УДК 004.932

С.В. Павлов[1]., д-р техн. наук, проф.,
С.И. Вяткин[2], канд. техн. наук,
С.А. Романюк[1]
[1]Винницкий национальный технический университет
[2]Институт автоматики и электрометрии СО РАН

Многоуровневая объемная визуализация для медицинских приложений

Исследуется объемный рендеринг на основе 3D текстур в интерактивном режиме для множества данных. Предлагается иерархический метод представления текстуры в памяти и способ управления большими массивами вокселей. Иерархическая структура имеет преимущество для обработки однородных областей, представляющих меньший интерес. Предлагаемый метод основан на обходе восьмеричного дерева представления данных объема. Качество изображения определяется пользователем, которое зависит от области интереса и важности данных. С помощью переменной модели разрешения текстуры можно уменьшить размер памяти текстуры и повысить скорость рендеринга.

Ключевые слова: объемный рендеринг, восьмеричное дерево, 3D отображение текстуры, многоуровневое представление данных.

DOI: 10.31474/1996-1588-2018-1-26-55-62

Введение

В последние годы актуальными есть новые методы визуализации, поддерживающие режим реального времени и обрабатывающие большие объемы данных. Использование специализированных аппаратных подсистем позволило достичь реального времени при прямом рендеринге объема [1, 2]. В частности, 3D отображение текстуры стало классическим методом объемного рендеринга для высокопроизводительных графических рабочих станций. Хотя аппаратная реализация позволяет генерировать изображения с высоким качеством, на практике ее использование в интерактивном режиме ограничено. Размер текстурной памяти на стандартных графических подсистемах ограничен и трудно поддерживать интерактивную частоту кадров, когда большие массивы данных подлежат обработке. Когда большой массив данных не помещается в доступную память текстур, необходима декомпозиция данных. В этом случае загрузка и выгрузка данных текстурной памяти, необходимых для визуализации всего объема, существенно уменьшают производительность рендеринга. Сокращение объема памяти текстуры, необходимого для визуализации массива данных и, таким образом, уменьшение времени загрузки текстуры, является основной целью данной работы.

Также предлагается способ управления памятью текстур. Он основан на многоуровневом представлении массива данных объема в виде 3D-пространства текстур. В предлагаемом методе используется структура восьмеричного дерева для исходного массива данных, которая может быть вычислена во время предварительной обработки. Во время рендеринга, выбираются узлы восьмеричного дерева. Выбор узлов управляется определяемыми пользователем критериями точности представления данных и их важности. Степень разрешения (количество текселей), необходимая для представления каждого узла в пространстве текстур фиксирована. Выбор узлов определяется декомпозицией объема. Узлы одинакового размера могут представлять разное количество текселей. Модель текстуры с переменным разрешением гарантирует пользователю желаемое качество конечного изображения.

Объемная визуализация на основе 3D текстуры

Аппаратная реализация рендеринга 3D текстуры впервые упоминается в работах [1, 3]. В них используется алгоритм BTF. Этот алгоритм отображает объем данных в виде параллельных плоскостей. В результате множество плоскостей рисуется как набор текстурированных полигонов, которые смешиваются для получения окончательного изображения. Повышение эффективности достигается путем использования трилинейной интерполяции, поддерживаемой графикой подсистемы. Операции отображения и композиции текстур выполняются аппаратно очень быстро. Различают два основных метода рендеринга объема: рендеринг в объектном пространстве или рендеринг прямого хода (object-space rendering/forward rendering) и метод пространства изображения (image-space method/backward viewing method).

В первом методе при обходе всего объема каждый воксель рассматривается как 3D точка, которая преобразуется по матрице видового пре-

образования, затем проецируется в Z-буфер и рисуется на экране. Существуют два алгоритма обхода объема: алгоритм BTF (back to front) по существу совпадает с алгоритмом Z-buffer и отличается только предварительной сортировкой массива вокселей, которая позволяет сканировать его компоненты в порядке уменьшения или увеличения расстояния до наблюдателя, используя предварительно отсортированные массивы вокселей. В алгоритме BTF обход объема производится в порядке уменьшения расстояния до наблюдателя, а Z-buffer нужен для удаления невидимых частей объекта [4].

