УДК 004.925.84 doi: 10.26102/2310-6018/2019.24.1.019

В. Д. Шакаев, А.Г.Кравец

СПОСОБЫ ПРЕДСТАВЛЕНИЯ ВОКСЕЛЬНОГО ЛАНДШАФТА ПРИ ПРОЕКТИРОВАНИИ СИСТЕМ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ

ФГБОУ ВО Волгоградский государственный технический университет
Волгоград, Россия

В статье исследованы способы представления и форматы хранения объёмных (воксельных) данных, которые могут быть использованы для моделирования воксельных ландшафтов с реконструкцией острых углов и рёбер, что необходимо для представления искусственных элементов ландшафта в САПР для архитектуры и строительства и в системах виртуальной реальности с разрушаемым окружением. Для хранения редактируемых блоков воксельного ландшафта предложено использовать лучевое представление, дополненное материалами и нормалями к поверхности, и точечное представление с неявной связанностью. В первой форме представления трёхмерный объект описывается как набор сплошных интервалов вдоль трёх координатных осей. Во второй объект представлен трёхмерным массивом вокселей, являющихся идентификаторами материалов, и облаком точек, существующих внутри ячеек, находящихся на границах раздела областей с различными материалами. Оба вида представления позволяют выполнять булевы операции, поддерживают различные материалы и содержат информацию для реконструкции острых углов поверхности, а лучевое представление применяется в настоящее время для геометрического моделирования в CAD/CAM/CAE-системах. Проведено экспериментальное тестирование предложенных способов представления и методов хранения блоков воксельного ландшафта, отмечены их преимущества и ограничения. Предложены рекомендации по выбору методов, наиболее подходящих для решения конкретных классов задач САПР. Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-07-01200.

Ключевые слова: объёмные данные, воксель, ландшафт, геометрическое моделирование, полигональная сетка, триангуляция.

Введение

В настоящее время объёмное или воксельное представление рассматривается, как одно из самых перспективных представлений для трёхмерных ландшафтов в таких приложениях как системы виртуальной реальности (ВР), геоинформационные системы, военные и аэрокосмические тренажёры, архитектурные и ландшафтные редакторы, видеоигры с большими открытыми пространствами. Воксельные ландшафты в силу ряда преимуществ (возможность моделировать пещеры и «нависающие» части, а также внутреннюю геологическую структуру, расширенные возможности процедурной генерации, робастность при редактировании) всё чаще применяются вместо традиционных ландшафтов

на картах высот [1, 2]. В последние годы в связи с ростом мощности компьютеров появилась возможность использовать воксельные данные для представления искусственных элементов ландшафта, таких как здания и крупные технические объекты [3]. Результатом является комплексное решение, сочетающее в себе представление данных о рельефе ландшафта и информацию для реконструкции острых рёбер и конических вершин поверхности, необходимую для визуализации искусственных объектов.

Для интерактивной визуализации воксельного ландшафта, как правило, сперва строится поверхностная аппроксимация в виде треугольной сетки, которая затем прорисовывается с помощью графического оборудования. Для триангуляции воксельного ландшафта с наличием искусственных объектов требуются методы извлечения изоповерхностей [4], способные восстанавливать её острые углы и рёбра, такие как метод дуальных контуров (Dual Contouring, DC) [5], EMC [6], DMC [7], CMS [8] и другие.

В настоящей статье рассматриваются смешанные, гибридные «воксельно-геометрические» представления, в которых, помимо трёхмерных скалярных данных, также содержатся сведения и об особенностях поверхности на границе объекта.

Материалы и методы

Для использования в вышеуказанных приложениях формат хранения воксельных ландшафтов должен удовлетворять следующим требованиям:

1) возможность разбиения ландшафта на блоки (для редактирования, сохранения, загрузки и визуализации только необходимых частей ландшафта);
2) высокая скорость построения адаптивной триангуляции;
3) возможность хранения информации для восстановления острых рёбер и углов поверхности, что необходимо для отображения искусственных элементов ландшафта, таких как здания и крупные технические объекты;
4) поддержка различных материалов объектов (для описания составных сред);
5) компактность представления в памяти;
6) высокая скорость сжатия (компрессии) и распаковки (для обеспечения интерактивности при редактировании ландшафта);
7) высокая степень сжатия (для компактности хранения в базе данных);
8) возможность генерации уровней детализации (без которых немыслима визуализация больших и детализированных ландшафтов);
9) возможность создания из полигональных моделей (для интеграции с существующими системами).

Кроме того, для обеспечения возможности интерактивной модификации ландшафта формат хранения воксельных данных должен предоставлять:

1) возможность выполнения операций редактирования (например, закраска различными материалами, CSG-операции) ландшафта в интерактивном режиме;

2) возможность интерактивной генерации уровней детализации.

Удовлетворить всем этим требованиям одновременно трудно, поэтому существует множество способов представления и форматов хранения воксельных данных, которые строятся на основе некоторого компромисса и имеют свои преимущества и недостатки. Все описанные далее способы предназначены для хранения отдельных блоков воксельного ландшафта. Для решения подобных задач применяются:

1) Знакоопределённое поле расстояний с градиентами [9];

2) Воксельная решётка с Эрмитовыми данными [5, 6].

В ходе исследования были предложены следующие способы представления объёмных данных для проектирования воксельных ландшафтов в системах виртуальной реальности:

1) Лучевое представление с нормалями;

2) Точечное представление с неявной связностью.

