УДК 004.932
doi: 10.21685/2587-7704-2024-9-2-11

Open Access RESEARCH ARTICLE

Использование алгоритма quad tree для разработки высокопроизводительных решений симуляции водных поверхностей

Максим Аркадьевич Писарев
Пензенский государственный университет, Россия, г. Пенза, ул. Красная, 40
Pisarevmx@gmail.com

Аркадий Петрович Писарев
Пензенский государственный университет, Россия, г. Пенза, ул. Красная, 40
Pisarev602@mail.ru

Аннотация. В статье рассматривается методы оптимизации работы графического конвейера, с использованием концепции Level of Detail и тесселяции. Разъясняется суть использования методики деревьев квадрантов для генерации сетки 3д объектов. Демонстрируется практическая реализация тесселяции на языке C++ при моделировании водной поверхности в среде Unreal Engine.

Ключевые слова: графический конвейер, Level of Detail, тесселяция, триангуляция, шейдеры, Unreal, деревья квадрантов

Для цитирования: Писарев М. А., Писарев А. П. Использование алгоритма quad tree для разработки высокопроизводительных решений симуляции водных поверхностей // Инжиниринг и технологии. 2024. Т. 9 (2). С. 1–4. doi: 10.21685/2587-7704-2024-9-2-11

Using the quad tree algorithm in high-performance water surface simulation solutions development

Maxim A. Pisarev
Penza State University, 40 Krasnaya Street, Penza, Russia
Pisarevmx@gmail.com

Arkady P. Pisarev
Penza State University, 40 Krasnaya Street, Penza, Russia
Pisarev602@mail.ru

Abstract. The article discusses methods for optimizing the operation of the graphics pipeline, using the concept of Level of Detail and tessellation. The essence of using the quadrant tree technique to generate a grid of 3d objects is explained. Demonstrated practical implementation of tessellation in the C++ on water surface example in Unreal Engine.

Keywords: graphics pipeline, Level of Detail, tessellation, triangulation, shaders, unreal, quadtree

For citation: Pisarev M.A., Pisarev A.P. Using the quad tree algorithm in high-performance water surface simulation solutions development. *Inzhiniring i tekhnologii = Engineering and Technology*. 2024;9(2):1–4. (In Russ.). doi: 10.21685/2587-7704-2024-9-2-11

С момента появления рендеринга в реальном времени исследователи и разработчики сталкиваются с рядом сложных задач, связанных с достижением баланса между высокой степени детализации и производительности решений. Задачи моделирования и визуализации различных объектов реального мира актуальны в различных системах, в том числе в системах медицинского назначения [1]. Требуется, с одной стороны, обрабатывать и отображать объекты с высокой степенью реалистичности [2]. С другой стороны, необходимо обеспечивать возможности работы в реальном масштабе времени [3].

Появление устройств виртуальной реальности внесло коррективы в требования к производительности. При комфортных на мониторе 60 fps (кадрах в секунду) пользователь виртуальной реальности

сталкивается с неприятными ощущениями, такими как головокружение, головные боли и тошнота [4]. Комфортными при этом являются значения 90 fps и выше.

Высокие требования подталкивают разработчиков к поиску новых методов и технологий для улучшения производительности их решений при рендеринге в реальном времени. Такой рендеринг можно представить как этапы графического конвейера, т.е. некоторой последовательности операций, которая, используя вершины, текстуры и дополнительные данные на входе, получают на выходе пиксели изображения. Чем меньше вершин объектов нам нужно пропустить через этот конвейер, тем быстрее он будет работать.

В этой области активно применяется концепция Level of Detail (LOD). Она позволяет управлять уровнем детализации в зависимости от различных факторов, например расстояние от камеры. Важно отметить, что снижение визуального качества модели при отдалении от камеры часто остается незамеченным, так как оно оказывает небольшое влияние на внешний вид объекта в дали.

Обычная методика включают в себя создание нескольких отдельных моделей для каждого уровня детализации. В современных 3D-редакторах существуют способы автоматической генерации таких моделей, что в значительной степени упрощает использование этой концепции.