Алгоритм FTB (front to back) в основном совпадает с BTF, однако в нем воксели обрабатываются в порядке возрастания расстояния [5, 6]. Это надо отметить, имея в виду то, что алгоритм Z-buffer не может обеспечивать правильный рендеринг полупрозрачных материалов, так как воксели отображаются на экран в произвольном порядке.

Во втором методе через каждый пиксель плоскости изображения пускается луч от взгляда наблюдателя сквозь объём [7, 8]. В каждом полученном пересечении луча с объемом вычисляется коэффициент непрозрачности, учитывая влияние ближайших вокселей. На протяжении траектории луча вычисляется сумма непрозрачности пересечений, пока она не достигнет определённого значения. Рассчитывается тень и освещение для полученного изображения. То есть при таком методе на один пиксель плоскости изображения влияет несколько вокселей.

С момента использования 3D текстур для объемной визуализации, было предложено несколько улучшенных подходов [9–11]. Представление информации об объеме в текстурной памяти - один тексель (элемент текстуры) на воксель. Эта стратегия гарантирует хорошее качество изображения, но ограниченный размер памяти текстур часто делает невозможным хранение всего набора данных сразу в памяти текстуры. В некоторых подходах было предложено снятие этого ограничения. Например, метод основанный на разложении воксельного пространства и загрузки фрагментов текстурных данных в память текстуры по запросу. Фрагмент соответствует подмножеству воксельных данных, которые по определению, меньше или равны доступному объему памяти. Каждый фрагмент обрабатывается независимо, и окончательный результат зависит от этого вклада. К сожалению, загрузка текстур и перекачка данных резко снижают частоту кадров. Процесс обработки фрагментов может быть выполнен с различными стратегиями, в основе которых лежит нижеследующие критерии: простое разделение данных. Объем можно разложить на наименьшее число фрагментов с максимальным размером. Этот критерий сокращает объем избыточных данных, которые необходимо хранить между смежными фрагментами для сохранения непрерывности [12]. Метод отсутствия пустого воксельного представления. При разложение фрагментов пытаться уменьшить процент пустых вокселей, представленных в памяти текстур. Пустые области обнаруживаются во время предварительной обработки [9, 10]. В представлении со многими разрешениями разложение фрагментов основано на оценке пользовательских параметров, таких как степень интереса, степень однородности, расстояние от наблюдателя, и т.д. [13, 14]. Иерархические структуры данных и адаптивные методы описаны в [15, 16, 17]. В [9] применяется восьмеричное дерево для обработки больших массивов данных, с пропуском пустых областей объема; аналогичный подход был использован в [10]. В этих методах, окончательное текстурное представление объема - один тексель на воксель, таким образом для больших массивов, данных ограничение памяти текстур все еще может быть проблемой. Иерархическая организация данных показана в [14], где разделам могут быть назначены различные разрешения исходного набора данных.

Новая стратегия, в которой данные текселя представляют значения плотности и градиента [18, 19]. В этом случае представление текстуры больше не зависит от вида и затененных изображений. Подход, представленный в этой статье, аналогичен [14]. Основные отличия: представление данных объема и критерии, используемые для организации данных; подход, разработанный для выбора массива данных (фрагменты) и для определения разрешения, при котором представлен каждый фрагмент. Предлагаемый метод не учитывает зависимые от представления критерий (которые, как правило, основаны на искажении перспективы и сокращении дальних фрагментов), потому что, в объемном рендеринге различия в проецировании очень ограничены. Из-за ограниченного пространственного массива данных объема, все воксели имеют приблизительно равный размер проекции. Причем, параллельные проекции используются очень часто. Поэтому в центре внимания остаются конструкции независимого от наблюдателя многозначного критерия, который должен учитывать однородность данных и область интереса.

Визуализация больших массивов данных

Цель данной работы заключается в ускорении рендеринга больших массивов данных на основе трехмерной текстуры. Рассматривается большой массив данных объема, размер которого превышает доступную память текстур. Предлагается метод, основанный на иерархическом представлении данных с помощью восьмеричного дерева, в котором выполняется выборочный обход узлов для получения адаптивного представления данных в трехмерно текстурном пространстве.