Лучевое представление с нормалями

Возможно, наиболее эффективным на сегодняшний момент представлением трёхмерных сплошных тел для быстрого выполнения булевых операций является лучевое представление (Ray Representation, Ray-Rep) [10]. Для возможности реконструкции острых углов поверхности объекта лучевое представление может быть дополнено нормалями к поверхности. Полученную форму представления можно рассматривать как дальшейшее развитие Marching Intersections [11] и расширение тройного лучевого представления (Triple Ray-Rep, Tri-Dexel) [12], в которых объект кодируется интервалами вдоль трёх координатных осей. Лучевое представление с нормалями можно описать как набор «стержней», пронизывающих объект вдоль каждого координатного направления, на каждом из которых хранятся координаты точек пересечения с поверхностью объекта и единичные нормали к поверхности объекта в этих точках [13–19]. При построении лучевого представления из закрытого объекта на каждом луче будет создано чётное количество точек пересечения с поверхностью объекта — *сёрфелей* (*surfels*), или, соответственно, нечетное количество «сплошных», расположенных внутри объекта интервалов — *декселей* (*dexel*, сокр. от *depth pixel*). Для описания неоднородных, составных объектов сёрфели или дексели могут хранить индекс соответствующего материала. Визуализацию тройного лучевого

представления (далее Ray-Rep), дополненное нормалями для возможности реконструкции особенностей поверхности, можно увидеть на Рисунке 1.

Триангуляция. Как и в предыдущем формате данных, для триангуляции Ray-Rep могут быть использованы любые ячеечные методы: Marching Cubes [23], Surface Nets [24], EMC, DC, DMC, CMS и т.д. При этом рёбра кубических ячеек располагаются вдоль лучей Ray-Rep, а центры вокселей (или углы ячеек) находятся на пересечениях лучей. Неоднозначности типа «внутри»/«снаружи» разрешаются «голосом большинства» (majority voting): центр вокселя считается расположенным внутри объекта, если он находится на пересечении как минимум двух декселей (интервалов, расположенных полностью внутри объекта).

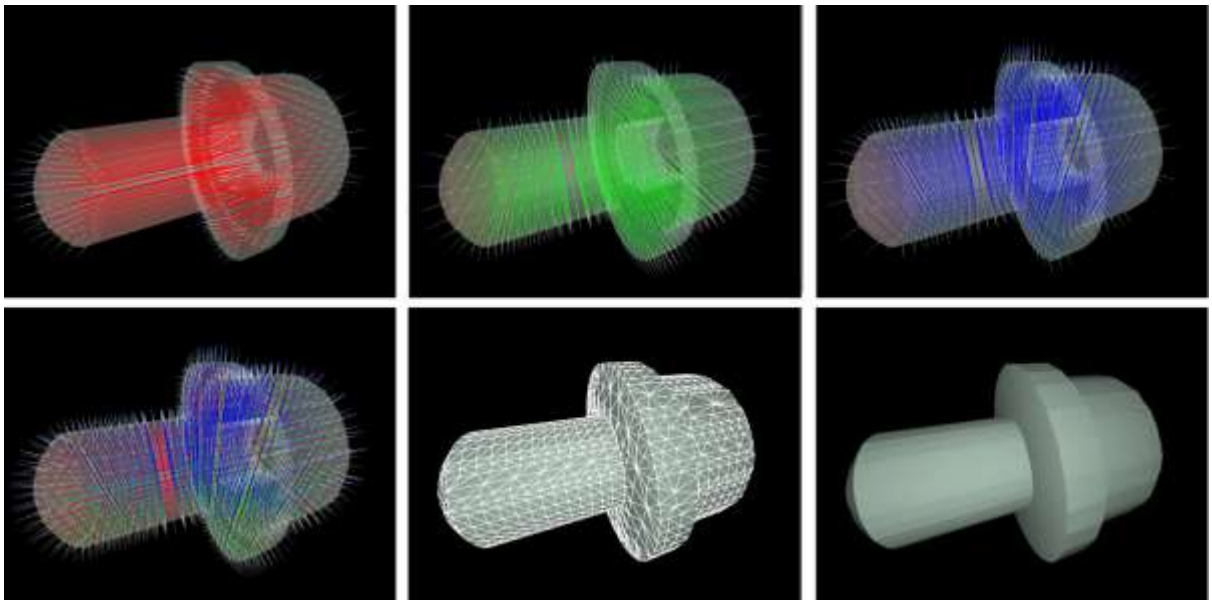

Рисунок 1 – Тройное лучевое представление с нормалями и реконструированная поверхность. Трёхмерный объект представлен набором интервалов вдоль трёх координатных направлений, на концах интервалов хранятся точки пересечения с поверхностью тела и единичные нормали.

В отличие от воксельной решётки, для триангуляции которой необходимо рассмотреть все ячейки, Ray-Rep позволяет быстро отбросить пустые участки пространства и идентифицировать только ячейки, пересекающие поверхность. Пропуск пустого пространства может применяться для ускорения адаптивных методов триангуляция, в которых иерархическая структура строится сверху вниз на основе эвристических правил. При этом в качестве структуры разбиения пространства более предпочтительными, чем октодеревья, являются *k*D-деревья, поскольку они лучше адаптируются к особенностям поверхности.

Способы создания. Ray-Rep может быть создано теми же способами, что воксельные решётки с Эрмитовыми данными: в основе получения Ray-Rep лежит операция пересечения луча с поверхностью.

Среди всех разработанных форматов трёхмерных данных, Ray-Rep обладает самой высокой скоростью построения из полигональных моделей. Ray-Rep могут быть получены с любой заданной точностью с помощью трассировки лучей или растеризации полигональной сетки.

