АЛГОРИТМЫ ПРОГРАММНОГО РЕНДЕРИНГА ТРЕХМЕРНОЙ ГРАФИКИ ДЛЯ ЗАДАЧ МЕДИЦИНСКОЙ ВИЗУАЛИЗАЦИИ

Ю.В. Книгавко, О.Г. Аврунин
Харьковский национальный университет радиоэлектроники
Кафедра БМЭ, ХНУРЭ , просп. Ленина, 14, г.Харьков, 61166, Украина
Тел.: 702-13-64, e-mail: yukni@mail.ru, gavrun@list.ru

***Annotation* –The steps of the development of software graphics library are described in this paper. The advantages and disadvantages of the software implementation of visualization system are present.**

***Key words* – visualization, rendering, ray tracing, spiral computer tomography, index buffer.**

ВВЕДЕНИЕ

Медицинская визуализация является активно развивающейся отраслью компьютерной графики в наши дни. Современные интроскопические средства постоянно совершенствуются, позволяя получать все более и более точные изображения исследуемых органов. Сопряжение возможностей диагностических систем с вычислительной мощью современных центральных и графических процессоров открывает возможность для разработки сложных интерактивных систем медицинской визуализации [1].

Широкое распространение трехмерной компьютерной графики в сфере развлекательных технологий (игры, симуляторы) стимулировало быстрое развитие аппаратных средств ускорения рендеринга и библиотек по работе с ними. Эволюция графических процессоров компьютера привела к тому, что для рядового пользователя стало возможным приобрести вычислительное устройство, обладающее пиковой производительностью порядка нескольких терафлоп в секунду. Прогресс средств диагностики и производительность вычислительных систем диктует повышение требований к качеству (и соответственно сложности) алгоритмов обработки интроскопических данных и визуализации результатов такой обработки в масштабе реального времени [2].

АСПЕКТЫ РАЗРАБОТКИ АЛГОРИТМОВ ПРОГРАММНОЙ ВИЗУАЛИЗАЦИИ

Вычислительные системы современных томографов, как правило, содержат в своем составе высокопроизводительный многоядерный процессор. Его основной задачей является выполнение реконструкции срезов по проекциям. Возможное отсутствие производительных графических процессоров общего назначения на борту этих ЭВМ и установленные специализированные операционные системы не позволяют использовать для визуализации популярные GAPI (Graphics application program interface). В связи с этим, актуальной становится задача разработки собственной графической библиотеки, выполняющей рендеринг томографических данных и результатов их обработки, полностью на центральном процессоре.

К преимуществам такой библиотеки можно отнести:

- хорошая переносимость (кросплатформенность). Подсистема визуализации, если она полностью написана на языке программирования высокого уровня, позволяет легко перекомпилировать ее под любую архитектуру процессора и целевую платформу.
- высокая программируемость и гибкость конвейера визуализации, на которую не влияют особенности аппаратной реализации алгоритмов рендеринга.
- независимость от графической подсистемы компьютера.
- возможность легкой отладки любого из этапов визуализации

Главным недостатком систем программной визуализации является сравнительно невысокая скорость отображения данных, по сравнению с аппаратно ускоренной графикой.

Чтобы сократить разрыв в скорости между разрабатываемыми алгоритмами и современными GAPI, необходимо изначально разрабатывать их структуру и оптимизировать вычислительный процесс с учетом архитектур современных многоядерных центральных процессоров [3].

Для расчета сложной трехмерной сцены за минимальное время требуется, чтобы алгоритм растеризации быть легко распараллеливаемым и поддерживал многопоточность, эффективно использовал наборы SIMD инструкций, которые позволяют проводить одну операцию на несколькими наборами операндов (от 2 до 16) одновременно, сокращал число кэш-промахов (обращение к ячейке памяти, которая не была предварительно загружена в сверхбыструю память процессора), минимизировал количество «тяжелых операций» и условных переходов при растеризации примитивов и т.д.

