=0; j < m; j++)
               F += g1[j][i];
            g[0][i] = F;
         }

         for(g[0][m]=0, F=i=1; i<m; i++) {
            for(j=i, F +=g[0][i]; j< m; j++)
               g[i][j] = g[i-1][j] + g1[i-1][j-1] - g1[i-1][j];
            g[i][m] = 0;
         }

         for(i=1, g[m][m]=1; i <= m; i++)
            for(j=0; j < i; j++)
               g[i][j] = g[m-i][m-j];

         for(i=0; i <= m; i++)
            for(j=0; j<=m; j++)
               g1[i][j]=g[i][j];
   }

   for(i=0,  printf("m! = %ld", F); i<m; i++)
      for(j=0,  printf("\r\n"); j<m; j++)
         printf("%8d", g[i][j]);
}
```

Пользуясь данной программой, рассчитаем матрицы пересчета для равномерных сплайнов 3,..,7 порядков.

$$G_3 = \frac{1}{3!}\begin{pmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{pmatrix}; \qquad G_4 = \frac{1}{4!}\begin{pmatrix} 1 & 11 & 11 & 1 & 0 \\ 0 & 8 & 14 & 2 & 0 \\ 0 & 4 & 16 & 4 & 0 \\ 0 & 2 & 14 & 8 & 0 \\ 0 & 1 & 11 & 11 & 1 \end{pmatrix};$$

$$G_5 = \frac{1}{5!}\begin{pmatrix} 1 & 26 & 66 & 26 & 1 & 0 \\ 0 & 16 & 66 & 36 & 2 & 0 \\ 0 & 8 & 60 & 48 & 4 & 0 \\ 0 & 4 & 48 & 60 & 8 & 0 \\ 0 & 2 & 36 & 66 & 16 & 0 \\ 0 & 1 & 26 & 66 & 26 & 1 \end{pmatrix};$$

$$G_6 = \frac{1}{6!}\begin{pmatrix} 1 & 57 & 302 & 302 & 57 & 1 & 0 \\ 0 & 32 & 262 & 342 & 82 & 2 & 0 \\ 0 & 16 & 212 & 372 & 116 & 4 & 0 \\ 0 & 8 & 160 & 384 & 160 & 8 & 0 \\ 0 & 4 & 116 & 372 & 212 & 16 & 0 \\ 0 & 2 & 82 & 342 & 262 & 32 & 0 \\ 0 & 1 & 57 & 302 & 302 & 57 & 1 \end{pmatrix};$$

$$G_7 = \frac{1}{7!}\begin{pmatrix} 1 & 120 & 1191 & 2416 & 1191 & 120 & 1 & 0 \\ 0 & 64 & 946 & 2416 & 1436 & 176 & 2 & 0 \\ 0 & 32 & 716 & 2336 & 1696 & 256 & 4 & 0 \\ 0 & 16 & 520 & 2176 & 1952 & 368 & 8 & 0 \\ 0 & 8 & 368 & 1952 & 2176 & 520 & 16 & 0 \\ 0 & 4 & 256 & 1696 & 2336 & 716 & 32 & 0 \\ 0 & 2 & 176 & 1436 & 2416 & 946 & 64 & 0 \\ 0 & 1 & 120 & 1191 & 2416 & 1191 & 120 & 1 \end{pmatrix}.$$

Таким образом, мы показали, что сплайн можно представить набором сопряженных кривых Безье, которые, в свою очередь, могут быть растрируемы с помощью генератора лекальных кривых.

Для сплайнов третьей и четвертой степеней можно получить наглядную иллюстрацию полученных результатов (<Z> - опорные точки сплайна, <P> - опорные точки кривой Безье (сегмента сплайна)). На приведенной ниже иллюстрации левый рисунок соответствует сплайнам третьего порядка, правый - сплайнам четвертого порядка.

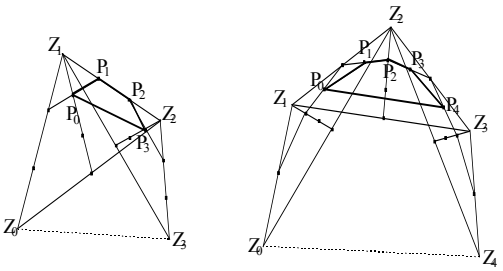

Рис. 12. Связь между управляющими точками сплайнов и соответствующих им кривых Безье.

Приведем ниже пример представления равномерного сплайна третьего порядка набором сопряженных кривых Безье третьего порядка. Отметим, что первая и последняя опорные точки сплайна являются "висящими", их задание неочевидно, однако, если кривая замкнута, то эта проблема снимается. Отметим, что построение сплайнов (интерполяционных или сглаживающих), строго говоря, не является нашей задачей, наша цель визуализировать заданный сплайн доступными нам средствами, каковыми являются, например, кривые Безье.

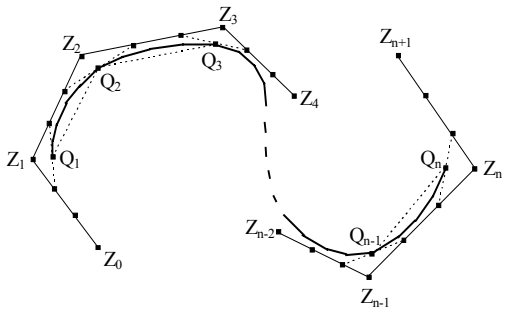

Рис. 13. Иллюстрация представления равномерного кубического сплайна набором сопряженных кривых Безье.

Литература.

[1]. Чириков. С.В., Палташев Т.Т. *Конические дуги - их свойства и метод растрирования.* ГРАФИКОН 2000.