В первом случае для создания Ray-Rep с разрешением n вокселей вдоль каждого координатного направления бросается $n \times n$ лучей, проходящих через центры вокселей (и вдоль рёбер ячеек решётки разбиения). На каждом луче записываются все точки пересечения луча с поверхностью объекта и единичные нормали к поверхности. Нечётное число точек пересечения указывает на ошибки в исходной модели или наличие в ней слишком мелких деталей. Расхождения в классификации вокселей разрешаются «голосом большинства» [20].

Второй подход предпочтителен для использования на GPU и позволяет почти в интерактивном режиме мультипроходным рендерингом (Depth Peeling) [13, 14] или более современным методом Order Independent Translucency (OIT) [21] получать детальные Ray-Rep из массивных полигональных моделей. После создания Ray-Rep для исправления неоднозначностей, возникающих из-за избыточности Ray-Rep, рекомендуется выполнять регуляризацию [17, 18].

Редактирование. Лучевое представление позволяет просто и эффективно реализовать операции конструктивной сплошной геометрии (CSG), офсеттинга, построения суммы и разницы Минковского. На GPU процессы создания Ray-Rep из полигональной модели, выполнения вышеперечисленных операций и триангуляции полученного Ray-Rep могут быть выполнены в реальном времени даже для Ray-Rep с разрешениями 257-513 лучей (или вокселей). Авторская реализация на CPU позволяет в интерактивном режиме выполнять серии булевых операций над Ray-Rep с разрешением в 65 лучей.

Булевы операции над Ray-Rep сводятся к операциям над лучевыми сегментами: слияние, сортировка и удаление нулевых областей [14–16, 18, 19]. В результате редактирования может нарушиться целостность Ray-Rep, поэтому для поддержания согласованности необходимо выполнять *регуляризацию* [17, 18]: удаление декселей с длиной меньше ребра ячейки, удаление или слияние изолированных декселей для исправления неоднозначностей и поддержания не более одной точки пересечения на рёбрах ячеек, поддержание чётного количества точек пересечения на каждом луче в замкнутой модели и т.д.

Сжатие. Ray-Rep является компактной, разреженной структурой данных, обладающей высокой избыточностью и не предназначенной для долговременного хранения и сжатие. Для снижения объёма рабочей памяти в авторской реализации Ray-Rep расстояния вдоль лучей огрубляются до 16-битных целых чисел, а нормали к поверхности в точках

пересечения лучей с границей объекта хранятся с полной точностью (по три числа типа float на каждую нормаль).

Упрощение. Ray-Rep не предназначены для упрощения. Обычно Ray-Rep строится из полигональной модели для быстрого выполнения булевых операций и затем «выбрасывается». Генерация уровней детализации, сжатие и хранение Ray-Rep нецелесообразны.

Точечные представления с неявной связностью

Среди предложенных способов представлений для воксельных ландшафтов самой широкой функциональностью обладают точечные представления с неявной связностью (Point with Implicit Connectivity, PIC) [22]. В данном представлении 3D-объект хранится, как воксельная решётка и множество приграничных ячеек с соответствующими им точками, лежащими на поверхности объекта, для которых не заданы в явном виде понятия топологии связей или непрерывной поверхности. Объёмные данные организованы в готовом виде для создания поверхностной триангуляции каким-либо двойственным методом (например, DC или DMC). Точки становятся вершинами полигональной сетки, а топология их связи (информация о смежности, необходимая для построения сетки) неявно следует из воксельных данных: из знаков (значений «внутри»/«снаружи» в случае бинарной воксельной решётки) или индексов материалов, известных в углах каждой ячейки.

Эрмитовы данные также не хранятся в PIC в явном виде, а вычисляются в случае необходимости (например, перед выполнением операций редактирования). Для получения Эрмитовых данных выбранным методом триангуляции создаётся фрагмент полигональной сетки — частичный контур (partial contour), который затем пересекается с (активными) рёбрами, для которых требуется найти точки пересечения с поверхностью и единичные нормали к поверхности в этих точках.

Регулярное разбиение. PIC на регулярной решётке формально можно описать в виде воксельной решётки V и облака точек S. Каждый блок воксельного ландшафта имеет форму куба, на котором задана равномерная декартова сетка C, состоящая из n^3 ячеек. Воксельная решётка V задана набором значений вокселей (индексов подобластей или материалов) в узлах сетки C (в углах ячеек $c_{i,j,k}$):

$$V = \left\{ v_{i,j,k} \in \mathbb{Z} \mid \; i,j,k \in \mathbb{N},\; 0 \le i,j,k \le n \right\}.$$

Облако точек S задано множеством вершин, которые существуют только внутри неоднородных (с наличием различных материалов или знаков) ячеек сетки C (т.е. внутри ячеек, углы которых не имеют совпадающие материалы или знаки):

$$S = \left\{ \mathbf{p}_{i,j,k} \in \mathbb{R}^3 \mid i,j,k \in \mathbb{Z},\; 0 \le i,j,k < n \wedge c_{i,j,k} \text{ — неоднородная ячейка} \right\}.$$

Позиции вершин $\mathbf{p}_{i,j,k}$ помещаются вблизи острых углов поверхности, представляющей границу раздела между различными материалами. Если для триангуляции используется метод DC, то внутри каждой граничной ячейки будет создано по одной острой вершине, если DMC, то может быть создано от одной до четырёх острых вершин в зависимости от знаковой конфигурации ячейки.

Таким образом, каждый блок воксельного ландшафта в формате PIC хранится, как «плотный» 3D-массив вокселей и разреженный 3D-массив точек, которые становятся вершинами поверхностной сетки. В целях экономии памяти позиции острых вершин квантуются относительно границ (AABB) ячейки (достаточно 8 бит на каждую координатную компоненту).

