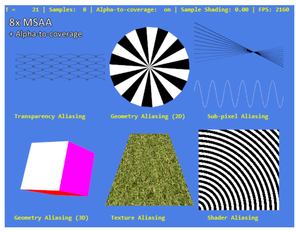
PatientZero 6 декабря в 10:10

Алгоритмы антиалиасинга в реальном времени

https://www.beyond3d.com/content/articles/122/

Разработка игр, Работа с 3D-графикой

Перевод



Алиасинг (aliasing) — это, возможно, наиболее фундаментальный и самый широко обсуждаемый артефакт 3D-рендеринга всех времён. Однако в игровом сообществе его часто недопонимают. В этой статье я подробно расскажу о теме сглаживания (антиалиасинга, anti-aliasing, AA) в реальном времени, особенно о том, что касается игр, и в то же время буду излагать всё достаточно простым языком.

Различные типы алиасинга и сглаживания, обсуждаемые в статье, будут в основном иллюстрироваться при помощи скриншотов из OpenGL-программы, предназначенной для демонстрации вариаций артефактов алиасинга.

Эту программу можно скачать здесь.

Прежде чем начать, позвольте мне сказать несколько слов о производительности: поскольку она является самым важным аспектом графики реального времени, мы в основном сосредоточимся на том, *почему* и *как* сегодня реализуется антиалиасинг. Я упомяну характеристики производительности, но строгая оценка всех представленных в этой статье способов антиалиасинга во разнообразных случаях реального использования будет слишком широкой темой для поста.

Природа алиасинга

> «Если ты знаешь себя и знаешь врага, то не подвергнешься опасности и в сотне битв»

Как учит нас Сунь Цзы, чтобы победить врага, нам нужно сначала понять его. Врагом — простите меня за излишнюю драматичность — методов сглаживания являются артефакты *алиасинга*. Поэтому нам первым делом нужно понять, как и откуда появляется *алиасинг*.

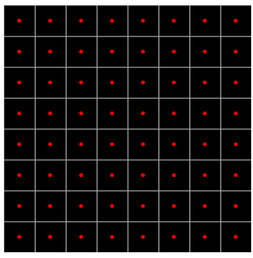
Термин *алиасинг* был впервые введён в области обработки сигналов, в которой он изначально описывал эффект, возникающий, когда разные непрерывные сигналы становятся неразличимыми (или начинают *искажать* друг друга) при дискретизации. В 3D-рендеринге этот термин обычно имеет более конкретное значение: он относится ко множеству нежелательных артефактов, которые могут возникать, когда 3D-сцена рендерится для отображения на экране, состоящем из фиксированной сетки пикселей.

В этом случае 3D-сцена является непрерывным сигналом, а процесс генерирования значений цветов для каждого пикселя *дискретизирует* этот сигнал для создания выходных данных рендеринга. Цель методов антиалиасинга заключается в том, чтобы выходные данные как можно точнее походили на сцену на заданной сетке пикселей, при этом минимизируя визуально искажающие артефакты.

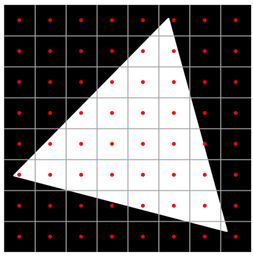
На Рисунке 1 показан алиасинг в простой сцене, состоящей из единственного белого треугольника на чёрном фоне. На этапе растеризации стандартного рендеринга сэмплируется *центральная* позиция каждого пикселя: если он находится в треугольнике, то пиксель будет *закрашен* белым, в противном случае он закрашивается чёрным. В результате получается хорошо заметный эффект *«лесенки»*, один из самых узнаваемых артефактов алиасинга.

При идеальном сглаживании для каждого пикселя определяется, какая часть его площади закрыта треугольником. Если пиксель закрыт на 50%, то он должен быть заполнен цветом на 50% между белым и чёрным (средним серым). Если он закрыт меньше, то должен быть пропорционально темнее, если больше — то пропорционально светлее. Полностью закрытый пиксель является белым, полностью незакрытый — чёрным. Результат этого процесса показан на четвёртом рисунке. Однако выполнение этого вычисления в реальном времени в общем случае является невыполнимой задачей.

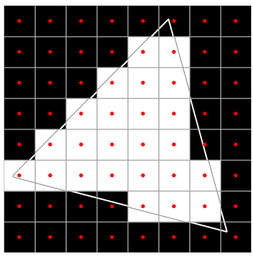
***Рисунок 1**. Простейший алиасинг.*



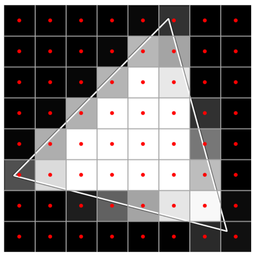
1-1. Сетка 8x8 с помеченными центрами



1-2. Сетка 8x8 с треугольником



1-3. Сетка 8x8 с растеризированным треугольником



1-4. Сетка 8x8 с идеально сглаженными выходными данными

▸ То же самое в виде GIF

Типы алиасинга

Хотя все артефакты алиасинга можно свести к проблеме дискретизации представления непрерывного сигнала на фиксированной сетке, состоящей из ограниченного количества пикселей, конкретные причины их возникновения очень важны для выбора устраняющего их эффективного способа сглаживания. Как будет видно в дальнейшем, некоторые методы антиалиасинга могут идеально справляться с простым геометрическим алиасингом, показанным на Рисунке 1, но терпеть неудачу при исправлении алиасинга, создаваемого другими процессами рендеринга.