[2]. Bresenham J.E. *A Linear Algorithm for Incremental Digital Display of Circular Arcs.* // Communication of the ACM. - 1977. - V. 20(2). - P. 100-106.

[3]. Чириков С.В. *Целочисленный алгоритм аппроксимации лекальной кривой равномерным сплайном и представление его в виде последовательности сопряженных кривых Безье.* // Вопросы радиоэлекроники. Серия ОВР. - 1991. - Вып. 3. с.85-92.

[4]. Чириков С.В. *Целочисленный алгоритм поточечной генерации геометрически заданных конических дуг - основа генератора графических примитивов для устройств отображения растрового типа.* // Вопросы радиоэлектроники. Серия ОВР. - 1991. - Вып. 7.

[5]. Чириков С.В. *Алгоритмы и структуры генерации лекальных кривых в растровых графических видеотерминалах:* // Дис... канд. техн. наук 11.03.92 - защищена. С-Перербург 180с..

[6]. Chirikov S.V., Paltashev T.T. *Integer algorithm for L-curve rasterization and application of L-curve for Bezier curve representation.* // GRAFICON 93 (Conf.) Russia.

[7]. Chirikov S.V., Paltashev T.T. *Curve Rasterisation Algorithm Design: An Unifying Approach.* // EuroGraphics 94 (Conf.) Norway.

[8]. Chirikov S.V., Paltashev T.T., P.A.Balabko. *New Method of Parametric Curve Rasterization and its Application for Fast Bezier Patch Shading.* // GRAFICON 95 (Conf.) Russia

[9]. Chirikov S.V., Paltashev T.T. , P.A.Balabko. *New Method of Parametric Curve Rasterization and its Application for Fast Bezier Patch Shading.* // WSCG'97 (Conf.) Czech Republic.

[10]. Bezier P., *Numerical Control: Mathematics and Applications*, // Wiley, Chihester, UK, 1972.

[11]. E. Catmull, *A Subdivision Algorithm for Display of Curved Surfaces,* // PhD dissertation, Univ. Of Utah, UTEC-Csc-74-133, December 1974.

[12]. Fuhua Cheng, Kuen-Rong Hsieh, Re-Ron Huang, and YehHao Chin, *Bezier Curve Generator: A Hardware Approach to Curve Generation*, // Proceedings of the Second International Symposium of VLSI Technology, Systems and Applications, May 1985, Taipei, Taiwan, pp.278-281.

[13]. Tony DeRose, Thomas Holman *The Triangle: A Multiprocessor Architecture for Fast Curve and Surface Generation*, // Technical Report 87-08-07,August 1987, Department of Computer Science, FR-35, University of Washington, Seattle, WA, 98195.

[14]. S-L. Lien, M.Shantz, and V.Pratt. *Adaptive Forward Differencing for Rendering Curves and Surfaces,* // Computer Graphics, 21(4), July 1987, pp.111-118.

[15]. Chang S.L., Shantz M.,and Rochetti R., *Rendering Cubic Curves and Surfaces with Integer Adaptive Forward Differencing*, // Computer Graphics, 1989, Vol.23, No.3, pp.157-166.

[16]. Sheue-Ling Lien, Michael Shantz, Jeral Evans, Serdar Egrene, Susan Carrie. *Method and Apparatus for Rendering Vectors Using Bresenham Parameters*, // United States Patent # 4,855,935, August 8, 1989.

[19]. Ari Rappoport, *Rendering curves and surfaces with hybrid subdivision and forward differencing* // ACM Transactions on Graphics, 10(4), pp. 323-341, October 1991.

[20]. R.Victor Klassen. *Integer Forward Differencing of Cubic Polynomials: Analysis and Algorithms*, // ACM Transactions on Graphics, 10(2), pp.152-181, April 1991.

[21]. **R. Victor Klassen.** *Exact Integer Hybrid Subdivision and Forward Differencing of Cubics*, // ACM Transactions on Graphics, 13(3), pp. 240-255, July 1994.