Рассмотрим обобщенную структуру конвейера визуализации (последовательности операций, которая, в конечном счете, приводит к формированию трехмерного изображения) разработанной графической системы.

Чтобы получить трехмерную модель любого объекта его поверхность необходимо предварительно аппроксимировать с помощью набора полигонов (многоугольников) [4], как правило, используются треугольники. Пространственные координаты вершин записываются в специальный массив – буфер вершин. Кроме координат в трехмерном пространстве структура, описывающая вершину треугольника, может содержать в себе текстурные координаты, вектор нормали, цвет и другие параметры, которые необходимы для растеризации треугольника. Если одна вершина является общей для более чем одного полигона, имеет смысл применять индексирование

вершин, при котором на отрисовку передается не только поток вершин, а и номера (смещения) в буфере вершин. Индексирование позволяет избегать повторений одинаковых вершин и может значительно сократить объем памяти, требуемый для описания объекта. Кроме того сокращение количества вершин может значительно ускорить рендеринг в целом.

Алгоритм растеризации начинается с выборки из буфера трех вершин, описывающих треугольник. Если применяется индексирование, считываются 3 индекса в формате целого беззнакового числа разрядностью 16 или 32 бит и загружаются вершины, соответствующие данным индексам. Иначе, из потока извлекаются 3 последовательно расположенные вершины.

После того, как определены 3 вершины треугольника, они подвергаются ряду преобразований [5]. Пространственные координаты переводятся из мировой системы координат в экранную систему путем умножения на матрицу модельно-видово-проекционных преобразований размером 4x4 элемента (1).

$$\begin{bmatrix} x' & y' & z' & w \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \quad (1)$$

Затем, чтобы внести эффект перспективы в трехмерную сцену, x' и y' делятся на w, в результате чего площадь проекции треугольников, расположенных дальше от виртуальной точки наблюдения, получается меньше чем проекция ближних. Для окончательного трансформации координат вершин треугольника необходимо перевести их в экранное пространство в соответствии с (2) и (3).

$$x_{экр} = (\frac{x'}{2 \cdot w} + 0.5) \cdot W_{экр}; \quad (2)$$

$$y_{экр} = (\frac{x'}{2 \cdot w} + 0.5) \cdot H_{экр}, \quad (3)$$

где $W_{экр}$ - ширина графического окна в пикселях;

$H_{экр}$ - высота графического окна в пикселях.

Если структура вершины содержит в себе вектор нормали, то его необходимо преобразовать в видовое пространство, умножив на матрицу 3x3, которая рассчитывается как минор последнего элемента видовой матрицы, и нормализовать (привести к единичной длине) (4).

$$\begin{bmatrix} N'_X & N'_Y & N'_Z \end{bmatrix} = \frac{\begin{bmatrix} N_X & N_Y & N_Z \end{bmatrix}}{|N|} \cdot \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix} \quad (4)$$

После получения двумерных экранных координат треугольника наступает этап заполнения пикселей, находящихся внутри этого треугольника (рис.1). Вершины выводимого треугольника сортируются по y координате. Сам треугольник разбивается на две части горизонтальной линией V_2V_4, проходящей через вершину, которая имеет значение ординаты промежуточное между ординатами вершин V_1 и V_3. Дальнейшее заполнение двух образовавшихся треугольников производится раздельно. Растеризация треугольников осуществляется в цикле по строкам получаемого изображения. В каждой итерации цикла по формуле (5), которая получена из уравнения прямой, проходящей через две точки, рассчитываются координаты крайних пикселей этой строки, лежащие внутри треугольника, а затем заполняется пиксельное пространство между ними.

$$x = \frac{(y - y_1) \cdot (x_2 - x_1)}{y_2 - y_1} + x_1, \quad (5)$$

где x_1, x_2 – абсциссы вершин, образующих данное ребро треугольника;

y_1, y_2 – ординаты вершин.

