УДК 004.5

ГРАФИЧЕСКОЕ ПРИЛОЖЕНИЕ ДЛЯ СОЗДАНИЯ NURBS-КРИВЫХ

Филипенко М.И, Абрамова О.П.
Донецкий национальный университет

Геометрические параметры криволинейных поверхностей часто являются определяющими для проектных характеристик объекта. К таким объектам относятся изделия с динамическими обводами (или функциональными поверхностями) в авиастроении, автомобилестроении, судостроении, энергетическом и химическом машиностроении, сельхозмашиностроении. Актуальной задачей является выработка формальных критериев визуальной оценки геометрической определенности поверхности, моделируемой на сети кривых, на проекциях комплексного чертежа. К качеству геометрических характеристик функциональных поверхностей предъявляются особые требования. Методы моделирования должны соответствовать требуемому качеству функциональной поверхности изделия. Благодаря своим свойствам фактическим стандартом представления кривых и поверхностей в настоящее время являются NURBS (Non-Uniform Rational B-Splines – неравномерные рациональные B-сплайны) кривые и поверхности.

Данная работа посвящена разработке графического приложения для моделирования NURBS кривых [1]. Кривые на основе NURBS широко используются в конструировании и моделировании. Они известны своими «гибкими» свойствами:

- их математическое описание генерируется без изначального знания формы кривой, лишь по некоторому набору контрольных точек;
- изменяя величину весовых коэффициентов для контрольных точек, можно более точно управлять формой кривой;
- если над кривой необходимо провести перспективные преобразования, то достаточно их применить только к контрольным точкам, а не к каждой точке кривой в отдельности.

NURBS кривые задаются следующими формулами [1, 2]

$$P(t)=\sum_{i=0}^{n+1} P_i\, R_i(t), \qquad R_i(t)=\frac{w_i N_{i,k}(t)}{\sum_{i=0}^{n+1} w_i N_{i,k}(t)}, \tag{1}$$

где

$$N_{i,k}(t)=\frac{t-x_i}{x_{i+k-1}-x_i}N_{i,k-1}(t)+\frac{x_{i+k}-t}{x_{i+k}-x_{i+1}}N_{i+1,k-1}(t);\quad k=2,3... \tag{2}$$

$$N_{i,1}(t)=\begin{cases}1, & \textit{если} \quad x_i \le t \le x_{i+1};\\ 0, & \textit{иначе.}\end{cases}$$

Здесь t – параметр; P_i – контрольные точки, формирующие контрольный полигон; w_i – вес контрольной точки (однородная координата); n – количество контрольных точек; $N_{i,k}(t)$ – базисная функция B-сплайна; k – порядок базисной функции B-сплайна; x_i – элементы узлового вектора, удовлетворяющего соотношению

$x_i \le x_{i+1}$ и содержащего $k+n$ компонент. Функция $P(t)$ из (1) выражает параметрическое задание NURBS кривой. Пример простой NURBS кривой приведен на рис. 1.

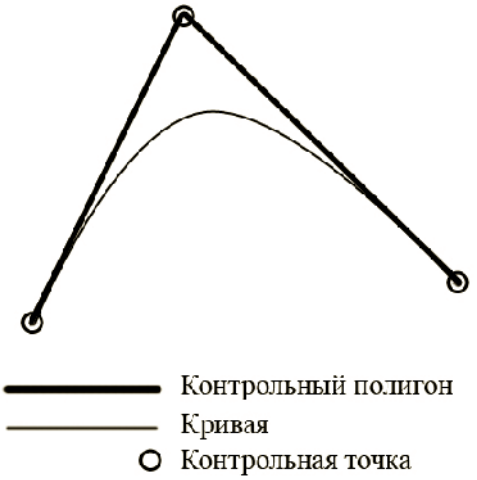

Рисунок 1 – Простая NURBS кривая

Характерной особенностью NURBS кривых является возможность задания веса (однородной координаты) для любой контрольной точки. Кривая больше «притягивается» к контрольной точке с наибольшим весом, а если веса всех контрольных точек одинаковые, то данная NURBS кривая вырождается в B-сплайн кривую.

NURBS кривые являются аппроксимирующими кривыми, которые лишь приближаются к значениям контрольных точек. Тем не менее, они всегда интерполируют первую и последнюю точки контрольного полигона. Более того, задав сравнительно большой вес контрольной точки, можно ее интерполировать.

Стоит отметить, что аппроксимация кривой в случае NURBS является не недостатком, а скорее методом конструирования и управления кривой. Кривая попадает в выпуклую оболочку контрольного полигона, за рамки которого она не выходит. На самом деле, это свойство данной кривой даже сильнее: кривая (при любом значении параметра t) лежит внутри выпуклой оболочки, образованной m последовательными контрольными точками, где m – порядок кривой. Таким образом, чем больше порядок кривой m, тем больше ее можно «ограничить». На практике чаще всего используются кривые четвертого порядка (кубические кривые).

