Адаптивный рендеринг трехмерных сцен на основе иерархических ячеек

Александр Жирков, Александр Паршин
Лаборатория Компьютерной Графики факультета ВМиК МГУ
Москва, Россия
azh@graphics.cs.msu.ru, savo@rambler.ru

Аннотация

В данной статье речь идёт о работе со сценами, представленными в виде структуры иерархических ячеек (Hierarchical Cells, HC). Вводится и раскрывается ключевое понятие реализации структуры HC, рассматриваются операции над ними, а также критерии выбора наилучшей реализации. Такой подход, сохраняя заданную скорость рендеринга, позволяет адаптивно контролировать детализацию участков сцен в зависимости от множества факторов, таких как положение и направление взгляда наблюдателя, динамика его движения, мощность графического акселератора, полоса пропускания канала в клиент-серверной архитектуре. Также рассматривается пример использования этой структуры для рендеринга сцен исходно представленных в виде *октоизображений*.

Keywords: *иерархические ячейки, адаптивный рендеринг, октоизображение, HC, Hierarchical Cells, DIBR, octreeimage, adaptive rendering, progressive transmission*.

1. ВВЕДЕНИЕ

В настоящее время главной задачей интерактивного рендеринга является обеспечение его адаптивности. Понятие адаптивности подразумевает способность технологий максимизировать качество выполнения задачи при заданных условиях. В контексте адаптивной графики, такой задачей является максимизация качества, при сохранении интерактивности рендеринга трехмерных объектов/сцен, которая зависит от таких факторов как:

- Скорость процессора и графического акселератора, начиная от карманных компьютеров (PDA), заканчивая графическими станциями
- Местоположение и скорости поступления 3Д-контента, будь то жесткий диск локального компьютера или соединение через сеть интернет
- Сложность 3Д-контента, варьируемая от простого 3Д-объекта до детализированных представлений сцены города
- Метод создания объекта, будь то метод моделирования в пакете 3D-моделирования, или созданный полуавтоматически, переданный из реального мира посредством 3D-сканеров, или компьютерного зрения

Одновременное выполнение требований универсальности и адаптивности могут привести к слишком громоздкому устройству системы, подстраивающейся под конкретный контекст. Поэтому, разумным выглядит компромисс между универсальностью, адаптивностью и сложностью системы. На настоящий момент существует несколько методов, решающих эту задачу.

Одним из первых и наиболее популярным на сегодня является метод представления данных на основе полигональных сеток. Разработано множество методов адаптивного контроля уровня сложности трехмерных полигональных объектов [1, 2]. Но все эти подходы адекватны только для работы с объектами с достаточно простой топологией. Но даже в этом случае, во многих приложениях, например играх, для достижения максимальной производительности используют ручную оптимизацию.

К другим способам упрощения трехмерных сцен и объектов, относятся методы, заменяющие трехмерные объекты изображениями, например [3]. Такие методы не содержат топологических ограничений, однако, их использование возможно только для отдаленных объектов, вблизи же возникают эффекты параллакса и нестыковки граней.

Рассмотрим представления, которые имеют более мягкую топологию, нежели полигональные представления и имеют более широкую область применения, нежели изображения-заменители:

- Точечные представления (Splat [4]).
- Представления на основе изображений с глубинами [5,6]
- Различные гибридные представления, сочетающие свойства первых двух. Например, *октоизображения* [10, 11], включающие в себе так же свойства и объёмных представлений.

Технология Qsplat [4] предназначена для объектов, составленных из точек, полученных из трехмерного сканера. В нем точки структурированы в виде вложенных сфер, таким образом адаптивный контроль детализации лежит в основе данного представления. Другим иерархическим представлением, ориентирующимся на представление сцен, а не объектов, является представление LDI-Tree[5]. В этом представлении строится октодерево, в каждом узле которого храниться слоистое изображение с глубиной (Layered Depth Image) с фиксированным разрешением. Условием выбора определенного уровня детализации было достижение уровня размеров пикселей, сопоставимое с разрешением LDI.

Ванд и Штрабер [7] использовали аналогичный метод при работе с анимированными сценами.

Однако во всех этих подходах стояла задача максимизации скорости, но не решалась задача фиксированной скорости рендеринга.

К другим, отмеченным нами областям адаптивности, относится адаптивная передача 3D-контента по сети. До настоящего момента выходило только несколько статей посвященных прогрессивной передачи 3D. Так, к передаче объекта относится технология Streaming Qsplat [8] прогрессивно передающая QSplat по сети.