Поэтому чтобы в полной мере обсудить относительные сильные и слабые стороны техник сглаживания, мы сгруппировали артефакты алиасинга, возникающие при 3D-рендеринге, в пять отдельных категорий. Это группирование зависит от точных условий генерирования артефактов. На Рисунке 2 показаны эти типы алиасинга на реальном примере, отрендеренном с помощью OpenGL.

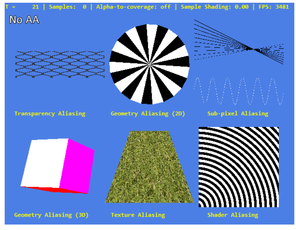


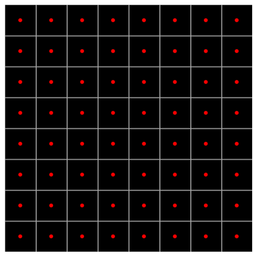
***Рисунок 2**: Различные типы алиасинга. Слева направо, сверху вниз:*
• Единственный выравненный относительно экрана прямоугольник с частично прозрачной текстурой.
• «Мельница», состоящая из выравненных относительно экрана переменных белых и чёрных треугольников.
• Несколько чёрных линий различной ширины, начиная с 1 пикселя сверху до 0,4 пикселя снизу, и белая линия толщиной 0,5, отображающая синусоиду.
• Куб, состоящий из шести плоских закрашенных прямоугольников
• Наклонная плоскость, текстурированная высокочастотной текстурой травы.
• Выравненный относительно экрана прямоугольник с пиксельным шейдером, определяющим цвет каждого пикселя на основе функции синуса.

Самым распространённым типом алиасинга, о котором мы уже говорили, является **геометрический алиасинг**. Этот артефакт возникает, когда какой-то примитив сцены (обычно треугольник) частично пересекается с пикселем, но это частичное перекрытие не учитывается в процессе рендеринга.

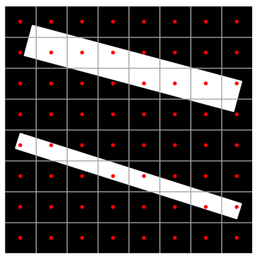
Алиасинг прозрачности возникает в текстурированных примитивах с частичной прозрачностью. Верхнее левое изображение на Рисунке 2 отрендерено с использованием одного прямоугольника, заполненного частично прозрачной текстурой сетчатого забора. Поскольку сама текстура — это просто фиксированная сетка пикселей, её нужно сэмплирования в точках, на которые накладывается каждый пиксель отрендеренного изображения, и для каждой такой точки должно приниматься решение, нужна ли в ней прозрачность. В результате возникает та же проблема сэмплирования, которую мы уже встречали на сплошной геометрии.

Несмотря на то, что фактически он является типом геометрического алиасинга, **подпиксельный алиасинг** требует особого рассмотрения, так как он ставит уникальные задачи для аналитических методов сглаживания, которые недавно получили большую популярность в рендеринге игр. Мы подробно рассмотрим их в статье. Подпиксельный алиасинг возникает тогда, когда растеризируемая структура накладывается менее чем на один пиксель в сетке буфера кадров. Такое чаще всего происходит в случае узких объектов — шпилей, телефонных или электрических линий, или даже мечей, когда они находятся достаточно далеко от камеры.

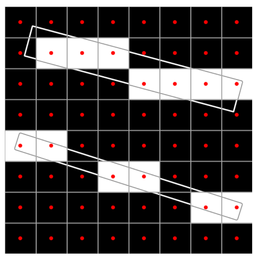
***Рисунок 3.** Иллюстрация подпиксельного алиасинга.*



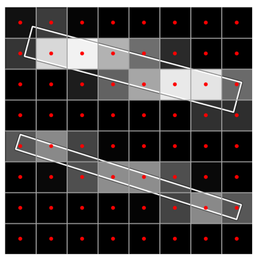
3-1. Сетка 8x8 с отмеченными центрами



3-2. Сетка 8x8 с двумя отрезками прямых



3-3. Сетка 8x8 с растеризированными отрезками, без AA



3-4. Сетка 8x8 с идеально сглаженным треугольником

▸ То же самое в виде GIF

На Рисунке 3 показан подпиксельный алиасинг в простой сцене, состоящей из двух отрезков прямых. Верхний имеет ширину в один пиксель, и хотя при растеризации он демонстрирует знакомый артефакт «лесенку» геометрического алиасинга, результат всё равно в целом соответствует по форме входным данным. Нижний отрезок имеет ширину полпикселя. При растеризации часть пересекаемых им столбцов пикселей не имеет одного центра пикселя в пределах отрезка. В результате он разделяется на несколько несвязанных фрагментов. То же самое можно заметить на прямых линиях и кривой синусоиды на Рисунке 2.

Текстурный алиасинг возникает при недостаточном сэмплировании текстуры, особенно в случаях анизотропного сэмплирования (это случаи, когда поверхность сильно наклонена относительно экрана). Обычно артефакты, создаваемые таким типом алиасинга, не очевидны на неподвижных скриншотах, но проявляются в движении как мерцание и неустойчивость пикселей. На Рисунке 4 это показано на нескольких кадрах программы-примера в режиме анимации.

