ВИЗУАЛИЗАЦИЯ КРИВЫХ ПРИ ПОМОЩИ ПРОГРАММИРУЕМОГО ГРАФИЧЕСКОГО КОНВЕЙЕРА

А. А. Козик

С каждым годом индустрия разработки игр развивается все более быстрыми темпами, захватывая большее количество разнообразных платформ и средств распространения. Требования потребителей растут еще быстрее. Одним из наиболее значимых критериев в пользу покупки игры для пользователя является качество графики, зрелищность и реалистичность. По этой причине разработчикам приходится постоянно искать способы удовлетворить желания игроков.

Первым шагом к реалистичности был переход игр из 2D пространства к 3D. Подобно реальному миру, в трехмерных играх игровой мир также состоит из трехмерных объектов. Поверхность каждого объекта строится при помощи треугольников, которые, в свою очередь, задаются тремя вершинами, расположенными в пространстве.

Для отображения поверхности, ограниченной отрезками, соединяющими три вершины, используются так называемые материалы. Материал задает параметры отображения поверхности, цвет, характер и наличие бликов, способ затенения. Цвет материала может быть сплошным, задаваться процедурно, либо использовать растровую текстуру. Таким образом, проведя аналогию с типами графических изображений, можно сказать, что цвет материала может быть процедурным (векторным), либо растровым. Как и в графике растровые и процедурные текстуры имеют свои достоинства и недостатки. Наиболее существенным недостатком процедурных текстур является сложность передачи реалистичных изображений, а также изображений с большим количеством деталей. Из чего следует, что процедурные текстуры могут быть применены только при решении некоторых специфических задач. Одной из таких задач является визуальное представление кривых Безье.

Итак, задачей является представление квадратичной кривой Безье в пространственном треугольнике, как показано на Рис. 1. Сложность представление кривой Безье заключается в вычислительной сложности ее параметрического представления, а именно:

$$B(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2, \quad (1)$$

Так как все расчеты должны быть выполнены на этапе фрагментного шейдера, это приведет к существенной нагрузке и значительному снижению производительности приложения.

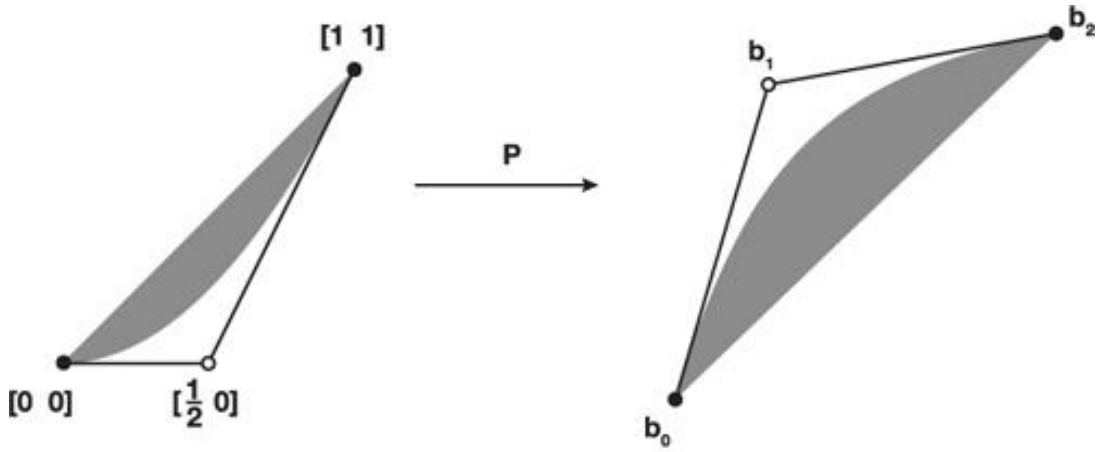

Рис. 1. Отображение кривой Безье методом Блина-Лупа

Рассмотрим метод Блина-Лупа. Рассмотрим треугольник (Рис. 1), вершины b_0, b_1, b_2 будут являться опорными точками квадратичной кривой Безье. Предположим, что текстурными координатами этих вершин являются (0; 0), (0,5; 0) и (1; 1) соответственно. Покажем, что точки текстурного пространства, удовлетворяющие равенству

$$u^2 - v = 0, \tag{2}$$

где *u, v* – соответствующие текстурные координаты точки, будут являться визуальным отображением квадратичной кривой Безье.