С развитием аппаратного обеспечения и появлением Direct 11 (а позже и других графических API) появилась возможность использовать тесселяцию, т.е. динамический процесс добавления или вычитания деталей из трехмерной полигональной сетки и ее контуров силуэта на основе управляющих параметров. Как показало время, тесселяция не стала полной заменой созданию уровней детализации моделей. Причиной этому стал более низкий уровень общей производительности. При этом она часто используется в связке с обычными методами либо в тех местах, где использование LOD может привести к ошибкам в отображении, например при отрисовке больших объектов – ландшафта или поверхностей воды (см. рис. 1). Чтобы скрыть такие дефекты, зону первого уровня детализации приходится значительно увеличить, что приводит к соответствующему снижению производительности. Использование тесселяции для первого уровня позволит снизить нагрузку на графический конвейер.

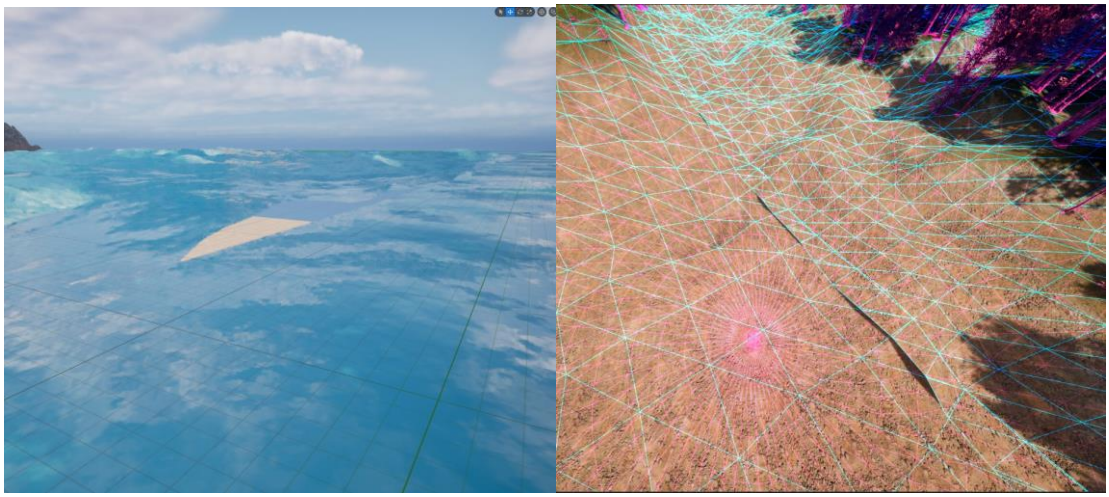

Рис. 1. Некорректное отображение стыка уровней детализации

Система тесселяции представляет собой триангуляцию набора точек, меняющихся в зависимости от расстояния до камеры. Такая триангуляция должна иметь определенные свойства, такие как неоднородность, т.е. необходимость избегать слишком «тонких» треугольников, что заметно улучшит визуальное качество сетки. К неоднородности также относится рендеринг крупных треугольников в разреженных областях и маленьких – в плотных.

Для создания сеток с указанными свойствами можно использовать деревья квадрантов. Кроме того, метод обладает приемлемыми показателями производительности [5, 6].

Рассмотрим лист дерева квадрантов и соответствующую ему ячейку. Ячейка считается сбалансированной для генерации сетки, если стороны ячейки пересекаются угловыми точками соседних ячеек не более одного раза с каждой стороны. Когда это условие выполняется для всех листьев дерева, мы можем говорить о сбалансированности всего дерева квадрантов. Графическое представление этого состояния показано на рис. 2.