При конструировании NURBS кривой также важным аспектом является то, что она не может совершать больше «колебаний», чем ее контрольный полигон.

Для разработки графического приложения выбрана среда программирования Visual Studio .NET [3], язык программирования – C++. В работе использована графическая библиотека OpenGL [2].

Задача создания приложения разбита на несколько этапов:

3. анализ вычислительного алгоритма NURBS и рендеринга кривой, в результате которого выделяем графические примитивы для моделирования кривой;
4. разработка библиотеки взаимосвязанных классов отдельных графических примитивов;

5. разработка пользовательского интерфейса с помощью инструментария Visual Studio .NET, обеспечение взаимосвязи интерфейса и разработанной библиотеки классов.

Остановимся подробнее на каждом из перечисленных этапов.

На этапе анализа вычислительного алгоритма NURBS сделаны следующие выводы. NURBS кривая задается узловым вектором и набором контрольных точек, которые формируют контрольный полигон. Узловой вектор формируется автоматически, в зависимости от текущего количества контрольных точек в контрольном полигоне. Контрольные точки задаются в однородных координатах, то есть к абсциссе и ординате каждой точки добавляется третья координата – так называемый весовой коэффициент (обязательно положительный и не равный нулю). В отличие от контрольных точек, точки самой кривой, рассчитываемые по формулам (1), (2), не обязаны быть представлены в однородных координатах. Таким образом, выделены следующие графические примитивы:

- точка;
- точка в однородных координатах;
- контрольный полигон (состоит из точек в однородных координатах);
- кривая (состоит из точек и узлового вектора).

При выводе кривой на экран (рендеринге), учтено, что она, на самом деле, будет заменена ломаной линией путем соединения точек, рассчитанных по формулам (1), (2). Реальное количество этих точек должно быть достаточным, чтобы имитировать гладкую кривую.

На втором этапе – этапе разработки библиотеки классов поставлена цель раскрыть сущность каждого графического примитива и представить его в виде класса со своим набором членов данных и функций.

Каждый примитив, будь то точка или кривая, выводится на экран. Это означает, что у всех примитивов должна быть одна общая функция – функция вывода на экран (прорисовки). Таким образом, применив объектно-ориентированный подход, приходим к выводу о необходимости использования механизма наследования [4]. Это означает, что все классы графических примитивов будут наследниками одного абстрактного класса, который будет иметь прототип функции вывода на экран. Тогда все наследники этого абстрактного класса могут быть объединены в единый массив указателей, что позволить управлять ими одновременно при прорисовке.

Класс «точка» содержит две свои координаты (абсциссу и ординату) в качестве членов данных. Функции данного класса представляют собой функции доступа к членам данных, то есть функции, позволяющие получить или изменить значения членов данных.

Класс «точка в однородных координатах» абсолютно аналогичен классу «точка», за исключением того, что к нему добавляется еще один член данных – однородная координата (весовой коэффициент) и соответствующие функции доступа к членам данных. Поэтому рассматриваемый класс является наследником класса «точка».

Класс «контрольный полигон» включает в качестве членов данных массив точек в однородных координатах. Сам по себе, контрольный полигон является основным средством управления кривой: добавляя, удаляя или перемещая конкретные контрольные точки, можно изменять форму кривой. Исходя из этого, в классе «контрольный полигон», кроме функций доступа к членам данных, реализованы функции добавления, удаления и перемещения произвольной точки из массива точек в однородных координатах. Под перемещением также понимается изменение весового коэффициента контрольной точки. При добавлении новой контрольной точки ей присваивается стандартное значение весового коэффициента – единица.

Разработанный класс «кривая» в качестве членов данных содержит:

- массив точек кривой;
- число, характеризующее количество точек кривой;
- число, характеризующее порядок кривой;
- узловой вектор (целочисленный массив);
- контрольный полигон.

Создавать отдельный класс для узлового вектора нет необходимости, несмотря на то, что, варьируя его значения, можно управлять поведением кривой. Тех же результатов можно добиться перемещая контрольные точки, что безусловно является более интуитивно-понятным средством управления.

Таким образом, для класса «кривая» была реализована функция для автоматического генерирования контрольного полигона. Кроме того, были реализованы функции доступа ко всем членам данных и функции для расчета точек кривой по формулам (1), (2).

На рис. 2 приведена схема наследования разработанных классов.