Дерево строится во время предварительной обработки. Для этого используются атрибуты, характеризующие соответствующий воксельный субрегион в терминах однородности данных и хранятся в каждом узле восьмеричного дерева. Набор узлов определяет текущую декомпозицию, используемую для визуализации объема. Учитывая представление восьмеричного дерева массива данных объема, определяется отрезок C = {n0, n1, ..., nnb} такой, что для каждого возможного пути от корня к листьям дерева, один и только один из его узлов содержится в выбранном узле. Набор текстурных фрагментов (количество и пространственная непрерывная область в воксельном пространстве), будет использоваться при рендеринге. Для выполнения рендеринга, количество фрагментов не может быть слишком большим, а пространственная протяженность не может быть маленькой, иначе количество текстурированных полигонов резко увеличится. Воксели исходного массива данных могут быть представлены с разной точностью, путем присвоения текстур фрагментам различного разрешения в каждом узле дерева. Для каждого узла, определяется оптимальная текстура фрагмента Bi, соответствующая шестигранной области, охватываемой этим узлом. Разрешение устанавливается в соответствии с пользовательским запросом к качеству изображения, т. е. параметры однородного содержания (порога точности представления) и / или важности соответствующего региона (определяется выбором областей фокуса), которые определяют степень точности (максимальную погрешность) для рендеринга.

Рендеринг с обходом восьмеричного дерева

Предлагаемый метод состоит из трех шагов: (1) Конструирование восьмеричного дерева, (2) выбор узлов и (3) рендеринг выбранных данных.

Конструкция дерева нуждается в определении критериев, чтобы определить, является ли область объема важной. Эти критерии варьируются от самых простых, которые оценивают "присутствие" или "отсутствие" значимых данных, до более сложных, которые определяют, связаны ли образцы объема с узлом и меняются в зависимости от данной реконструкции. Это сделано для того, чтобы обнаружить однотипные зоны. Большие разделы данных, где значения полей являются однородными, могут быть представлены с определенной степенью приближения, по восьми значениям в углах, связанной области пространства (и все внутренние данные могут аппроксимироваться с помощью интерполяционного ядра, например, трилинейной интерполяции). В пространственной иерархии дерева требуется не только идентифицировать почти однородные области, но также оценить и сохранить точность, связанную с альтернативными представлениями при разных разрешениях одной и той же области узла. Она необходима, чтобы интегрировать зависящий от пользователя критерий в онлайн режиме выбор приоритетных направлений (регионов с большим интересом, где представление при максимальном разрешении всегда предпочтительно). Восьмеричное дерево строится из входного набора вокселей. Воксельные значения представлены неявно в узлах дерева (непрерывная область вычисляется непосредственно с помощью индекса узла). Когда построение дерева завершено, одноуровневые узлы объединяются в родительском узле. Для каждого узла ni уровня K восьмеричного дерева с глубиной l макс, сохраняется список узловых ошибок. При Узловой ошибке все воксельные значения в регионе, заменяются трилинейной интерполяцией на основе восьми угловых значений. Эти узловые ошибки дают меру степени однородности в области, охватываемой узлом. На практике, для каждого узла Узловая ошибка дает нам меру точности, когда присваивается узлу ni фрагмент текстуры определенного разрешения.

Перед выбором узла и определением фрагментов, анализируется представления узла восьмеричного дерева в памяти текстуры, и определяется оптимальная с точки зрения точности экономия памяти.

Для представления области, покрытой узлом ni в текстурной памяти можно рассмотреть классическое присваивание узла с текселем на лист дерева (воксель - в полном восьмеричном дереве), или более обобщающее представление, в котором один тексель может представлять несколько терминальных узлов (рис. 1). В первом случае, значения в памяти текстуры соответствуют исходным данным объема. Во втором случае, получается более компактное представление данного узла. То есть разный результат рендеринга, можно получить путем применения различных соотношений воксель-тексель: 1: 1, 1: 2, 1: 4. На каждом шаге, размер текстуры будет соответствовать 1/8 предыдущему образцу. Для каждой вершины ni уровня k имеется соответствующее поддерево глубины l max и соответствующая воксельная область разрешения. Можно представить воксельную область, связанную с помощью одного из регулярно уменьшенного представления, построенного из исходных воксельных значений. Они явно не хранятся в восьмеричном дереве, но могут быть просто извлечены из иерархического представления воксельного массива данных (пирамида 3D воксельных массивов, каждый из которых получен из предыдущего уровня). Поэтому, если выбирается массив на самом глубоком уровне, то процесс заканчивается (один тексель - на каждый узел). Во всех остальных случаях выбирается более компактное представление, соответствующее узлам, которые по уровням ниже в структуре восьмеричного дерева. В этом случае