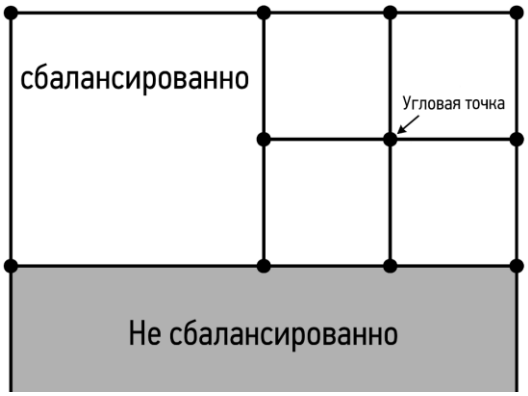

Рис. 2. Сбалансированный и несбалансированный прямоугольник

Рассмотрим ячейку *V* и группу одинакового размера ячеек 5 × 5 с центром в центре ячейки *V*. Назовем эту группу расширенным кластером *V*. Дерево квадрантов будет хорошо сбалансировано, если оно сбалансировано, и расширенный кластер каждого листа *U*, содержащего точку из исходного набора точек, также находится в дереве квадрантов, и при этом кластер листа *U* не содержит никакой другой точки из этого исходного набора.

Рассмотрим ячейку *V* и группу одинакового размера ячеек *n* × *n* с центром в середине ячейки *V*. Назовем эту группу расширенным кластером *V*. Введем новое понятие для дерева – хорошо сбалансировано. Оно применимо, если дерево квадрантов сбалансировано, и в каждом листе *U* содержится точка из исходного набора, то расширенный кластер этого листа также должен быть частью дерева квадрантов, при этом ни одна другая точка из исходного набора не должна находиться в кластере листа *U*.

Для каждого узла листа *U* проверяем, содержит ли расширенный кластер другую точку набора, если да, то дополнительно делим и балансируем дерево. Когда дерево будет хорошо сбалансировано, триангулируем его.

Для этого примем угловые точки ячеек как вершины для нашей триангуляции. В начале они будут представлять собой набор квадратов с точками и без. Далее для каждого квадрата с точкой мы искривляем ближайший угол ее ячейки, чтобы он соединился с другой вершиной, а затем триангулируем получившиеся точки.

Для каждого квадрата (без точек внутри и без угловых точек на его сторонах) добавим диагональ. Благодаря свойству хорошей балансировки, которым мы разделили точки, ни один квадрат с углом, пересекающим сторону, не будет искривленным.

При этом значение *n* будет определять степень тесселяции. Чем оно больше, тем более плотной будет сетка объекта.

Используя эти теоретические знания, на языке C++ был разработан алгоритм тесселяции поверхности, подходящий для визуализации поверхности воды. Результат его работы с различными настройками плотности, интерпретированными из значения *n*, показан на рис. 3.

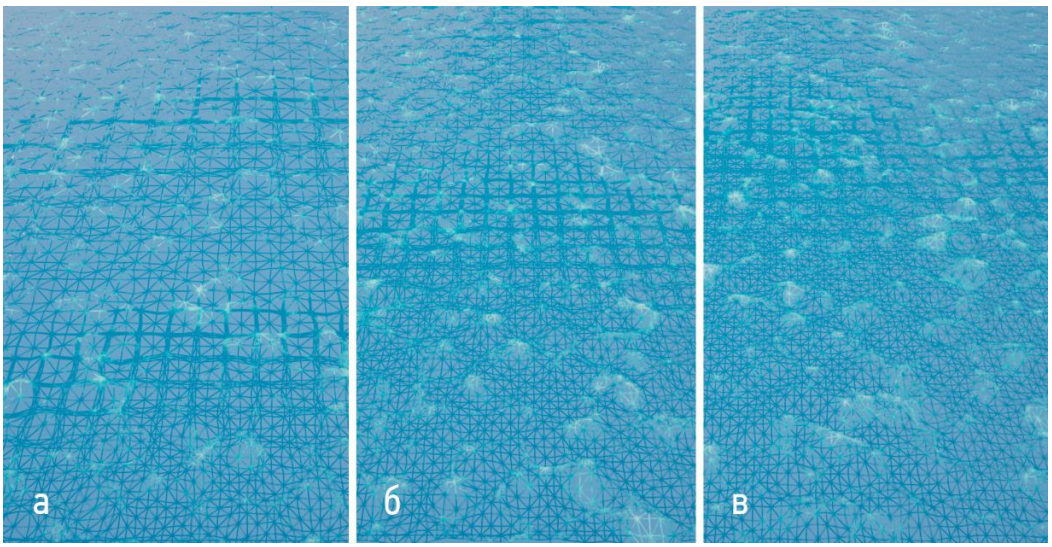

Рис. 3. Результат работы тесселяции при значениях: *а* – 1; *б* – 0,66; *в* – 0,33

Тестирование показало, что симуляция с использованием деревьев квадрантов снижает нагрузку на графический конвейер. Результаты тестирования (на аппаратной платформе – Intel i5 11400h, Rtx 3060 6 Гб, ОЗУ 16 Гб) при различных разрешениях приведены в табл.