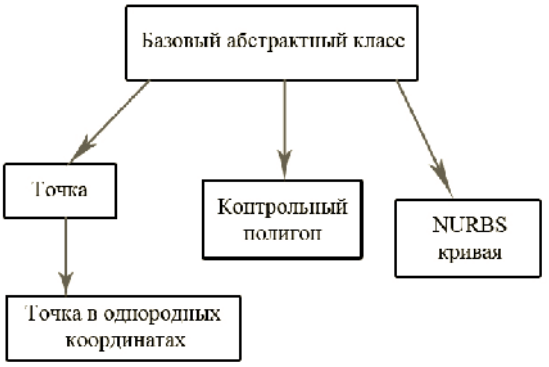

Рисунок 2 – Схема наследования разработанных классов

На третьем этапе – этапе разработки пользовательского интерфейса выбрана среда программирования Microsoft Visual Studio .NET, тип проекта – MFC Application. На данном этапе подключаются все необходимые библиотеки, в том числе разработанная библиотека графических примитивов и библиотека OpenGL.

Стоить отметить, что пользователь получает необходимую форму кривой за счет изменения контрольного полигона. Поэтому хранить кривую как член данных не имеет смысла – она каждый раз перерисовывается при обновлении экрана. Контрольный полигон, напротив, является членом данных, так как данные о его точках должны передаваться при обращении к обработчикам сообщений мыши и пунктов меню. Обработчики сообщений, после вызова соответствующих функций-членов контрольного полигона, вызывают функцию перерисовки рабочей области.

Таким образом, проведен анализ особенностей вычислительного алгоритма (1), (2) и возможностей Visual Studio .NET. В итоге, разработано графическое приложение, позволяющее создавать NURBS кривые, редактировать их и сохранять. Пользователь с помощью левой кнопки мыши имеет возможность задать контрольные точки в клиентской области приложения. По ходу задания контрольных точек, прорисовывается текущая кривая. На экран также выводится контрольный полигон, но эту функцию можно отключить. В главном меню доступны инструменты для перемещения и удаления контрольных точек. Указав контрольную точку с помощью правой кнопки мыши, можно задать вес этой контрольной точки в открывшемся

диалоговом окне. Посредством команды главного меню реализована функция сброса весов всех контрольных точек к стандартному значению.

На рис. 3, 4 представлен результат работы графического приложения. В клиентской области приложения отображены кривая и контрольный полигон. На рис. 3 кривая моделируется по набору из шести контрольных точек с одинаковыми весами. Порядок задания контрольных точек (левой кнопкой мыши) определяет форму моделируемой кривой. Для одной из контрольных точек (пятой) было увеличено значение веса, в результате чего кривая стала больше притягиваться к этой точке (рис. 4). Приложение позволяет сохранять построенные кривые и использовать их в более сложных объектах и анимации.

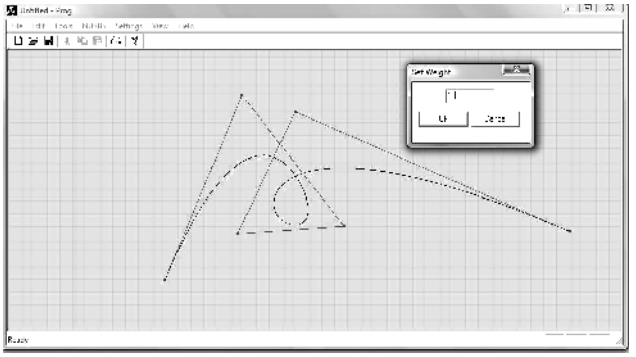

Рисунок 3 – Моделирование кривой по контрольным точкам с одинаковыми весами

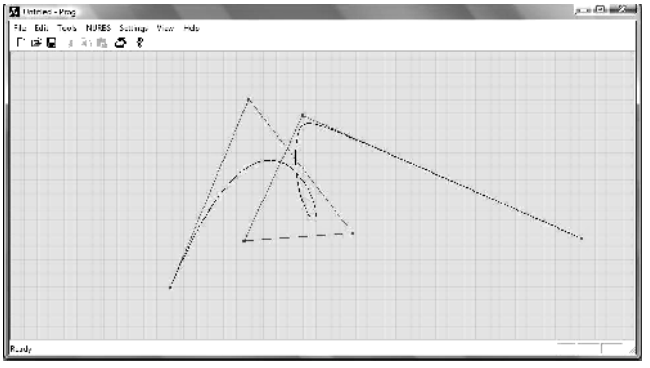

Рисунок 4 – Весовой коэффициент изменен для одной из контрольных точек

Литература

[1] Роджерс Д. Математические основы машинной графики / Роджерс Д., Адамс Дж. – Москва: «Мир», 2001. – 604 с.

[2] Хилл Ф. OpenGL. Программирование компьютерной графики. Для профессионалов. / Хилл Ф. – СПб.: Питер, 2002. – 1088 с.

[3] Понамарев В.А. Программирование на C++/C# в Visual Studio .NET 2003. – СПб.: БХВ-Петербург, 2004. – 352 с.

[4] Страуструп Б. Язык программирования C++ / Страуструп Б. – М.; СПб. : «Издательство БИНОМ» – «Невский диалект», 2001. – 1099 с.