▸ Рисунок 4: Анимированная высокочастотная текстура с артефактами мерцания

Текстурный алиасинг обычно можно предотвратить использованием mip-текстурирования и фильтрацией высококачественных текстур, но он всё равно иногда остаётся проблемой, особенно с некоторыми версиями драйверов популярных видеопроцессоров, субдискретизирующих высокоанизотропные текстуры. На него также влияют различные методы антиалиасинга, поэтому он тоже включён в демонстрационную программу.

И, наконец, **шейдерный алиасинг** возникает, когда программа пиксельного шейдера, выполняемая для каждого пикселя и определяющая его цвет, генерирует результат с алиасингом. Такое часто случается в играх с шейдерами, создающими контрастное освещение, например, зеркальные засветы на основании карты нормалей, или с техниками контрастного освещения типа cel shading или задней подсветкой. В демонстрационной программе это аппроксимируется одним шейдером, вычисляющим функцию синуса для координат текстуры и закрашивающего все отрицательные результаты чёрным, а все положительные — белым.

Техники сглаживания на основе сэмплирования

Вооружившись пониманием артефактов алиасинга и всех типов алиасинга, которые могут возникнуть при рендеринге 3D-сцены, мы можем начать *исследование техник антиалиасинга*. Эти техники можно разбить на две категории: техники, пытающиеся снизить алиасинг увеличением количества генерируемых рендерингом сэмплов и техники, пытающиеся смягчить артефакты алиасинга анализом и постобработкой сгенерированных изображений. Категория *техник сглаживания на основе сэмплирования* в более проста, поэтому стоит начать с неё.

Давайте снова рассмотрим наш первый пример с треугольником в сетке 8×8 пикселей. Проблема со стандартным рендерингом заключается в том, что мы сэмплируем только центр каждого пикселя, что приводит к генерации уродливой «лесенки» на рёбрах, которые не являются полностью горизонтальными или вертикальными. С другой стороны, вычисление покрытия каждого пикселя невозможно в реальном времени.

Интуитивным решением будет простое *увеличение количества сэмплов, взятых на пиксель*. Эта концепция показана на Рисунке 5.

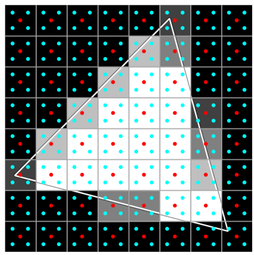


***Рисунок 5**: треугольник, растеризированный с четырьмя упорядоченными сэмплами на пиксель*

Центры пикселей снова помечены красными точками. Однако в каждом пикселе сэмплируются на самом деле четыре отдельных места (они помечены бирюзовыми точками). Если треугольник не закрывает ни один из этих сэмплов, то пиксель считается чёрным, а если закрывает их всех, то белым. Здесь интересна ситуация, когда закрыта только часть пикселей: если закрыт один из четырёх, то пиксель будет на 25% белым и на 75% чёрным. В случае двух из четырёх соотношение 50/50, а при трёх закрытых сэмплах результатом будет более светлый оттенок в 75% белого.

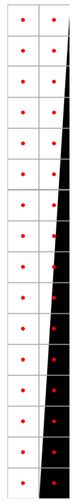
Эта простая идея является фундаментом всех методов антиалиасинга на основе сэмплирования. В этом контексте также стоит заметить, что когда количество сэмплов на пиксель стремится к бесконечности, то результат этого процесса будет стремиться к «идеальному» сглаженному примеру, показанному ранее. Очевидно, что качество результата сильно зависит от количества использованных сэмплов — но и производительность тоже. Обычно в играх используется 2 или 4 сэмплов на пиксель, а 8 и более обычно применяются только в мощных PC.

Существуют и другие важные параметры, изменение которых может влиять на качество получаемых результатов методов антиалиасинга на основе сэмплирования. В основном это *расположение сэмплов*, *тип сэмплов* и *группирование сэмплов*.

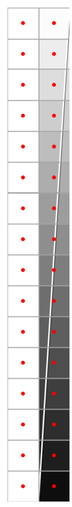
Расположение сэмплов

Расположение сэмплов внутри пикселя сильно влияет на конечный результат, особенно в случае небольшого количества сэмплов (2 или 4), которое чаще всего используется в графике реального времени. В предыдущем примере сэмплы располагаются так, как будто они являются центрами отрендеренного изображения в четыре раза больше исходного (16×16 пикселей). Это интуитивно понятно и легко достигается простым рендерингом изображения бОльших размеров. Этот метод известен как *антиалиасинг на упорядоченной сетке* (ordered grid anti-aliasing, OGAA), также его иногда называют *субдискретизацией* (downsampling). В частности, его реализуют *принудительным* увеличением разрешения рендеринга по сравнению с разрешением монитора.