Адаптивное разбиение. В [22] предложено адаптивное PIC на основе знакоопределённого октодерева, которое для возможности выполнения CSG-операций помимо «серых» листовых узлов (пересекающих поверхность объекта) может содержать «чёрные» листовые узлы (расположенные полностью внутри объекта) и «белые» узлы (расположенные полностью снаружи объекта). Каждый серый листовой узел-ячейка содержит позицию острой вершины, лежащей на поверхности внутри ячейки, знаки (значения «внутри» и «снаружи») в углах ячейки, и время последнего изменения (для построения частичного контура при выполнении CSG-операций). Использование октодерева приводит к дублированию знаков (индексов материалов) в углах смежных ячеек.

Триангуляция PIC сводится к соединению вершин, позиции которых уже известны и хранятся явно, в отличие от всех других рассмотренных форматов объёмных данных. Для соединения вершин можно использовать любой ячеечный двойственный метод триангуляции (DC, DMC, Closest Point Contouring и т.д.). Поскольку двойственные методы триангуляции обладают особенностью inter-cell dependency, при генерации бесшовной сетки, соединяющей каждый блок ландшафта с его соседями, необходимо включать в триангуляцию прилегающие к нему приграничные ячейки с соседних блоков. Во избежание обращений к внешней памяти и возможности распараллеливания процесса триангуляции на практике обычно применяется концепция «призрачных» ячеек (ghost cells), которая заключается в добавлении слоёв фиктивных ячеек для дублирования данных, принадлежащих другим блокам. (Призрачные ячейки физически расположены внутри соседнего блока.) Другими словами, для каждого блока в памяти хранятся, помимо собственных данных, также копии прилегающих к нему ячеек с соседних блоков. Образованные призрачными ячейками полигоны служат только для соблюдения граничных условий (исключения разрывов полигональной сетки и расчёта «гладких» вершинных нормалей) и не должны включаться

в конечную полигональную сетку. Для предотвращения разрывов достаточно перекрытия смежных блоков на слой толщиной в одну ячейку, для вычисления гладких вершинных нормалей необходимо перекрывать блоки на слой толщиной в две ячейки.

Редактирование. PIC позволяет эффективно выполнять булевы операции (CSG). Для необходимой части блока двойственным алгоритмом триангуляции (DC или DMC) строится фрагмент полигональной сетки и ищутся его пересечения с активными рёбрами ячеек для получения Эрмитовых данных. Затем над воксельной решёткой (точками) и Эрмитовыми данными (отрезками) выполняются CSG-операции, как описано в разделе 3.3, после чего вычисляются новые позиции острых вершин. В целом, даже с квантованными до восьми бит позициями вершин PIC обеспечивает довольно высокое качество реконструкции острых углов после выполнения над ландшафтом булевых операций.

PIC является единственным из предложенных форматов, поддерживающим операции сглаживания поверхности ландшафта (что может потребоваться, например, для сглаживания нежелательных острых углов и рёбер ландшафта или для улучшения качества треугольников в построенной сетки). Сглаживание поверхности выполняется путём сдвигания позиции вершины активной ячейки, затронутой операцией сглаживания, в центр масс вершин из соседних активных ячеек. Для получения хороших результатов для усреднения достаточно брать вершины тех соседних ячеек, которые имеют с текущей ячейкой общую активную грань (ячейка кубической формы может иметь до шести активных граней).

После выполнения операций редактирования перед триангуляцией блоков необходимо обновить призрачные ячейки всех затронутых блоков (halo exchange), что усложняет реализацию и затрудняет распараллеливание системы.

Результаты

Эксперименты по оценке эффективности предложенных форматов хранения для воксельного ландшафта проводились на компьютере, оснащённом процессором Intel® Core™ i7-2600K @ 3.40 ГГц и 16 Гб оперативной памяти.

В эксперименте оценивались скорость выполнения булевых операций и качество восстановления острых рёбер и углов поверхности.

Тестовая сцена построена из неявного описания: разности концентрической синусоидальной волны, заданной уравнением $\sin\left(\sqrt{x^2 + y^2}\right) = z$, и куба. Сцена состоит из 8^3 блоков, каждый блок содержит 32^3 ячеек и может иметь до четырёх уровней детализации, которые получены упрощением октодерева, построенного из данных

блока. Каждый блок хранит редактируемые воксельные данные в виде Ray-Rep с нормалями. На сцену были помещены CAD-объекты с поверхностными особенностями и плоскими и кривыми гранями: модели "Box", "Fandisk" и "Bolt", а затем к сцене в интерактивном режиме были применены булевы операции (Рисунок 2).

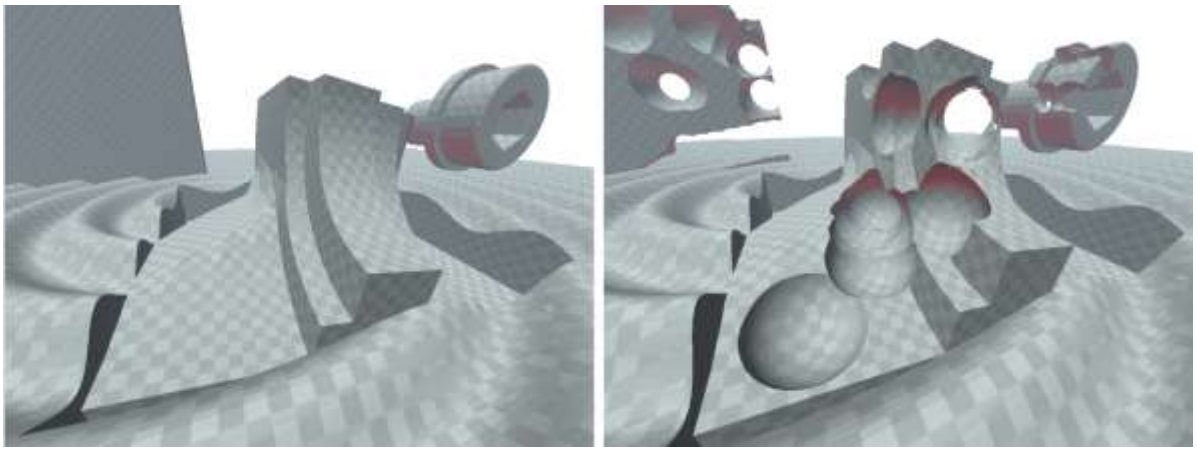

Рисунок 2 – Тестовая сцена до и после выполнения булевых операций вычитания.