необходимо рассмотреть точность текстурного представления, которая кодируется в соответствующей узловой ошибке узла. Память текстур, необходимая для узла и некоторые дополнительные ресурсы текстурной памяти, необходимые для хранения информации о границе между соседними фрагментами. Это значение зависит от используемой схемы фильтрации; в случае трилинейной интерполяции, необходимо добавить слой вокселей толщиной в один воксель в каждом направлении

Рисунок 1 – Воксель – тексель режим. Объем 256x256x256 вокселей

Обход восьмеричного дерева осуществляется сверху вниз, чтобы выбрать узлы отрезка. Сжать информацию можно, если рассматривать возможные однородные субрегионы массива данных. Также принимается во внимание два пользовательские параметра: порог ошибки и важность узла. Уровни, которые могут гарантировать отсутствие ошибки текстуры должны быть меньше, чем допускаемая пользователем ошибка. Чтобы оценить важность узла, необходимо определить функцию области интереса. Эта функция взвешивает важность узла с учетом региона интереса (ROI), что в свою очередь может быть определено пользователем с выбором 3D субрегиона домена данных или определенного поддиапазона значения поля по максимальным и минимальным значениям. По определению, важность узла ni по отношению к функции f будет значение в интервале [0,1]. Так как «важность» зависит от текущей функции F (которую можно динамически отрегулировать), значение важности не сохраняется в восьмеричном дереве, а вычисляется в процессе обхода восьмеричного дерева. Кроме того, определяется вторая функция в домене с интервалом [0,1], которая дает меру того, как много значений Imp (ni, f) равномерно в пространстве связаны с узлом ni. Поэтому, малое значение функции Uni f (ni, f) показывает ситуации, где только часть пространства, связанная с данным узлом содержится в области фокуса. Это может быть использовано для выбора разложения пространства (например, соответствие потомку узла), что может привести к лучшему представлению данной области. Учитывая все эти ограничения, можно оценивать важность узлов восьмеричного дерева во время его обхода. Ошибка приближения в текстурном представлении одного узла получается путем взвешивания выбранного пользователем порога важность узла. Каждый узел связан со своей собственной, определяемой пользователем "идеальной ошибкой". Для регионов с малым интересом будет маленькое значение, которое позволяет использовать грубое представление для связанного фрагмента. Разложение объема фрагмента осуществляется функцией обхода сверху вниз. Невозможно генерировать большое количество текстурных фрагментов (для поддержания издержек на визуализацию и избыточности данных на приемлемом уровне), поэтому ограничивается уровень узлов, которые можно использовать при отрисовки интервала [0, l макс], где l макс глубина исходного

восьмеричного дерева. Таким образом, в худшем случае, массив данных состоит из не более чем фрагментов, каждый из них представляет подмножество объемов по крайней мере $(2^{\min PartSize})^3$ вокселей. Для визуализации текстур с другим разрешением создается ряд плоскостей, пропорциональных разрешению текстуры. Если регион, связанный с корневым узлом достаточно однороден и представление минимального размера удовлетворяет (меньше или равно максимуму текстуры Tmax), то данные визуализируются. В противном случае процедура рекурсивно применяется к потомкам текущего узла, с ограничением, что сумма представления текстуры, выбранная для всех потомков узла должна быть меньше Tmax. Обход восьмеричного дерева продолжается, пока не найдено условие, когда каждый узел удовлетворяет однородности данным и точности текстуры. Может случиться так, что размер фрагментов декомпозиции больше чем доступная текстурная память Tmax. В этом случае, осуществляется ограничение по точности, и некоторые из узлов визуализируются с более компактными представлениями.