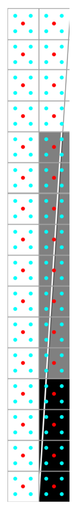
Однако упорядоченная сетка часто неоптимальна, особенно для почти вертикальных и почти горизонтальных линий, в которых как раз наиболее очевидны артефакты алиасинга. На Рисунке 6 показано, почему так *происходит*, и как *повёрнутая* или *разреженная* сетка сэмплирования обеспечивает гораздо *лучшие* результаты:



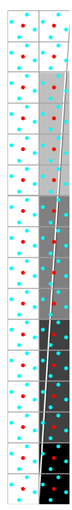
6-1. Сцена с почти вертикальной линией



6-2. Идеально сглаженная растеризация



6-3. Растеризация с четырьмя упорядоченными сэмплами



6-4. Сглаживание с четырьмя разреженными сэмплами

В этом почти вертикальном случае идеальный результат с четырьмя сэмплами должен иметь пять различных оттенков серого: черный при полностью незакрытых сэмплах, 25% белого при одном

закрытом сэмпле, 50% при двух и так далее. Однако растеризация с упорядоченной сеткой даёт нам всего три оттенка: чёрный, белый и 50/50. Так происходит, потому что упорядоченные сэмплы расположены в два столбца, а потому, когда один из них закрывается почти вертикальным примитивом, другой тоже скорее всего будет закрыт.

Как показано на изображении с разреженным сэмплингом, эту проблему можно обойти, изменив положение сэмплов внутри каждого пикселя. Идеальным расположением сэмплов для сглаживания является *разреженное*. Это означает, что при N сэмплов никакие два сэмпла не имеют одного общего столбца, строки или диагонали в сетке NxN. Такие паттерны соответствуют решениям задачи об N ферзях. Методы антиалиасинга, в которых используются такие сетки, называют выполняющими *антиалиасинг на разреженных сетках (sparse grid anti-aliasing, SGAA)*.

Типы сэмплов

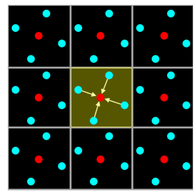
Самый простой подход к антиалиасингу изображения на основе сэмплирования заключается в том, что все вычисления выполняются для «реального» пикселя каждого сэмпла. Хотя этот подход высокоэффективен для удаления всех типов артефактов алиасинга, он также является очень вычислительно затратным, потому что при N сэмплах увеличивает в N раз затраты на затенение, растеризацию, занимаемую полосу пропускания и память. Техники, при которых все вычисления выполняются для каждого отдельного сэмпла, называются **сглаживанием суперсэмплингом (super-sampling anti-aliasing, SSAA)**.

Примерно в начале этого века в графическое оборудование была встроена поддержка **антиалиасинга мультисэмплингом (multi-sample anti-aliasing, MSAA)**, являющегося оптимизацией суперсэмплинга. В отличие от случая SSAA, в MSAA каждый пиксель затеняется только один раз. Однако для каждого сэмпла вычисляются значения глубины и стенсила, что обеспечивает то же качество сглаживания на рёбрах геометрии, что и в SSAA, при значительно меньшем снижении производительности. Кроме того, возможны дальнейшие улучшения производительности, особенно занятой полосы пропускания, если поддерживается сжатие Z-буфера и буфера цвета. Они поддерживаются во всех современных архитектурах видеопроцессоров. Из-за способа оптимизации сэмплирования MSAA, с алиасингом прозрачности, текстур и шейдеров таким образом напрямую справляться невозможно.

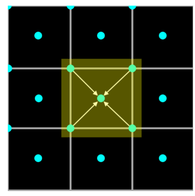
Третий тип сэмплирования был представлен компанией NVIDIA в 2006 году в технологии **антиалиасинг покрытия сэмплирования (coverage sampling anti-aliasing, CSAA)**. MSAA отделяет затенение от поликсельного вычисления глубины и стенсила, а CSAA добавляет сэмплы *покрытия*, которые не содержат значений цвета, глубины или стенсила — в них хранится только двоичное значение покрытия. Такие двоичные сэмплы используются для помощи в смешивании готовых сэмплов MSAA. То есть режимы CSAA добавляют сэмплы покрытия к режимам MSAA, но не имеет смысла выполнять сэмплирование покрытия без создания множества сэмплов MSAA. В современном оборудовании NVIDIA используется три режима CSAA: 8xCSAA (4xMSAA / 8 сэмплов покрытия), 16xCSAA (4x/16), 16xQCSAA (8x/16) и 32xCSAA (8x/32). У AMD есть похожая реализация с 4x EQAA (2x/4), 8xEQAA (4x/8) и 16xEQAA (8x/16). Дополнительные сэмплы покрытия обычно только незначительно влияют на производительность.

Группировка сэмплов

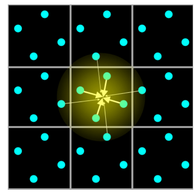
Последним ингредиентом методов AA на основе сэмплирования является способ группировки сэмплов, то есть то, как отдельные сэмплы, сгенерированные при рендеринге, собираются в конечный цвет каждого пикселя. Как показано на Рисунке 7, для этой цели используются различные фильтры группировки. На рисунке показаны пиксели 3×3 — бирюзовые точки обозначают позиции сэмплов, а жёлтый оттенок обозначает фильтр группировки сэмплов.



7-1. Фильтр Box



7-2. Фильтр Quincunx



7-3. Фильтр Tent

Очевидный и самый распространённый метод группировки просто накапливает каждый сэмпл в квадратной области, представляющей пиксель с равными весами. Это называется фильтром «box», и используется во всех обычных режимах MSAA.

Одним из первых подходов, пытавшихся улучшить эффект сглаживания с помощью малого количества сэмплов, является антиалиасинг «quincunx». В нём на пиксель вычисляется всего два сэмпла: один в центре, и один смещённый на поликсель вверх и влево. Однако вместо этих двух сэмплов накапливается пять сэмплов, составляющих паттерн, показанный на Рисунке 7. Это приводит к значительному снижению алиасинга, но в то же время размывает всё изображение, потому что значения цветов окружающих пикселей группируются в каждый пиксель.