Можно увидеть, что полученная сетка содержит довольно качественную реконструкцию острых углов и рёбер поверхности. Время между запросом на CSG-операцию, изменением лучевого представления, регенерацией всех уровней детализации, триангуляцией и отображением результата для каждого блока находилось в интервале 8-12 мсек. Если при редактировании было затронуто много блоков, то в процессе выполнения в фоновом потоке булевых операций и операций перестроения треугольной сетки в течение короткого времени становятся заметными швы между смежными блоками. Кроме того, из-за избыточности лучевого представления после операций редактирования возможно появление швов и разрывов на границах между смежными блоками, во избежание которых необходимо периодически синхронизировать данных смежных блоков и регуляризацию Ray-Rep.

Поэтому в качестве основного формата для хранения редактируемых воксельных данных лучше использовать точечные представления, которые обеспечивают такое же качество реконструкции особенностей поверхности (Рисунок 3), но обладают несколько меньшей скоростью выполнения CSG-операций. При этом для обеспечения интерактивности (при выполнении булевых операций и повторной триангуляции) оптимальный размер блока составляет 32^3 ячейки.

Результаты сравнения предложенных способов приведены в Таблице 1.

Обсуждение

Отметим преимущества и недостатки предложенных способов представления и хранения блоков воксельного ландшафта.

Преимуществами лучевого представления с нормалями являются:

– гибкость: можно без каких-либо модификаций использовать любой ячеечный (cube-based) метод триангуляции;

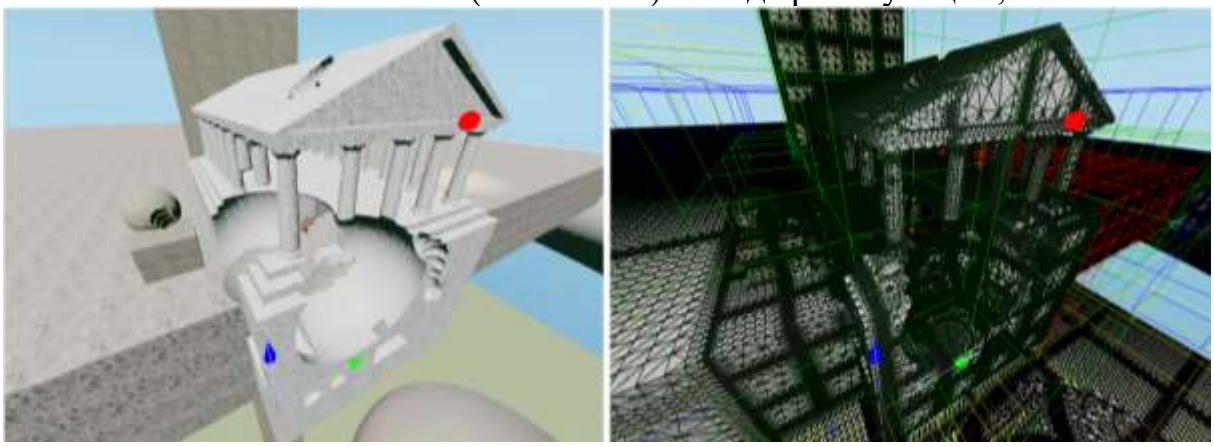

Рисунок 3 – Результаты редактирования сцены, представленной в виде точечного представления с неявной связанностью. На сцену были помещены различные модели, к которым затем были применены булевы операции вычитания. Справа показана треугольная сетка поверхности и визуализация уровней детализации.

Таблица 1 – Сравнительные характеристики предложенных способов представления блоков воксельного ландшафта

Название способа / Критерии	Лучевое представление, дополненное нормалями к поверхности	Точечное представление с неявной связностью (на регулярной решётке)
Скорость триангуляции двойственными методами	Высокая	Очень высокая
Поддержка других ячеечных методов триангуляции	Да	Нет
Объём занимаемой рабочей памяти в несжатом виде	Низкий	Средний
Объём занимаемого места в сжатом виде (при хранении во внешней памяти)	Средний	Низкий
Разреженная форма представления	Да	Нет
Скорость выполнения булевых операций	Очень высокая	Средняя
Чувствительность к ошибкам округления	Да	Нет
Необходимость в регуляризации после редактирования	Да	Нет
Поддержка операции сглаживания	Нет	Да
Сложность реализации	Низкая	Высокая
Скорость создания из закрытых полигональных моделей	Очень высокая	Низкая
Параллелизуемость операций редактирования	Да	Нет

- очень простая и регулярная структура, допускающая параллельную реализацию как на CPU, так и на GPU;
- простота и высокая скорость операций редактирования (самая высокая скорость выполнения булевых операций среди всех разработанных форматов);
- высокая скорость создания из большинства типов входных данных (самая высокая скорость построения из сложных полигональных моделей);
- высокое качество реконструкции острых рёбер и углов;
- возможность хранения дополнительных данных в точках пересечения рёбер ячеек с поверхностью (например, текстурных координат, материалов);
- низкое потребление рабочей памяти из-за разреженной структуры данных;
- высокая скорость построения адаптивной триангуляции за счёт пропуска пустых областей.