По методу Телера и Лишински [9] передаётся уже не один объект, а целая группа объектов, каждый из которых может быть передан либо изображением-заменителем, либо полигональным представлением с определенным уровнем детализации. Заметим, что в данной технологии адаптивно выбирается только объем передаваемых данных в зависимости от полосы канала, но не скоростные возможности рендеринга на клиенте.

В данной статье делается попытка обобщить данные по этой проблеме и ввести теоретическую основу для универсального представления сцен, способного адаптироваться как к ширине канала, так и к скорости рендеринга и положению наблюдателя. Таким образом, ставится задача одновременно реализовывать и прогрессивную передачу данных, и адаптивный рендеринг.

Наиболее актуальными эти задачи становятся для целей рендеринга больших сцен на маломощных компьютерах, например, на КПК, а в будущем и на мобильных телефонах. Для этого, исходная сцена должна быть представлена специальным образом, позволяющим быстро выбирать информацию, необходимую для пользователя в данный момент. При наличии клиент-серверной архитектуры, возникает задача быстрой (не требующей больших затрат на извлечение на сервере и добавления на клиенте) передачи этой информации.

При этом, наиболее эффективно представлять сцену, в виде, позволяющем совмещать в себе объекты в различных представлениях, каждое из которых эффективно отображает свою часть сцены при заданных условиях на соотношения качества и скорости.

Однако, в данной статье рассматривается только одно представление - *октоизображение*, поскольку как было сказано ранее, обладает преимуществами, как точечных представлений, так и представлений на основе изображений и сочетает в себе объемно-воксельную структуризацию пространства.

Для решения рассмотренных задач, предлагается использовать специальное иерархическое представление, называемое Hierarchical Cells (иерархические ячейки), или сокращенно HC. Основная идея работы с HC заключается в адаптивном многомасштабном разбиении сцены на отдельные независимые ячейки, каждую из которых можно визуализировать отдельно. Такой подход, с одной стороны даёт большую свободу для адаптации к различным параметрам, а с другой позволяет сохранять консистентность сцены.

2. ИЕРАРХИЧЕСКИЕ ЯЧЕЙКИ (HC)

Для представления сцены в HC, необходимо создать несколько версий сцены с различными уровнями детализации. HC является октодеревом, каждому узлу которого соответствует область пространства сцены. Каждому *слою* в дереве (*слой* – множество узлов в дереве, имеющих одинаковое расстояние от вершины) соответствует одна из версий сцены. Для любой вершины, её потомки – это ячейки, полученные путём разбиения данной ячейки на 8 подъячеек. Ячейка каждого следующего уровня занимает 1/8 объёма (в пространстве сцены) родительской ячейки. *Сложность ячеек* может изменяться от уровня к уровню по-разному, в зависимости от содержания сцены. Будем считать, что отношение *сложности ячеек* каждого слоя содержит в 2^D раз меньше информации, чем с предыдущего слоя и будем называть D - *размерностью HC*. Под *сложностью ячеек* будем подразумевать количество элементарных элементов сцены, так, для случая представлений ячеек в виде бинарно-вольюметрического октодерева (БВО) [5], оно будет равно просто количеству непустых вокселов в данном 3D детальном уровне. Введенная нами размерность HC, имеет непосредственную связь с наиболее популярным определением фрактальной размерности Колмогорова-Хаусдорфа [12], по которой, размерность множества в *n*-мерном фазовом пространстве, равна:

$$D = \lim_{\varepsilon \to 0} \frac{\ln(M(\varepsilon))}{\ln(\varepsilon^{-1})}$$

где $M(\varepsilon)$ - минимальное число *n*-мерных кубиков с ребром ε, необходимых для покрытия множества. В нашем случае, размерность пространства равна трем, а ребро куба равно ребру соответствующего воксела на данном разрешении. Например, при приближении БВО известной тестовой модели “Bunny”, получим:

Таблица 1 Размерность БВО для модели "Bunny"

Уровень БВО	Размер ребра	Количество кубов	Приближенная Хаусдорфова размерность
7	1/128	15100	1,983
8	1/256	61010	1,987
9	1/512	252388	1,993
10	1/1024	1034937	1,998

Аналогичные результаты были получены и для других моделей, полученных из полигонов? Таким образом, мы практически подтвердили, что размерность HC Хаусдорфа действительно стремится к 2-м при измельчении разбиения. Модели, полученные из других источников, могут иметь и другие размерности. Так, размерность точечных моделей при увеличении будет стремиться к нулевой, а объёмных представлений - к трем. Понятие размерности пригодится нам далее для оценки количества реализаций сцены.