Более гибкий подход был представлен в 2007 году компанией AMD в серии видеопроцессоров HD 2900. В них используется программируемая группировка сэмплов, что позволяет реализовать режимы группировки «патчи tent» и «wide tent». Как показано выше, каждый сэмпл не имеет одинаковый вес. Вместо этого используется функция взвешивания, зависящая от расстояния до центра пикселя. Узкий (narrow) и широкий (wide) варианты используют разный размер ядра фильтра. Эти способы группировки можно сочетать с различным количеством сэмплов, и некоторые из полученных результатов показаны на общем сравнении. Что касается quincunx AA, то эти методы представляют собой компромисс между резкостью изображения и снижением алиасинга.

Сравнение AA сэмплирования

На Рисунке 8 показано сравнение всех рассмотренных нами методов AA на основе сэмплирования с различным количеством сэмплов. На изображении «ground truth» показано ближайшее к «реальному», идеальное представление сцены. Оно создано сочетанием 8xSGSSAA и 4×4 OGSSAA.

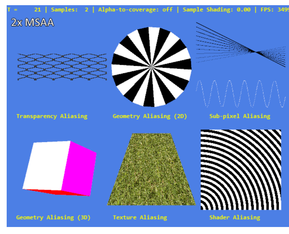
Стоит заметить аналогичное качество SGMSAA и SGSSAA с одинаковым количеством сэмплов при геометрическом алиасинге, и нехватку антиалиасинга прозрачности, текстур и шейдеров в случае MSAA. Недостатки упорядоченных паттернов сэмплинга, особенно для почти горизонтальных и почти вертикальных линий сразу заметны при сравнении 4x SGSSAA и 2×2 OGSSAA. При всего двух сэмплах на пиксель OGSSAA ограничен только горизонтальным (2×1) или только вертикальным (1×2) AA, а разреженный паттерн в какой-то мере может покрывать оба типа рёбер.

Методы AA с фильтрами группировки сэмплов, отличающиеся от обычного фильтра box, обычно обеспечивают более качественное снижение алиасинга на сэмпл, но страдают от эффекта размытия всего изображения.

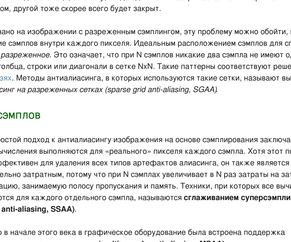
Нужно заметить ещё один важный пункт — особенно в свете последующего обсуждения аналитических методов AA – все эти методы на основе сэмплирования одинаково хорошо применяются и к подпиксельному алиасингу, и обычному геометрическому алиасингу.