Недостатки представления Ray-Rep вытекают в первую очередь из-за его избыточности:

- неоднозначности в классификации вокселей;
- необходимость в регуляризации для поддержания целостности и исправления неоднозначностей после создания или редактирования;
- высокая чувствительность к ошибкам округления;
- отсутствие поддержки операций сглаживания;
- при хранении объём занимаемого пространства выше, чем у других форматов объёмных данных, которые могут сжиматься в десятки раз.

Технологические рекомендации применения представления Ray-Rep для проектирования систем ВР. Лучевое представление обладает избыточностью и поэтому нуждается в регуляризации. Поэтому отдельные блоки воксельного ландшафта целесообразно хранить в виде треугольных сеток и конвертировать в Ray-Rep и обратно при необходимости (современное графическое оборудование позволяет выполнять все эти операции «на лету»). Для представления объёмного ландшафта в лучевой форме с минимальной избыточностью может быть использовано только одно координатное направление (например, вертикальное). Реализация Ray-Rep на GPU получила название многослойных карт глубины с нормалями (Layered Depth-Normal Images, LDNI) [13], поскольку до появления технологии GPGPU для хранения глубины пикселя (расстояние от опорной плоскости до точки пересечения с поверхностью) и нормали использовались массивы текстур, как в многослойных картах глубины

(Layered Depth Images, LDI). С появлением технологий CUDA, OpenCL и вычислительных шейдеров стало возможным на GPU строить списки точек для каждого фрагмента изображения [21]. В [19] лучевое представление с нормалями названо Z-list-buffer.

Преимущества точечных представлений с неявной связностью:

- высокая скорость триангуляции двойственными методами: позиции острых вершин внутри ячеек уже вычислены, и остаётся только их соединить;
- полный контроль над позициями вершин в ячейках, что позволяет легко реализовать операцию сглаживания (smoothing) ландшафта и предоставляет возможность корректировки вершин для повышения качества построенной сетки; после выполнения операций редактирования при триангуляции не требуется заново вычислять позиции острых вершин в незатронутых ячейках блоках;
- высокое качество восстановления острых рёбер и углов;
- встроенная поддержка составных, мультиматериальных областей на регулярной решётке (в адаптивном варианте возникает дублирование данных);
- возможность хранения дополнительных данных в острых вершинах (например, коэффициент гладкости, «затенение» точки (ambient occlusion));
- снижение вычислительной сложности операций редактирования и триангуляции за счёт пропуска однородных частей пространства в адаптивном варианте PIC;
- низкое потребление памяти в распакованном, несжатом виде (данный формат — самый компактный из остальных предложенных, например, в PIC для каждой активной ячейки достаточно хранить позицию острой вершины против трёх векторов (единичных нормалей) и трёх скаляров (направленных расстояний) в воксельной решётке с Эрмитовыми данными);
- высокая степень сжатия (для сжатия позиций вершин в ячейках разумно использовать квантование и энтропийное кодирование с предсказанием).
- Недостатки точечных представлений с неявной связностью:
- данный формат может быть использован только с определённым двойственным методом триангуляции;
- усугубление проблемы inter-cell dependency: при редактировании блока требуется подгружать данные из

соседних блоков для избежания артефактов (разрывов сетки и резких переходов освещения) на границах блоков;
- сложность операций редактирования, поскольку точки пересечения рёбер ячеек с поверхностью и нормали к поверхности не хранятся в явном виде;
- низкая скорость создания: требуется нахождение острых вершин;
- необходимость синхронизации приграничных данных («призрачных» ячеек) между смежными блоками, что значительно усложняет реализацию и затрудняет распараллеливание.

Технологические рекомендации применения точечных представлений с неявной связностью для проектирования систем ВР. Среди рассмотренных форматов представления объёмов точечное представление является самым технологичным и обладает наиболее широкой функциональностью, однако создаёт зависимости между смежными блоками ландшафта (при редактировании блока могут потребоваться данные его соседей). На основе анализа свободно доступной документации и видеоматериалов можно предположить, что точечное представление на регулярной решётке используется для хранения ландшафта в Voxel Farm.

Заключение

В системах виртуальной реальности и серьёзных играх с изменяемым воксельным ландшафтом важна производительность операций редактирования, поэтому на первое место выходит лучевое представление, позволяющее обрабатывать каждый блок ландшафта независимо от его соседей.

В системах архитектурной визуализации необходимо создание сеток с качественной реконструкцией мелких деталей, острых рёбер и углов поверхности, поэтому предпочтение отдаётся адаптивным точечным представлениям и лучевым представлениям в высоком разрешении.

Оба предложенных объёмных представления подходят для применения в видеоиграх для моделирования разрушаемого окружения с поддержкой различных материалов и возможностью передавать основные особенности поверхности. Варьируя разрешение блоков воксельного ландшафта, можно добиться требуемого соотношения скорости работы и потребления рабочей памяти при редактировании и сжатии/декомпрессии блоков.