2.1 Реализации HC

Реализация HC – множество ячеек HC, обладающих следующим свойством: любой путь от корня к листу HC проходит ровно через одну из этих ячеек. Каждый уровень детализации в HC дереве является частным случаем реализации. С точки зрения сцены это означает, что все ячейки реализации не пересекаются в пространстве и при объединении представляют сцену с различными уровнями детализации. Ставится задача определения оптимальной реализации сцены при заданном критерии оптимальности. Оценим кол-во возможных реализаций $P_k(n)$. Для полного k-ичного дерева глубины n, оно выражается следующей рекуррентной формулой:

$$\begin{cases} P_k(1) = 1, \\ P_k(m+1) = P_k(m)^k + 1\ ,\ m = 2\ldots n, \end{cases}$$

Хотя полное НС является 8-ичным, но в реальности, НС имеет в среднем только порядка 4-х ветвлений. Это объясняется тем, что Хаусдорфова размерность дерева, представляющего поверхность, близка к двум, а не к трем. Но даже для такого неполного дерева количество реализаций уже для случая часто используемого 7-ми уровнего НС достигает порядка миллиарда, и определить перебором оптимальную реализацию при сохранении интерактивности не представляется возможным. Поэтому, далее рассмотрим не переборные методы решение этой задачи при заданном критерии оптимальности.

Для примера, приведем типичную эволюцию реализаций при прогрессивной визуализации сцены с изменением положения наблюдателя:

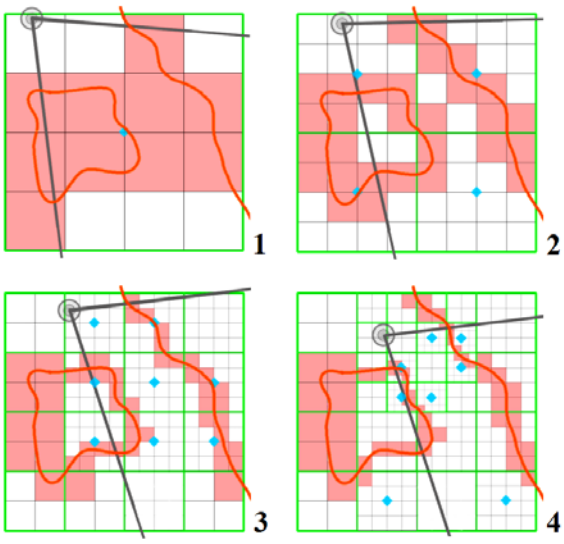

Рисунок 1 НС Реализации сцены в различные моменты времени. Зелёные границы – границы ячеек, серые границы – границы отдельных вокселей, красные квадраты – единичные воксели, белые квадраты – пустое пространство. Красные линии – реальные границы объекта.

На этом примере показана прогрессивная передача сцены, когда вначале пользователь видит наиболее грубую версию сцены, а далее в процессе получения новых ячеек, детализация сцены постепенно улучшается. Так же, видно, что отдельные участки сцены передаются в разной периодичности, в зависимости от положения наблюдателя. А именно, детальность объектов, попадающих в область видимости пользователя, увеличивается, при чем ближе объекты, тем детальность выше. Это объясняется тем, что реализация подбирается так, чтобы сделать проекции ячеек на экране сопоставимыми по размеру, в этом случае достигается квази-оптимальный баланс между количеством ячеек и детальностью получаемого после рендеринга изображения. Каким именно образом определить критерий оптимальности реализации и как его достичь, рассказывается в следующих главах.

2.2 Первичный отбор реализаций

Первичный отбор реализаций заключается в построении такого НС поддерева, который точно включает в себя оптимальную реализацию, но отбрасывает большое количество заведомо не оптимальных. Для построения такого поддерева воспользуемся двумя условиями на оптимальность реализации:

- оптимальная реализация не содержит ячеек с качеством выше допустимого
- оптимальная реализация содержит невидимые для данного положения наблюдателя ячейки с минимальным качеством

Введем функцию, вычисляющую качество визуализации ячейки для данного положении пользователя. Оценка качества должна удовлетворять следующему условию: если ячейка B – потомок ячейки A в НС, то качество B больше A для любого положения пользователя. Это требование автоматически следует из метода создания НС, по которому B более детализовано, чем A. Введем параметр α - максимально допустимое качество визуализации ячейки на клиенте. Это ограничение естественно вытекает, например, из ограниченности разрешения экрана и не способности визуализировать детали меньшие, чем размеры пиксела. Тогда из дерева можно будет выкинуть все ячейки со слишком высоким качеством без изменения качества картинки у пользователя:

$f_\alpha(A) = \{E \in A \mid G(E) < \alpha\}$ – поддерево A, состоящее из ячеек с качеством не выше заданного, где α – наилучшее возможное качество на клиенте, а G - критерий качества. Полученное множество ячеек будет деревом в силу условия на критерий качества. Далее удалим все ячейки, целиком содержащиеся вне области видимости текущего положения наблюдателя. Таким образом, мы получили НС поддерево, содержащие в себе оптимальную реализацию, и далее опишем метод поиска оптимальной реализации уже только в данном НС поддереве.