.После выбора узла и представлений с выборками для каждого узла, необходимо преобразовать значения вокселей в тексели. Текстурные данные загружаются в память текстур, которые прежде нужно упаковать в прямоугольный параллелограмм, каждое измерение ограничено степенью два. Качество изображения должно быть сохранено для этого каждая текстура должна иметь общую границу с соседними узлами текстуры. Также необходимо учитывать непрозрачность, назначенную текселям. Чтобы уменьшить разницу между различными выбранными разрешениями текстур (рис. 2), применяется коэффициент коррекции непрозрачности.

Сложность вычисления этого значения зависит от критериев, применяемых для построения восьмеричного дерева. Однако, поскольку применяются более грубые представления в областях меньшего интереса правило дает достаточно хороший результат. Хотя это приближение приводит к некоторым ошибкам, они не критичны.

Функция визуализации узла основана на алгоритме, предложенном в [20]; все преобразования текстурного пространства и отсечение плоскости текстурирования / композитинга реализованы с помощью стандартных функций графической библиотеки. Описание функции визуализации узла применяется к ni разрезу. Текстурное представление этого узла - Ti, а npi - количество полигонов, которые будут использоваться для рендеринга. Функцию можно разложить на следующие действия:

1. Определение ограничивающего куба.

Ограничивающий куб по центру узла определен ni. Соответствующие полигоны npi должны быть визуализированы серией срезов через bni, параллельно к наблюдателю xy.

2. Активация плоскостей отсечения.

Набор полигонов npi для визуализации всегда остается параллельным плоскости проекции. Объем узла, представленного в Ti вращается в bni в зависимости от направления просмотра. Чтобы гарантировать, что только информация об объеме, представленном в текстуре будет соответствовать окончательному изображению, определен набор плоскостей отсечения. Отсекающие плоскости расположены согласно направлению просмотра на каждой из граней объема узла.

3. Позиционирование узла.

После отсечения полигонов, которые представляют узел объема, определяются места, где они должны быть спроецированы. Положение этих полигонов фиксируется положением узла восьмеричного дерева и направлением просмотра. Эти параметры определяют текущий набор геометрических преобразований. Геометрическая трансформация определяет положение согласно позиции восьмеричного дерева. Новая ориентация требует вращения пространства текстуры. Это вращение является независимым для каждого узла и зависимым для направлении просмотра

Заключение

В данной работе исследуется объемный рендеринг 3D текстур больших массивов данных, основанный на представлении восьмеричного дерева данных объема. В методе предлагается компактное представление текстуры с использованием однородности данных и важности информации для уменьшения требуемой памяти текстуры и скорости вычислений. В основе метода лежит эффективное управление текстурами, в котором назначается память текстур по степени значимости региона и содержанию воксельных данных. Сочетание данных, ошибки интерполяции и важности данных восьмеричного дерева определяют выбранный набор узлов. Эти узлы определяют, как должен быть объем разложен и представлен в памяти текстур. Экспериментальные результаты показывают хорошее качество визуализации. Тестирование производилось на компьютере с процессором Intel Core2 CPU E8400 3.0 GHz, и графическим акселератором GTX 470.