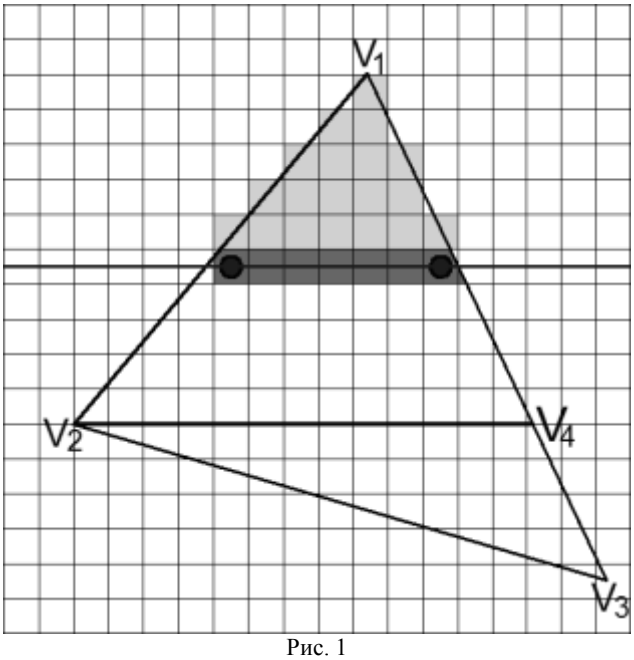

Рис. 1

ВОЗМОЖНОСТИ ПО УСКОРЕНИЮ ВИЗУАЛИЗАЦИИ

В современных GAPI цвет каждого пикселя рассчитывается непосредственно во время заполнения экранного буфера. Растеризация одного фрагмента треугольника может включать в себя операции, выполнение которых может потребовать значительного процессорного времени: интерполяция атрибутов вершин для данного пикселя (текстурных координат, вектора нормали, цвета), расчет освещения от нескольких источников света, выборка из одной или нескольких текстур с фильтрацией результата и т.д. Очевидно, что все эти сложные вычисления могут не повлиять на конечное изображение и быть напрасными, если в дальнейшем они будут перекрыты

фрагментом другого треугольника, расположенного ближе к точке наблюдения. В связи с этим, из соображений оптимизации, была усложнена организация структуры данных для того, чтобы расчет цвета каждого пикселя экрана выполнялся один раз за кадр.

Кроме буфера цвета (формируемое изображение), популярные библиотеки рендеринга содержит невидимый вспомогательный буфер глубины (Z-буффер), имеющий ту же размерность, что и буфер цвета. В него записывается относительное расстояние от точки наблюдения до видимого фрагмента треугольника. Основное назначение это буфера – прекращать дальнейшую обработку фрагмента треугольника, если он находится дальше уже нарисованного фрагмента. Для того, что отложить расчет цвета видимых фрагментов, необходимо так изменить структуру внеэкранного буфера, чтобы после заполнения всех треугольников, для каждого пикселя была возможность быстро восстановить его параметры. С этой целью в разработанной системе структура внеэкранного буфера была дополнена вспомогательными полями: идентификатором треугольника (ID) и двумя весовыми коэффициентами (Ki_1, Ki_2) (рис. 2), которые заполняются во время первичной обработки набора треугольников.

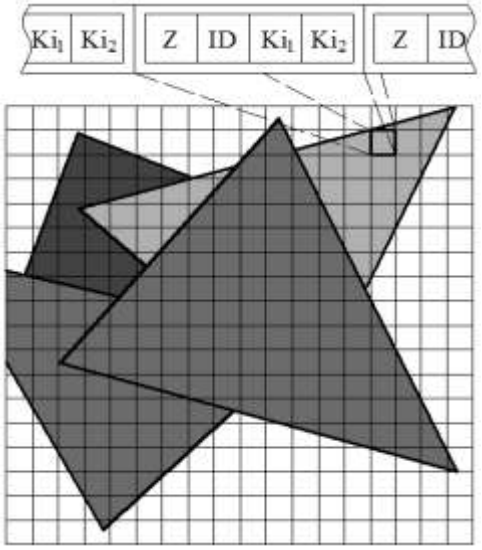

Рис. 2

Каждый из трех весовых коэффициентов определяет степень влияния вершины треугольника на данный пиксель, лежащий в пределах полигона. Так, например, вычисление глубины фрагмента треугольника Z_i производится по формуле (6).

$$Z_i = Z_1 \cdot Ki_1 + Z_2 \cdot Ki_2 + Z_3 \cdot Ki_3, \qquad (6)$$

где Z_1, Z_2, Z_3 – глубина вершин треугольника.