2.3 Критерии оптимальности реализации

Будем выбирать оптимальную реализацию исходя из следующего критерия:

$$Q(G(C_0), G(C_1), \ldots, G(C_n)) \rightarrow \max$$

$$V(C_0) + V(C_1) + \ldots + V(C_n) < \text{const},$$

где C_k – ячейки реализации,

G – критерий качества для отдельной ячейки

Q – оценка качества реализации по качеству её ячеек,

$V(C_k)$ – характеристика сложности визуализации ячеек:

$\sum V(C_k)$ – общая сложность реализации

Рассмотрим следующий пример критерия оптимальной реализации:

$$G(C_k) = P^{q-n}d,$$

где P – параметр, задающийся при задании сцены, $P > 1$

d – минимальное расстояние от наблюдателя до объектов ячейки C_k

q – номер слоя в НС (n – слой листовых ячеек, 0 – корень)

Этот критерий подходит, т.к. если A – потомок B, то расстояние в сцене до A не меньше расстояния до B, а q увеличится на 1 (P^{q-n} увеличится в $P>1$ раз). Если рассмотреть бинарные воксельные модели и предположить, что размер вокселя на экране определяется только расстоянием до него, то 1/G пропорционально максимальному размеру вокселя на экране.

$Q(x_0, x_1, \dots, x_n) = \min(x_0, x_1, \dots, x_n)$, т.е. мы пытаемся сделать как можно меньше максимальный видимый воксел.

$V(C_k) = \frac{S_0}{2^q P^{(n-q)\delta}}$ – средний размер ячейки на q-ом уровне, если C_k попадает в пирамиду видимости и 0 в противном случае.

Для такого критерия Q, предлагается следующий алгоритм поиска оптимальной реализации.

Для каждой ячейки C_k подсчитаем $G(C_k)$. Все ячейки поместим в линейный массив и отсортируем его по убыванию параметра G. В качестве начальной реализации R возьмем все листовые ячейки НС. Заметим, что последний элемент построенного массива всегда будет листовым. Пока не будет выполняться условие на $\sum V(C_k)$, будем выполнять операцию $S(C_n, R)$ и исключать из массива последний элемент C_n. После выполнения условия на $\sum V(C_k)$, получим оптимальную реализацию по построению.

3. ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ НС

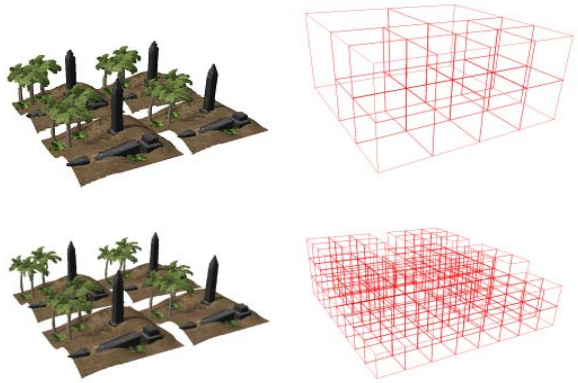

Рисунок 2 Различные реализации НС для случая разных параметров на допустимую сложность сцены. Соответственно на 100 и 450 тысяч точек.

В качестве формата представления сцены использовалось *октоизображение*, в геометрической основе которого лежит *БВО*(*бинарно-воксельное октодерево*х) с информацией о цвете в каждой вершине [10]. Каждый следующий уровень сцены получался путём уменьшения разрешения модели в 2 раза (уменьшения высоты БВО на 1), т.е. P = 2. Полная сцена хранится на сервере, на клиенте хранится лишь её часть. На каждом кадре клиент пересылает на сервер описание положения пользователя, сервер же возвращает описание оптимальной реализации для данного положения пользователя (и недостающие ячейки из этой реализации). Основная цель такой схемы работы – обеспечить постоянную скорость рендеринга сцены. Предполагается, что все воксели рисуются с одинаковой скоростью независимо от их размера на экране, т.е. что скорость рендеринга пропорционально количеству выводимых вокселей. Клиент задаёт наибольшее кол-во вокселей, которое он готов нарисовать на каждом кадре. Причём разница в положении камеры между клиентом и сервером не учитывается, т.е. получаем следующую формулу для V:

$$\begin{cases} \text{Кол-во вокселей в ячейке, если она видима} \\ 0, \text{ в противном случае} \end{cases}$$

Для обеспечения прогрессивной загрузки надо добавить к условию $\sum V(C_k)$ условие на количество передаваемой информации за один раз $\sum S(C_k) < W$, где S – размер (в байтах) ячейки, W – константа, зависящая от ширины канала между сервером и клиентом.