***Рисунок 8**: Обработка различных типов алиасинга различными методами AA на основе сэмплирования.*

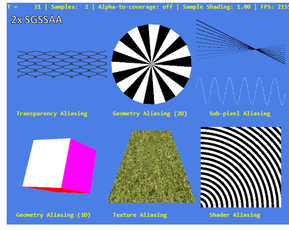
«Истинное» изображение



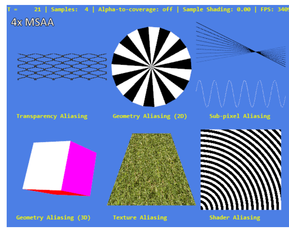
Без AA



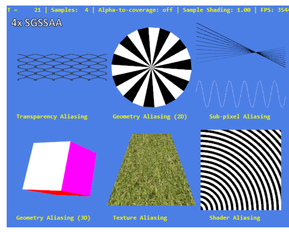
2x MSAA



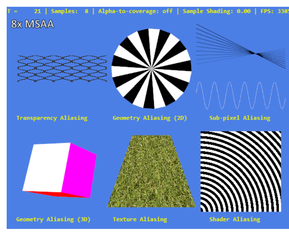
2x SGSSAA



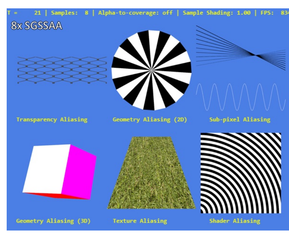
4x MSAA



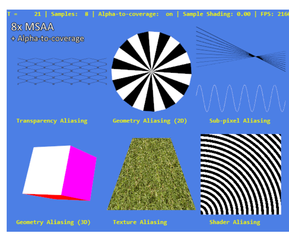
4x SGSSAA



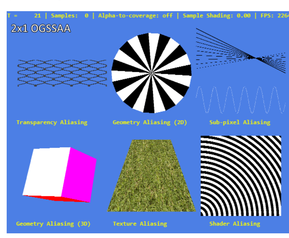
8x MSAA



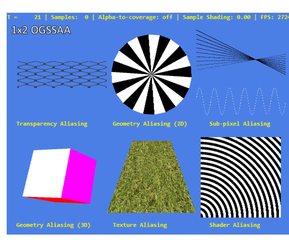
8x SGSSAA



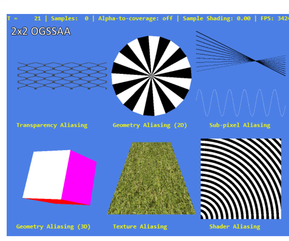
8x MSAA + alpha-to-coverage



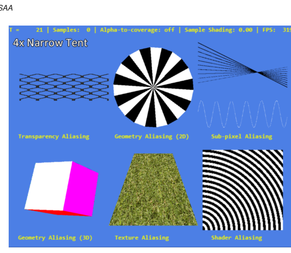
2x1 OGSSAA



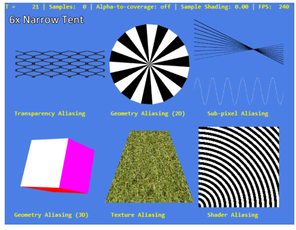
1x2 OGSSAA



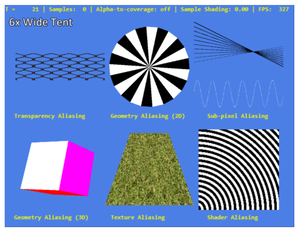
2x2 OGSSAA



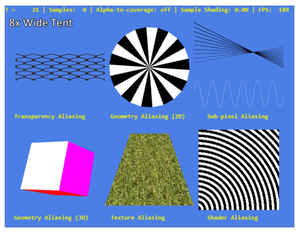
4x Narrow Tent



6x Narrow Tent



6x Wide Tent



8x Wide Tent

▸ То же самое в виде GIF

Аналитические техники антиалиасинга

Техники на основе сэмплинга интуитивно понятны и довольно хорошо работают с достаточно большим количеством сэмплов, но в то же время они затратны с точки зрения вычислений. Эта проблема усугубляется, когда используются способы рендеринга (например, отложенное затенение), которые могут усложнить применение эффективных типов сэмплирования с аппаратным ускорением. Поэтому исследуются другие способы уменьшения визуальных артефактов, создаваемых алиасингом при 3D-рендеринге. Такие методы рендерят обычное изображение с одним сэмплом на пиксель, а затем пытаются идентифицировать и устранить алиасинг, анализируя *изображение.*

Краткое введение и знакомство

Хотя идея сглаживания сгенерированных компьютером изображений популяризирована благодаря статье Решетова 2009 года о **морфологическом антиалиасинге** (который часто называют **MLAA**) [1], она ни в коем случае не является новой. Жуль Блументаль привёл сжатое описание этой техники в своей статье 1983 года для SIGGRAPH «Edge Inference with Applications to Antialiasing», которая активно применяется в современных методах [2]:

> «Ребро, сэмплируемое по точкам для отображения на растровом устройстве, и непараллельное оси дисплея, выглядит как лесенка. Этот артефакт алиасинга часто возникает в компьютерных изображениях, сгенерированных двухмерными и трёхмерными алгоритмами. Точная информация о ребре часто больше недоступна, но из множества вертикальных и горизонтальных сегментов, формирующих эту лесенку, можно извлечь аппроксимацию исходного ребра с точностью, превосходящей точность растра. Таким образом можно обеспечить сглаживание ребра лесенки.
>
> Такие извлечённые рёбра можно использовать для повторного затенения пикселей, которые они пересекают, таким образом сглаживая извлечённые рёбра. Сглаженные извлечённые рёбра оказываются более привлекательной аппроксимацией реальных рёбер, чем из аналоги с алиасингом».

В 1999 году Ишики и Куниэда представили первый вариант этой техники, предназначенной для использования в реальном времени, которая выполнялась сканированием пар строк и столбцов изображения, и могла быть реализована аппаратно [3].

В общем случае, все чисто аналитические методы антиалиасинга выполняются в три этапа:

1. Распознавание разрывов в изображении.
2. Воссоздание геометрических рёбер из паттерна разрывов.
3. Сглаживание пикселей, пересекающих эти рёбра, смешиванием цветов каждой их стороны.

Отдельные реализации аналитического антиалиасинга различаются тем, как реализованы эти шаги.

Распознавание разрывов

Простейший и самый распространённый вариант распознавания разрывов — это простое изучение конечного отрендеренного цветового буфера. Если разность цветов двух соседних пикселей (их *расстояние*) больше какого-то порогового значения, то присутствует разрыв, в противном случае его нет. Эти показатели расстояний часто вычисляются в цветовом пространстве, которое лучше моделирует человеческое зрение, чем RGB, например, в HSL.

На Рисунке 9 показан пример отрендеренного изображения, а также вычисленных из них горизонтальных и вертикальных разрывов.

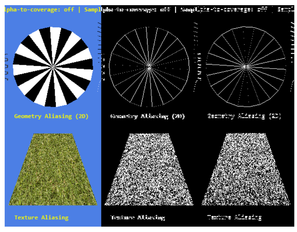


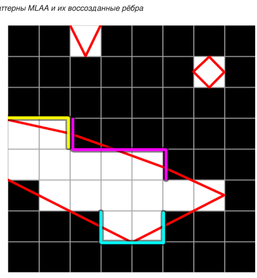
Рисунок 9: *Распознавание разрывов в цветовом буфере. Слева: изображение без AA. В центре: горизонтальные разрывы. Справа: вертикальные разрывы.*

Чтобы ускорить процесс распознавания разрывов или снизить количество ложноположительных распознаваний (например, в текстуре, или вокруг текста на Рисунке 9), можно использовать другие буферы, генерируемые в процессе рендеринга. Обычно для прямого и обратного рендерера доступен Z-буфер (буфер глубин). В нём хранится значение глубины для каждого пикселя, и его можно использовать для распознавания рёбер. Однако это только работает для распознавания *рёбер-силуэтов*, то есть внешних рёбер 3D-объекта. Чтобы рассматривать и рёбра внутри объекта, то нужно использовать ещё один буфер вместо или в дополнение к Z-буферу. Для отложенных рендереров часто генерируется буфер, хранящий направление поверхностей нормалей каждого пикселя. В таком случае для распознавания рёбер подходящей метрикой будет угол между соседними нормалями.