После того, как вычислена глубина пикселя, осуществляется тест глубины – сравнение полученной величины со значением Z-координаты, записанной в буфере. Если она меньше, то глубина данного фрагмента, номер треугольника и два весовых коэффициента записываются во вспомогательный буфер.

Определение весовых коэффициентов осуществляется за счет выполнения трехступенчатой интерполяций (рис 3). Сначала рассчитываются весовые коэффициенты для точек X1 и X2 в зависимости от расстояния до вершин, образующих данные ребра треугольника, а затем происходит еще одна линейная интерполяция для X между точками X1 и X2.

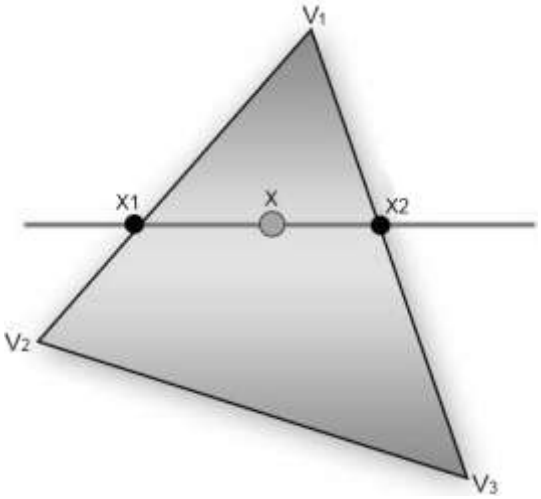

Рис. 3

После обработки всех треугольников наступает последний этап растеризации – цикл вычисления цвета пикселей экранного буфера. По идентификатору треугольника определяются 3 вершины, описывающие треугольник, а с помощью двух весовых коэффициентов можно вычислить третий (т.к. их сумма равняется единице) и произвести интерполяцию любого вершинного атрибута для конкретного фрагмента. Поскольку данные, необходимые для расчета цвета в одной итерации, не зависят от результатов других повторений цикла, то очевидно, что данный цикл может быть эффективно распараллелен и выполняться в нескольких потоках одновременно.

ИСПОЛЬЗОВАНИЕ ОСВЕЩЕНИЯ ПРОСТРАНСТВЕННОЙ МОДЕЛИ

Важной этапом визуализации трехмерных моделей является расчет оптических эффектов, придающих реалистичности визуализируемым объектам [6]. Для учета эффекта освещения, без которого модель будет смотреться плоским силуэтом, необходимо чтобы для каждого треугольника, описывающего элемент поверхности визуализируемой фигуры, была найдена нормаль – вектор, перпендикулярный плоскости треугольника. Освещенность точки пропорциональна косинусу угла θ между нормалью и вектором испускания света направленного источника освещения (7).

$$I = I_0 \cdot \cos\theta, \qquad (7)$$

где I_0 - интенсивность падающего света;

I - интенсивность отраженного света.

Чтобы быстро найти косинус угла между двумя векторами пользуются свойством скалярного произведения векторов:

$$\overline{N} \cdot \overline{L} = \left|\overline{N}\right| \cdot \left|\overline{L}\right| \cdot \cos\theta \,. \qquad (8)$$

Учитывая, что эти векторы были предварительно приведены к единичной длине, результат скалярного произведения будет равен величине косинуса угла между ними.