ЛИТЕРАТУРА

1. Lengyel E. Voxel-Based Terrain for Real-Time Virtual Simulations // PhD diss., University of California at Davis. — 2010. — 95 P.
2. Forstmann S. Research on Improving Methods for Visualizing Common Elements in Video Game Applications // PhD diss., Waseda University. — 2013. — 168 P.
3. Voxel Farm [Электронный ресурс] / Режим доступа: http://voxelfarm.com/ Дата последнего обращения: 29.01.2019.
4. A Survey on Implicit Surface Polygonization // ACM Computing Surveys. 2015. Vol. 47, Iss. 4. — P. 1-39.
5. Ju T., Losasso F., Schaefer S., Warren J. Dual Contouring of Hermite Data // ACM Transactions on Graphics, 21(3). — 2002. — P. 339–346.
6. Шакаев В.Д. View-Dependent Level of Detail for Real-Time Rendering of Large Isosurfaces / В.Д. Шакаев, Н.П. Садовникова, Д.С. Парыгин // Creativity in Intelligent Technologies and Data Science. Second Conference, CIT&DS 2017 – (Ser. Communications in Computer and Information Science ; Vol. 754) – P. 501-516.
7. Schaefer S, Warren J. Dual marching cubes: primal contouring of dual grids // Computer Graphics Forum, 24(2). —2005. — P. 195–201.
8. Ho C.-C., Wu F.-C., Chen B.-Y., Chuang Y.-Y., Ouhyoung M. Cubical marching squares: Adaptive feature preserving surface extraction from volume data // EUROGRAPHICS 2005, 24, 3. — 2005. — P. 537–545.
9. Frisken S.F., Perry R.N. Designing with distance fields // Proceedings of the International Conference on Shape Modeling and Applications 2005 (SMI '05), IEEE Computer Society. — 2005, Washington, DC, USA. – P. 58–59
10. Menon J., Marisa R., Zagajac J. More powerful solid modeling through ray representations // IEEE Computer Graphics and Applications, 14. — 1994. — P. 22–35.
11. Rocchini C., Cignoni P., Ganovelli F., Montani C., Pingi P., Scopigno R. Marching Intersections: an Efficient Resampling Algorithm // Shape Modeling International, IEEE Computer Society. — 2001. – P. 296–305.
12. Benouamer M.O., Michelucci D. Bridging the Gap between CSG and Brep via a Triple Ray Representation // Proceedings of the fourth ACM symposium on Solid modeling and applications (SMA '97), ACM. — 1997, New York, NY, USA. — P. 68–79.
13. Wang C.C.L., Chen Y. Layered Depth-Normal Images: a Sparse Implicit Representation of Solid Models // Technical Report, The Chinese University of Hong Kong. — 2007.

14. Wang C.C.L., Leung Y.-S., Chen Y., Solid modeling of polyhedral objects by Layered Depth-Normal Images on the GPU // Computer-Aided Design, Vol. 42, Iss. 6. — 2010. — P. 535–544.
15. Zhao H., Wang C.C.L., Chen Y., Jin X. Parallel and efficient Boolean on polygonal solids // The Visual Computer, Vol. 27, Iss. 6–8. — 2011. — P. 507–517.
16. Воронцов Г.В. Быстрое построение BVH дерева на GPGPU. /Воронцов Г.В., Преображенский А.П., Чопоров О.Н.// Моделирование, оптимизация и информационные технологии. 2018. Т. 6. № 2 (21). С. 24-34.
17. Wang C.C.L., Chen Y. Regulating complex geometries using layered depth-normal images for rapid prototyping and manufacturing // Rapid Prototyping Journal, Vol. 19, Iss. 4. — 2013. — P. 253–268.
18. Kwok T.-H., Chen Y., Wang C.C.L. Geometric Analysis and Computation Using Layered Depth-Normal Images for Three-Dimensional Microfabrication // Three-Dimensional Microfabrication Using Two-photon Polymerization (Micro and Nano Technologies), William A. Publishing. — 2016. — P. 119–147.
19. Ho C.-C., Tu C.-H., Ouhyoung M. Detail sculpting using cubical marching squares // Proceedings of the 2005 international conference on Augmented tele-existence (ICAT '05). — 2005, NY, USA. — P. 10–15.
20. Nooruddin F., Turk, G. Simplification and Repair of Polygonal Models Using Volumetric Techniques // ACM Transactions on Visualization and Computer Graphics, Vol. 9, Iss. 2. — 2003. — P. 191–205.
21. Lefebvre S. IceSL: A GPU Accelerated CSG Modeler and Slicer // AEFA'13, 18th European Forum on Additive Manufacturing. — 2013, Paris, France.
22. Zhang N., Qu H., Kaufman A. CSG Operations on Point Models with Implicit Connectivity // Computer Graphics International 2005. — 2005. — P. 87–93.
23. Lorensen W., Cline H. Marching Cubes: a high resolution 3D surface construction algorithm // Computer Graphics (SIGGRAPH 87 Proceedings). — 1987. — P. 163–169.
24. Gibson S. Constrained elastic surface nets: Generating smooth surfaces from binary segmented data // Proceedings of the First International Conference on Medical Image Computing and Computer-Assisted Intervention, MICCAI 1998. — 1998. — P. 888–898.

V. D. Shakaev, A.G. Kravets

REPRESENTATION AND STORAGE OF VOXEL TERRAIN FOR DESIGNING VIRTUAL REALITY SYSTEMS

Volgograd State Technical University (Volgograd, Russia)

In this paper, we develop methods for representing and storing volumetric (voxel) data. These methods can be used for modeling voxel terrains with sharp features that are necessary for representing human-made parts of the terrain in architectural CAD and virtual reality systems with destructible environments. We propose that each editable chunk of the voxel terrain be stored in a form of ray-representation augmented with surface normals and materials, or as a set of points with implicit connectivity. In the first case, a 3D object is encoded as a set of solid intervals along the three principal directions. In the second case, the object is described as a dense 3D array of voxels (serving as material indices), and a sparse point cloud, where points are stored only inside heterogeneous cells (where materials at corner voxels differ). Both representations allow to perform boolean operations, support multiple materials and store information for reconstructing sharp features of the surface, while the ray representation is used in CAD/CAM/CAE software for geometric modeling. We evaluate the proposed volumetric representations in a test environment and emphasize their advantages and limitations. This enables the reader to choose the best strategy according to the number of specific requirements for a CAD or a virtual reality system.The reported study was funded by RFBR according to the research project № 19-07-01200.