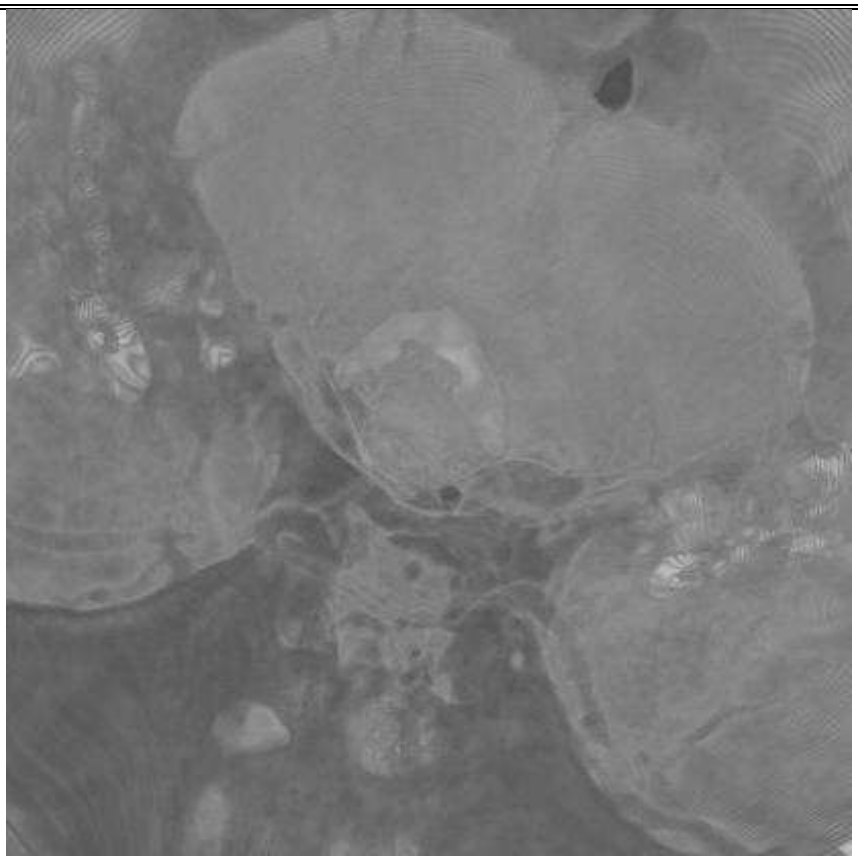

Рисунок 2 – Модель областей с различным разрешением объемного рендеринга

Список литературы

1. B. Cabral, N. Cam, J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In: Kaufman A, Krueger W (eds) Proceedings of the 1994 Symposium on Volume Visualization, Washington, DC, P. 91–98.

2. H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, L. Seiler. (1999) The VolumePro real-time ray-casting system. Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '99) Los Angeles, CA, P. 251–260.

3. S-Y. Guan, R. Lipes. Innovative volume rendering using 3D texture mapping. In Proceedings Medical Imaging – Image Capture, Formatting and Display, vol 2164, SPIE, 1994, P. 382–392.

4 . G. Frieder, D. Gordon, and R.A. Reynolds. Back-to-Front Display of Voxel-Based Objects, IEEE Computer Graphics and Applications, 5(1):52-60, January 1985.

5 R.A. Drebin, L. Carpenter, and P. Hanrahan. Volume Rendering, Computer Graphics, 22(4):65-74, August 1988.

6 . P. Lacroute, and M. Levoy, "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation", Proceedings SIGGRAPH' 94, Orlando, Florida, July 24-29, 1994. In Computer Graphics Proceedings, Annual Conference Series, 1994, ACM SIGGRAPH, pp.451-458.

7. H. K. Tuy and L.T. Tuy. Direct 2-D Display of 3-D Objects, IEEE Computer Graphics and Applications, 4(10):29-33, November 1984.

8 . C. Upson and M. Keeler. V-BUFFER: Visible Volume Rendering, Computer Graphics, 22(4):59-64, August 1988.

9 .R. Srinivasan, S. Fang, S. Huang. Rendering by template-based octree projection. In: Lefer W, Grave M (eds) Proceedings of the 8th Eurographics Workshop on Visualization in Scientific Computing, Boulogne-sur-Mer, France, Eurographics, 1997, P.155–163.

10 .X. Tong, W. Wang, W. Tsang, Z. Tang. Efficiently rendering large volume data using texture mapping hardware. In: Groller E, Loffelmann H, Ribarsky W (eds) Data Visualization '99, Eurographics. Springer, Vienna, 1999, P.121–132.

11. A. Van Gelder, K. Kim. Direct volume rendering with shading via three-dimensional textures. In Crawfis R, Hansen C (eds) 1996 Symposium on Volume Visualization, San Francisco, CA, 1996, P. 23–30.

12. R. Grzeszczuk, C. Henn, R. Yagel. Advanced geometric techniques for ray casting volumes. ACM SIGGRAPH '98, Course Notes N. 4, 1998, P. 1–239.

13. I. Boada, I. Navazo, R. Scopigno. A 3D texture-based octree volume visualization algorithm. In: Skala V (ed) Short Communication papers of the 8th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media '2000, University of West Bohemia, Plzen,

Czech Republic, 2000, P. 1–8.

14. E.C LaMar, B. Hamann, K.I Joy. Multiresolution techniques for interactive texture-based volume visualization. In: Ebert D, Gross M, Hamann B (eds) Proceedings of the 1999 IEEE Conference on Visualization (VIS-99). ACM Press, San Francisco, CA, 1999, P. 355–362.