Зная координаты вершин треугольника, его вектор нормали можно рассчитать как векторное произведение двух векторов, задающих ребра данного треугольника (9) как показано на рис. 4.

$$\overline{N} = \overline{v_1} \times \overline{v_2} \,. \qquad (9)$$

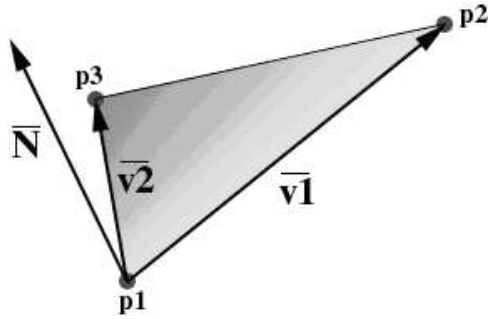

Рис. 4

Рассмотрим пример применения предложенных алгоритмов для задачи визуализации данных спиральной компьютерной томографии. Исходными данными при этом являются 106 изображений аксиальных томографических срезов (см. рис. 5), выполненных с шагом 2 мм и разрешением 512×512 пикселей. Результирующее изображение модели поверхности головы приведено на рис. 6.

Рис. 5

ЗАКЛЮЧЕНИЕ

В статье обоснована необходимость разработки алгоритмов программной визуализации медицинских данных, рассмотрены ключевые алгоритмы и методы машинной графики, перечислены возможные направления ускорения проводимых расчетов, указаны различия в программной и аппаратных реализациях алгоритмов рендеринга и предложены практические рекомендации по разработке алгоритмов при проектировании архитектур систем отображения пространственной информации.

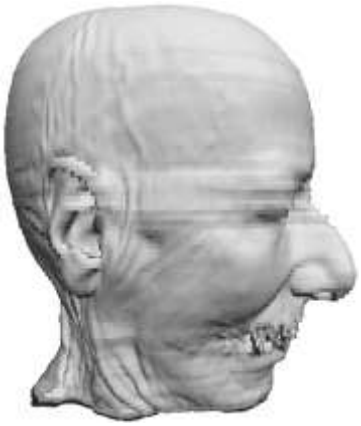

Рис. 6

Перспективным направлением развития методов пространственного рендеринга является внедрение более совершенных, и как следствие, сложных и ресурсоемких алгоритмов отображения данных, таких как трассировка лучей.

Также многообещающим является совмещение элементов разрабатываемых систем программного отображения с существующими библиотеками аппаратного ускорения г0рафики. Частичное замещение элементов графического конвейера этих библиотек позволит добиться улучшения качества получаемого изображения при сохранении высокой скорости вывода, присущей современным GAPI.

Интеграция разработанной системы в программное обеспечение томографических комплексов даст возможность избавиться от многих программных и аппаратных зависимостей, и как следствие, несколько снизить стоимость систем медицинской интроскопической диагностики.

Литература

[1] Аврунин О.Г., Шамраева Е.О. Реконструкция объемных моделей черепа и имплантата по томографическим снимкам // Системы обработки информации: зб. наук. пр. – Х.: ХУПС, 2007. – Вип. 9 (67). – С. 137-140.

[2] Jan Klein, Dirk Bartz, Ola Friman, Markus Hadwiger, Bernhard Preim, Felix Ritter, Anna Vilanova, Gabriel Zachmann. Advanced Algorithms in Medical Computer Graphics. Eurographics, 2008.

[3] Mikhail Smelyanskiy; David Holmes; Jatin Chhugani; Alan Larson; Douglas M. Carmean et al. "Mapping high-fidelity volume rendering for medical imaging to CPU, GPU and many-core architectures". IEEE Transactions on Visualization and Computer Graphics 15 (6): 1563–1570.- 2009.

[4] Levoy,M., "Display of Surfaces from Volume Data", IEEE Computer Graphics and Applications,8(5):29-37, May 1988.

[5] Никулин Е.А. Компьютерная геометрия и алгоритмы машинной графики.–СПб.: БХВ-Петербург, 2003.

[6] Хилл Ф. Программирование компьютерной графики. Для профессионалов. – СПб.: Питер, 2002.