Докажем, что мы можем спроецировать квадратичную кривую на текстурное пространство треугольника, представив ее в неявном виде:

$$f(u,\ v) = u^2 - v. \tag{3}$$

Квадратичную кривую представим в следующем виде:

$$C(t) = tC, \tag{4}$$

где $\boldsymbol{t} = [1 \quad t \quad t^2]$, $\boldsymbol{C} = \begin{bmatrix} x_0 & y_0 & w_0 \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix}$.

Сравним с функцией

$$F(t) = \boldsymbol{tF} = [t \quad t^2 \quad 1], \tag{5}$$

где $\boldsymbol{F} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$.

Так как F удовлетворяет $u(t) = t$ и $v(t) = t^2$, то выполняется желаемое неявное равенство

$$f(u,\ v) = u^2 - v = 0. \tag{6}$$

Таким образом, нам остается найти преобразование $\boldsymbol{\Psi}$, такое что:

$$\boldsymbol{C} = \boldsymbol{F}\,\boldsymbol{\Psi}^{-1}. \quad (7)$$

Очевидно, что $\boldsymbol{\Psi}^{-1} = \boldsymbol{F}^{-1}\boldsymbol{C}$. Таким образом, мы можем спроецировать квадратичную кривую на текстурное пространство треугольника, представив ее в неявном виде (3).

$$C(t) = \boldsymbol{t}(\boldsymbol{F}\boldsymbol{\Psi}^{-1}). \quad (8)$$

Используя в качестве преобразования базисную матрицу Безье для квадратичной функции, мы получим необходимые текстурные координаты для вершин треугольника, а именно (0; 0), (0,5; 0) и (1; 1), что и требовалось доказать.

Подытожим общий алгоритм визуализации квадратичной кривой Безье. Присваиваем вершинам треугольника указанные текстурные координаты и передаем примитив фрагментному шейдеру. Для получения изображения сходного с Рис. 1 воспользуемся следующим алгоритмом (в прямом или инвертированном виде):

$f(u, v) > 0$ – не закрашенная область;

$f(u, v) \leq 0$ – закрашенная область.

При растеризации GPU автоматически произведет все необходимые преобразования, что позволит добиться «идеального» масштабирования и отсутствия «артефактов» семплирования.

Данный метод позволяет использовать все преимущества процедурного текстурирования, являясь очень легким в вычислительном смысле в особенности в ракурсе использования преимуществ современных GPU (вычислительная мощность и многопоточная параллельная обработка). Кроме того, он позволяет добиться «идеального» масштабирования, а также исключить артефакты при перспективном отображении. Минимизирует затраты на этапе альфа-тестирования. Это позволяет, в первую очередь, широко применять его для отображения символов и текстов в трехмерном графике реального времени.

Литература

1. *Akenine-Moller T.* Real-time rendering. AK Peters, 2008.
2. *Biswaz S.* Bezier and Splines in Image Processing and Machine Vision. Springer, 2008.
3. *Flew T.* Games: Technology, Industry, Culture. Oxford University Press, 2005.
4. *Loop C., Blinn J.* Rendering Vector Art on the GPU // GPU Gems 3. Addison Wesley, 2007. Ch. 25. P. 543–562.
5. *Loop C., Blinn J.* Resolution Independent Curve Rendering using Programmable Graphics Hardware // Proc. of Siggraph. Los Angeles, 2005. P. 1000–1010.