Keywords: volumetric data, voxel, terrain, geometric modeling, polygonal mesh, isosurface extraction.

REFERENCES

1. Lengyel E. Voxel-Based Terrain for Real-Time Virtual Simulations // PhD diss., University of California at Davis. — 2010. — 95 P.
2. Forstmann S. Research on Improving Methods for Visualizing Common Elements in Video Game Applications // PhD diss., Waseda University. — 2013. — 168 P.
3. Voxel Farm [Электронный ресурс] / Режим доступа: http://voxelfarm.com/ Access date: 29.01.2019.
4. A Survey on Implicit Surface Polygonization // ACM Computing Surveys. 2015. Vol. 47, Iss. 4. — P. 1-39.
5. Ju T., Losasso F., Schaefer S., Warren J. Dual Contouring of Hermite Data // ACM Transactions on Graphics, 21(3). — 2002. — P. 339–346.
6. Shakaev, V. View-Dependent Level of Detail for Real-Time Rendering of Large Isosurfaces / V. Shakaev, N.P. Sadovnikova, D.S. Parygin // Creativity in Intelligent Technologies and Data Science. Second Conference, CIT&DS 2017 – (Ser. Communications in Computer and Information Science ; Vol. 754) – P. 501-516.
7. Schaefer S, Warren J. Dual marching cubes: primal contouring of dual grids // Computer Graphics Forum, 24(2). —2005. — P. 195–201.

8. Ho C.-C., Wu F.-C., Chen B.-Y., Chuang Y.-Y., Ouhyoung M. Cubical marching squares: Adaptive feature preserving surface extraction from volume data // EUROGRAPHICS 2005, 24, 3. — 2005. — P. 537–545.
9. Frisken S.F., Perry R.N. Designing with distance fields // Proceedings of the International Conference on Shape Modeling and Applications 2005 (SMI '05), IEEE Computer Society. — 2005, Washington, DC, USA. – P. 58–59
10. Menon J., Marisa R., Zagajac J. More powerful solid modeling through ray representations // IEEE Computer Graphics and Applications, 14. — 1994. — P. 22–35.
11. Rocchini C., Cignoni P., Ganovelli F., Montani C., Pingi P., Scopigno R. Marching Intersections: an Efficient Resampling Algorithm // Shape Modeling International, IEEE Computer Society. — 2001. – P. 296–305.
12. Benouamer M.O., Michelucci D. Bridging the Gap between CSG and Brep via a Triple Ray Representation // Proceedings of the fourth ACM symposium on Solid modeling and applications (SMA '97), ACM. — 1997, New York, NY, USA. — P. 68–79.
13. Wang C.C.L., Chen Y. Layered Depth-Normal Images: a Sparse Implicit Representation of Solid Models // Technical Report, The Chinese University of Hong Kong. — 2007.
14. Wang C.C.L., Leung Y.-S., Chen Y., Solid modeling of polyhedral objects by Layered Depth-Normal Images on the GPU // Computer-Aided Design, Vol. 42, Iss. 6. — 2010. — P. 535–544.
15. Zhao H., Wang C.C.L., Chen Y., Jin X. Parallel and efficient Boolean on polygonal solids // The Visual Computer, Vol. 27, Iss. 6–8. — 2011. — P. 507–517.
16. Vorontsov G.V. Quickly build a BVH tree on GPGPU. / Vorontsov G.V., Preobrazhensky A.P., Choporov O.N. / / Modeling, optimization and information technology. 2018. Vol. 6. No. 2 (21). Pp. 24-34.
17. Wang C.C.L., Chen Y. Regulating complex geometries using layered depth-normal images for rapid prototyping and manufacturing // Rapid Prototyping Journal, Vol. 19, Iss. 4. — 2013. — P. 253–268.
18. Kwok T.-H., Chen Y., Wang C.C.L. Geometric Analysis and Computation Using Layered Depth-Normal Images for Three-Dimensional Microfabrication // Three-Dimensional Microfabrication Using Two-photon Polymerization (Micro and Nano Technologies), William A. Publishing. — 2016. — P. 119–147.
19. Ho C.-C., Tu C.-H., Ouhyoung M. Detail sculpting using cubical marching squares // Proceedings of the 2005 international conference on Augmented tele-existence (ICAT '05). — 2005, NY, USA. — P. 10–15.

20. Nooruddin F., Turk, G. Simplification and Repair of Polygonal Models Using Volumetric Techniques // ACM Transactions on Visualization and Computer Graphics, Vol. 9, Iss. 2. — 2003. — P. 191–205.
21. Lefebvre S. IceSL: A GPU Accelerated CSG Modeler and Slicer // AEFA'13, 18th European Forum on Additive Manufacturing. — 2013, Paris, France.
22. Zhang N., Qu H., Kaufman A. CSG Operations on Point Models with Implicit Connectivity // Computer Graphics International 2005. — 2005. — P. 87–93.
23. Lorensen W., Cline H. Marching Cubes: a high resolution 3D surface construction algorithm // Computer Graphics (SIGGRAPH 87 Proceedings). — 1987. — P. 163–169.
24. Gibson S. Constrained elastic surface nets: Generating smooth surfaces from binary segmented data // Proceedings of the First International Conference on Medical Image Computing and Computer-Assisted Intervention, MICCAI 1998.— P. 888–898.