15 . J. Danskin, P. Hanrahan. Fast algorithms for volume ray tracing. Proceedings of the Workshop on Volume Visualization, ACM Press, New York, 1992, P. 91–98.

16. D. Laur, P. Hanrahan. Hierarchical splatting: a progressive refinement algorithm for volume rendering. (SIGGRAPH'91) Computer Graphics 25:285–288.

17. J. Wilhelms, A. Van Gelder. Multi-dimensional trees for controlled volume rendering and compression. In Kaufmann A, Krueger W(eds) Proceedings of 1994 Symposium on Volume Visualization. Washington, DC, 1994, P. 27–34.

18. M. Meissner, U. Hoffmann, W. Straber. Enabling classification and shading for 3D texture mapping based volume rendering. In: Ebert D, Gross M, Hamann B (eds) Proceedings of the 1999 IEEE Conference on Visualization (VIS-99), ACM Press, San Francisco, CA, 1999, P. 207–214.

19. R. Westermann, T. Ertl. Efficiently using graphics hardware in volume rendering applications. In: Cohen M (eds) Conference Proceedings, Annual Conference Series (SIGGRAPH '98), Addison Wesley, 1998, P. 169–178.

20. A. Van Gelder, K. Kim. Direct volume rendering with shading via three-dimensional textures. In Crawfis R, Hansen C (eds) 1996 Symposium on Volume Visualization, San Francisco, CA, 1996, P. 23–30.

References

1. B. Cabral, N. Cam, J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In: Kaufman A, Krueger W (eds) Proceedings of the 1994 Symposium on Volume Visualization, Washington, DC, P. 91–98.

2. H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, L. Seiler. (1999) The VolumePro real-time ray-casting system. Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '99) Los Angeles, CA, P. 251–260.

3. S-Y. Guan, R. Lipes. Innovative volume rendering using 3D texture mapping. In Proceedings Medical Imaging – Image Capture, Formatting and Display, vol 2164, SPIE, 1994, P. 382–392.

4 . G. Frieder, D. Gordon, and R.A. Reynolds. Back-to-Front Display of Voxel-Based Objects, IEEE Computer Graphics and Applications, 5(1):52-60, January 1985.

5 R.A. Drebin, L. Carpenter, and P. Hanrahan. Volume Rendering, Computer Graphics, 22(4):65-74, August 1988.

6 . P. Lacroute, and M. Levoy, "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation", Proceedings SIGGRAPH' 94, Orlando, Florida, July 24-29, 1994. In Computer Graphics Proceedings, Annual Conference Series, 1994, ACM SIGGRAPH, pp.451-458.

7. H. K. Tuy and L.T. Tuy. Direct 2-D Display of 3-D Objects, IEEE Computer Graphics and Applications, 4(10):29-33, November 1984.

8 . C. Upson and M. Keeler. V-BUFFER: Visible Volume Rendering, Computer Graphics, 22(4):59-64, August 1988.

9 .R. Srinivasan, S. Fang, S. Huang. Rendering by template-based octree projection. In: Lefer W, Grave M (eds) Proceedings of the 8th Eurographics Workshop on Visualization in Scientific Computing, Boulogne-sur-Mer, France, Eurographics, 1997, P.155–163.

10 .X. Tong, W. Wang, W. Tsang, Z. Tang. Efficiently rendering large volume data using texture mapping hardware. In: Groller E, Loffelmann H, Ribarsky W (eds) Data Visualization '99, Eurographics. Springer, Vienna, 1999, P.121–132.

11. A. Van Gelder, K. Kim. Direct volume rendering with shading via three-dimensional textures. In Crawfis R, Hansen C (eds) 1996 Symposium on Volume Visualization, San Francisco, CA, 1996, P. 23–30.

12. R. Grzeszczuk, C. Henn, R. Yagel. Advanced geometric techniques for ray casting volumes. ACM SIGGRAPH '98, Course Notes N. 4, 1998, P. 1–239.

13. I. Boada, I. Navazo, R. Scopigno. A 3D texture-based octree volume visualization algorithm. In: Skala V (ed) Short Communication papers of the 8th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media '2000, University of West Bohemia, Plzen,

Czech Republic, 2000, P. 1–8.