Таблица

Сравнение производительности (в fps) при различных разрешениях и уровнях тесселяции

Степень тесселяции	1080p	1440p	2160p
0	83	45	15
0,33	93	59	20
0,66	97	61	22
1	106	67	44

Метод тесселяции с использованием деревьев квадрантов, используемый совместно со стандартными методами LOD, рассмотренный в данной статье, снижает нагрузку на графический конвейер, позволяя более оптимально использовать ресурсы аппаратной платформы. Для демонстрации метода разработанная система тесселяции применена для поверхности воды в среде Unreal Engine 5.

Список литературы

1. Кузьмин А. В. Трехмерное моделирование и визуализация в медицине // Вестник Пензенского государственного университета. 2015. № 4 (12). С. 122–127.
2. Бодин О. Н., Кузьмин А. В. Синтез реалистичной поверхности модели сердца // Медицинская техника. 2006. № 6. С. 15–18.
3. Патент № 2295772 C1 Российская Федерация, МПК G06T 11/60. Способ генерирования текстуры в реальном масштабе времени и устройство для его реализации / Бодин О. Н., Гайдуков С. А., Кузьмин А. В., Малышкин А. А. ; заявитель Пензенский государственный университет (ПГУ). № 2005129968/09 ; заявл. 26.09.2005 ; опубл. 20.03.2007.
4. Меньшикова Г. Я., Ковалёв А. И. Векция в виртуальных средах: психологические и психофизиологические механизмы формирования // Национальный психологический журнал. 2015. № 4 (20). URL: https://cyberleninka.ru/article/n/vektsiya-v-virtualnyh-sredah-psihologicheskie-i-psihofiziologicheskie-mehanizmy-formirovaniya (дата обращения: 14.03.2024).
5. Har-Peled S. Geometric approximation algorithms. Providence : American mathematical society, 2011. Vol. 173. 362 p.
6. de Berg M., Cheong O., van Kreveld M., Overmars M. H. (2008). Computational Geometry Algorithms and Applications. 3rd ed. Berlin ; Heidelberg : Springer-Verlag. 2008.

References

1. Kuz'min A.V. Three-dimensional modeling and visualization in medicine. *Vestnik Penzenskogo gosudarstvennogo universiteta = Bulletin of the Penza State University*. 2015;(4);122–127. (In Russ.)
2. Bodin O.N., Kuz'min A.V. Synthesis of a realistic surface of the heart model. *Medicinskaja tehnika = Medical equipment*. 2006;(6):15–18. (In Russ.)
3. Patent № 2295772 C1 Russian Federation, MPK G06T 11/60. A method for generating textures in real time and a device for its implementation. Bodin O.N., Gajdukov S.A., Kuz'min A.V., Malyshkin A.A.; the applicant is Penza State University (PSU). № 2005129968/09; appl. 26.09.2005; publ. 20.03.2007. (In Russ.)
4. Men'shikova G.Ja., Kovaljov A.I. Vector in virtual environments: psychological and psychophysiological mechanisms of formation. *Nacional'nyj psihologicheskij zhurnal = National Journal of Psychology*. 2015;(4). (In Russ.). Available at: https://cyberleninka.ru/article/n/vektsiya-v-virtualnyh-sredah-psihologicheskie-i-psihofiziologicheskie-mehanizmy-formirovaniya (accessed 14.03.2024).
5. Har-Peled S. *Geometric approximation algorithms*. Providence : American mathematical society, 2011;173:362.
6. de Berg M., Cheong O., van Kreveld M., Overmars M.H. (2008). *Computational Geometry Algorithms and Applications*. 3rd ed. Berlin; Heidelberg: Springer-Verlag. 2008.

Поступила в редакцию / Received 15.02.2024

Принята к публикации / Accepted 15.03.2024