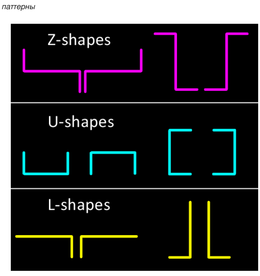
Воссоздание рёбер и смешивание

Способ воссоздания геометрических рёбер из разрывов немного различается в разных методах аналитического AA, но все они выполняют похожие действия по сопоставлению паттернов на горизонтальных и вертикальных разрывах для распознавания типичного паттерна «лесенки» артефактов алиасинга. На Рисунке 10 показаны паттерны, используемые по описанию Решетова в MLAA и способ воссоздания из них рёбер.

Рисунок 10: *Паттерны MLAA и их воссозданные рёбра*



Распознанные паттерны



Паттерны разрывов, использованные в MLAA

После воссоздания геометрических рёбер достаточно просто вычислить, насколько верхний/нижний

или правый/левый пиксель вдоль ребра должен вносить вклад в смешанный цвет пикселя, чтобы генерировать сглаженный внешний вид.

Достоинства и недостатки аналитического сглаживания

По сравнению с методами антиалиасинга на основе сэмплирования, аналитические решения имеют несколько важных преимуществ. При правильной работе (на правильно распознанных геометрических рёбрах) они могут обеспечить качество, равное качеству методов на основе сэмплирования с очень высоким количеством сэмплов, при этом тратя меньше вычислительных ресурсов. Более того, они легко применимы во множестве случаев, в которых АА на основе сэмплирования реализуется сложнее, например, в случае отложенного затенения.

Однако аналитический АА — это не панацея. Неотъемлемой проблемой чисто аналитических методов на основании единственного сэмпла является то, что они не могут справляться с **подпиксельным алиасингом**.

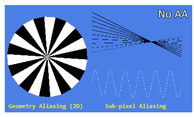
Если передать аналитическому алгоритму сглаживания структуру пикселей, показанную в правом верхнем углу Рисунка 2, то он никак не сможет понять, что разделённые группы пикселей на самом деле составляют линию. На этом этапе есть два равно неприятных способа решения этой проблемы: или размытие пикселей, снижающее видимый алиасинг, но и уничтожающее детали, или консервативная обработка только чётко определённых артефактов–«лесенок», которые совершенно точно вызваны алиасингом; при этом подпиксельный алиасинг сохранится и будет искажён.

Ещё одна проблема аналитических методов заключается в **ложных срабатываниях**. Когда часть изображения распознаётся как ребро с алиасингом, но на самом деле им не является, оно будет искажено смешиванием. Это особенно очевидно на тексте, и тоже требует идти на компромиссы: более консервативное распознавание рёбер приведёт к меньшему количеству ложных срабатываний, но при нём также будут упущены некоторые рёбра, которые на самом деле имеют алиасинг. С другой стороны, при расширении порога распознавания рёбер эти рёбра тоже будут включены, но это приведёт к ложноположительным срабатываниям. Поскольку анатилический антиалиасинг в своей основе пытается экстраполировать больше информации из растеризированного изображения, чем на самом деле в нём есть, от этих проблем невозможно избавиться полностью.

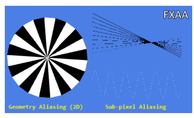
Наконец, интерпретация рёбер этими методами может сильно меняться в зависимости от разницы в единственном пикселе. Поэтому при использовании однопиксельных аналитических методов антиалиасинга может увеличиться или даже добавиться **мерцание и временная нестабильность изображения**: единственный изменившийся пиксель в исходном изображении может превратиться в сглаженных выходных данных в целую мерцающую линию.

На Рисунке 11 показаны некоторые из успешных и неудачных результатов использования аналитических методов АА на примере стандартных алгоритмов FXAA и SMAA1x. Последний обычно считается лучшим чисто аналитическим однопиксельным алгоритмом, который можно использовать в реальном времени.

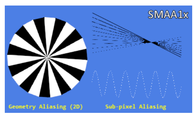
Рисунок 11: *Аналитические методы АА*



Без АА



FXAA

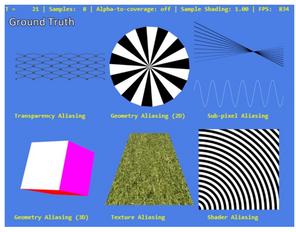


1x SMAA

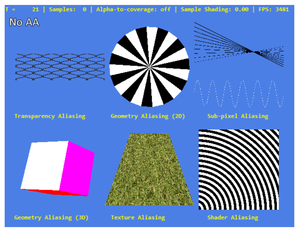
Сравнение аналитических методов антиалиасинга

На Рисунке 12 показано сравнение между результатами работы FXAA, SMAA1x и «идеальным» изображением, и изображениями без АА, с 4xMSAA и 4xSGSAAA из предыдущего сравнения.