14. E.C LaMar, B. Hamann, K.I Joy. Multiresolution techniques for interactive texture-based volume visualization. In: Ebert D, Gross M, Hamann B (eds) Proceedings of the 1999 IEEE Conference on Visualization (VIS-99). ACM Press, San Francisco, CA, 1999, P. 355–362.

15 . J. Danskin, P. Hanrahan. Fast algorithms for volume ray tracing. Proceedings of the Workshop on Volume Visualization, ACM Press, New York, 1992, P. 91–98.

16. D. Laur, P. Hanrahan. Hierarchical splatting: a progressive refinement algorithm for volume rendering. (SIGGRAPH'91) Computer Graphics 25:285–288.

17. J. Wilhelms, A. Van Gelder. Multi-dimensional trees for controlled volume rendering and compression. In Kaufmann A, Krueger W(eds) Proceedings of 1994 Symposium on Volume Visualization. Washington, DC, 1994, P. 27–34.

18. M. Meissner, U. Hoffmann, W. Straber. Enabling classification and shading for 3D texture mapping based volume rendering. In: Ebert D, Gross M, Hamann B (eds) Proceedings of the 1999 IEEE Conference on Visualization (VIS-99), ACM Press, San Francisco, CA, 1999, P. 207–214.

19. R. Westermann, T. Ertl. Efficiently using graphics hardware in volume rendering applications. In: Cohen M (eds) Conference Proceedings, Annual Conference Series (SIGGRAPH '98), Addison Wesley, 1998, P. 169–178.

20. A. Van Gelder, K. Kim. Direct volume rendering with shading via three-dimensional textures. In Crawfis R, Hansen C (eds) 1996 Symposium on Volume Visualization, San Francisco, CA, 1996, P. 23–30.

Надійшла до редакції 15.11.2017

SERGII V. PAVLOV[1] , SERGEY I. VYATKIN[2], SERGII O. ROMANYUK[1]
[1]Vinnytsia National Technical University,
[2]Institute of Automation and Electrometry SB RAS

MULTILEVEL VOLUMETRIC VISUALIZATION FOR MEDICAL APPLICATIONS

The size of the texture memory on standard graphics subsystems is limited and it is difficult to maintain the interactive frame rate, when large data sets are subject to processing. When a large array of data is not placed in the available texture memory, data decomposition is necessary. In this case, loading and unloading the texture memory data required to render the entire volume significantly reduces rendering performance.

In this paper, we study the volumetric rendering of 3D textures of large data sets based on the presentation of the octal data volume tree. The method proposes a compact representation of the texture using the uniformity of the data and the importance of information to reduce the required texture memory and computational speed. The method is based on the effective management of textures, in which texture memory is assigned according to the degree of significance of the region and the content of voxel data. The combination of data, interpolation errors, and the importance of the octal tree data determine the selected set of nodes. These nodes determine how the volume should be decomposed and represented in the texture memory. The experimental results confirmed the good quality of visualization.

Keywords: volumetric rendering, octal tree, 3D texture mapping, multi-level data representation.

С. В. ПАВЛОВ[1], С. И. ВЯТКИН[2], С. О. РОМАНЮК[1]
[1]Вінницький національний технічний університет
[2]Институт автоматики и электрометрии СО РАН

БАГАТОРІВНЕВА ОБ'ЄМНА ВІЗУАЛІЗАЦІЯ ДЛЯ МЕДИЧНИХ ЗАСТОСУВАНЬ

Досліджується об'ємний рендеринг на основі 3D текстур в інтерактивному режимі для множинних даних. Пропонується ієрархічний метод подання текстури в пам'яті та спосіб управління великими масивами вокселей. Ієрархічна структура має перевагу для обробки однорідних ділянок. Запропонований метод оснований на обході вісімкованого дерева представлення даних об'єму. Якість зображення визначається користувачем, яке залежить від виділеної ділянки і важливості даних. За допомогою змінної моделі дозволу текстури можна зменшити розмір пам'яті текстури та підвищити швидкість рендеринга.

Ключові слова: об'ємний рендеринг, вісімкове дерево, 3D відображення текстури, багаторівневе представлення даних.