4. ЗАКЛЮЧЕНИЕ

В статье изложен базовый принцип построения НС и поиска его оптимальных реализаций. Реализована программа, реализующую управлением реализациями НС в клиент-серверной архитектуре, с возможностью подключения КПК в качестве клиента.

Планируется обобщить понятие реализации для случая, когда каждая ячейка может быть представлена в различном виде, не только БВО, но и например в виде полигонов или просто изображений.

Выражаем благодарность институту передовых технологий Самсунг (SAIT) за поддержку данной работы.

5. СПИСОК ЛИТЕРАТУРЫ

[1]. Hoppe, H. Progressive Meshes. Proc. SIGGRAPH, 1996.

[2]. Puppo, E., Scopigno, R.: Simplification, LOD and Multiresolution Principals and Applications. In: EUROGRAPHICS 97 Tutorial Notes, 1997.

[3]. Shade, J., Lischinski, D., Salesin, D. H., DeRose, T., Snyder, J.: Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments. SIGGRAPH 96 Proceedings.

[4]. Rusinkiewicz, S., Levoy, M.: Qsplat: A Multiresolution Point Rendering System for Large Meshes. SIGGRAPH 2000 Proceedings.

[5]. Chun-Fa Chang, Gary Bishop. A Hierarchical Representation for Image-based Rendering. SIGGRAPH 1999, Computer Graphics Proceedings

[6]. Y. Bayakovski, L. Levkovich-Maslyuk, A. Ignatenko, A. Konushin, D. Timasov, A. Zhirkov, Mahnjin Han, In Kyu Park. Depth Image-Based Representations For Static And Animated Objects. ICIP'02

[7]. M. Wand and W. Straber. Multi-Resolution Rendering of Complex Animated Scenes. EG 2002 Proceedings

[8]. Szymon Rusinkiewicz, Marc Levoy. Streaming QSplat: A Viewer for Networked Visualization of Large, Dense Models. Symposium on Interactive 3D Graphics, 2001.

[9]. Eyal Teler, Dani Lischinski. Streaming of Complex 3D Scenes for Remote Walkthroughs. EG 2001 Proceedings

[10]. Alexander Zhirkov. Binary Volumetric Octree Representation For Image-Based Rendering. Graphicon'2001.

[11]. Alexander Zhirkov. View-Dependent Octree Image Rendering. Graphicon'2003.

[12]. Hausdorff G. Dimension und auberes Mab. Math. Ann. **79**, 157-179 (1919).

Об авторах

Александр Олегович Жирков –аспирант факультета вычислительной математики и кибернетики Московского Государственного Университета им. М.В. Ломоносова

Адрес: Москва, 119899, Воробьевы горы, МГУ, 2-й учебный корпус, факультет ВМиК, кафедра автоматизации систем вычислительных комплексов

E-mail: azh@graphics.cs.msu.ru

Александр Евгеньевич Паршин – студент пятого курса факультета вычислительной математики и кибернетики Московского Государственного Университета им. М.В. Ломоносова

Адрес: Москва, 119899, Воробьевы горы, МГУ, 2-й учебный корпус, факультет ВМиК, кафедра автоматизации систем вычислительных комплексов

E-mail: savo@rambler.ru

Adaptive rendering of three dimension scenes using hierarchical cells structure

Abstract

In this article discussed representation of scenes in the form of hierarchical cells (HC). HC realization is defined; operations on realization and criterion of optimal realization are discussed. Maintaining constant rendering speed, this approach allows us to control adaptively sampling density of different scene parts depending on set of factors such as viewer position and view direction, its movement characteristics, graphic accelerator capabilities, communication bandwidth, etc. The example of using HC structure for rendering octreeimage scenes is considering.

Keywords: *HC, Hierarchical Cells, DIBR, octreeimage, adaptive rendering, progressive transmission*.

About the author(s)

Alexander Zhirkov is a Ph.D student at Moscow State University, Department of Computer Science. His contact email is azh@graphics.cs.msu.ru

Alexander Parshin is a student at Moscow State University, Department of Computer Science. His contact email is savo@rambler.ru