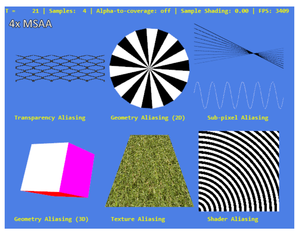
Рисунок 12: *Обработка различных видов алиасинга разными аналитическими и сэмплирующими методами АА*



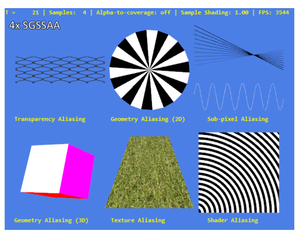
«Идеальное» изображение



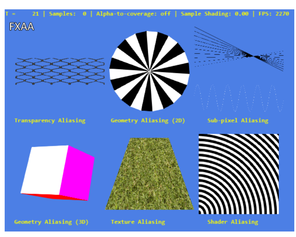
Без АА



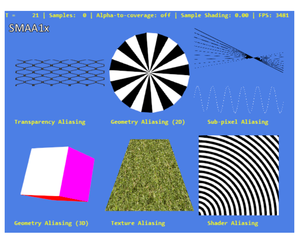
4x MSAA



4x SGSSAA



FXAA



SMAA

▸ То же самое в виде GIF

Заметьте, что в отличие от MSAA, эти аналитические методы не интересует, были ли причинами артефактов алиасинга геометрия, прозрачность или даже вычисление шейдеров. Все рёбра обрабатываются одинаково. К сожалению, то же самое относится и к рёбрам экранного текста, хотя искажение при SMAA1x и меньше, чем при FXAA.

Оба метода не справляются с антиалиасингом в случае подписельного алиасинга, однако они обрабатывают эту неудачу по-разному: SMAA1x просто решает вообще не влиять на отдельные белые пиксели синусоиды, а FXAA смешивает их с их окружением. Более желательная обработка зависит от контекста и личных предпочтений.

Более объективно можно сказать, что SMAA1x обрабатывает некоторые углы линий в тесте 2D-геометрии и кривые в примере с шейдерным алиасингом точно лучше, чем FXAA, обеспечивая более гладкий результат, который приятнее эстетически и ближе к «идеальному» изображению. Так получилось благодаря более сложному этапу воссоздания рёбер и смешивания, подробности которого объяснены в статье об SMAA 2012 года Хименеса et alia [4].

Будущее антиалиасинга

Мы получили хорошее понимание множества методов антиалиасинга (аналитических и на основе сэмплинга), которые активно используются сейчас в играх. Настало время немного порассуждать. Какие техники антиалиасинга можно будет увидеть на новом поколении консолей, поднимающих планку технологий? Как можно будет смягчить недостатки имеющихся методов, и как новое оборудование позволить использовать новые алгоритмы?

Ну, будущее уже частично здесь: **комбинированные методы на основе сэмплирования и аналитического антиалиасинга**. Эти алгоритмы создают несколько сэмплов сцены — посредством традиционного мульти- или суперсэмплирования или с помощью временного накопления между кадрами — и сочетают их с анализом, генерируя конечное изображение с антиалиасингом. Это позволяет им снизить проблемы подписксельного аляисинга и временной нестабильности односэмпловых чисто аналитических методов, однако всё равно даёт гораздо лучшие результаты на геометрических рёбрах, чем чистые методы сэмплирования со схожими характеристиками производительности. Очень простую комбинацию дополнительного сэмплирования и аналитического АА можно получить сочетанием односэмпловой аналитической техники наподобие FXAA с субдискретизацией из буфера с более высоким разрешением. Более сложными примерами таких методов являются SMAA T2x, SMAA S2x и SMAA 4x, а также TXAA. Методы SMAA подробно объясняются и сравниваются в этой статье, в то время как NVIDIA реализовала собственный подход к TXAA здесь. Высока вероятность, что такие комбинированные методы будет более широко использоваться в будущем.

Ещё одним вариантом, пока не получившим широкого распространения, но имеющим большой потенциал на будущее, является кодирование в процессе рендеринга дополнительной геометрической информации, которая позже будет использоваться на этапе аналитического антиалиасинга. Примеры такого подхода — это антиалиасинг с геометрической постобработкой (Geometric Post-process Anti-Aliasing, GPAA) и антиалиасинг с использованием буфера геометрии (Geometry Buffer Anti-Aliasing, GBAA), демо которых выложены здесь.

Наконец, общий пул памяти ЦП и видеопроцессора новых консольных платформ и будущих архитектур PC могут позволить использовать техники, предназначенные для эксплуатации таких общих ресурсов. В недавней статье «Asynchronous Adaptive Anti-aliasing Using Shared Memory» Барринджер и Мёллер описывают технику, выполняющую традиционный односэмплвый рендеринг, в то же время распознавая *важные* пиксели (например, находящиеся на ребре) и растеризируя для них в ЦП дополнительные разреженные сэмплы [5]. Хотя это требует серьёзной реструктуризации процесса выполнения рендеринга, результаты выглядят многообещающе.

Справочные материалы

[1] A. Reshetov, «Morphological antialiasing», в *Proceedings of the Conference on High Performance Graphics 2009 HPG '09*, New York, NY, USA, 2009, pp. 109–116.

[2] J. Bloomenthal, 'Edge Inference with Applications to Antialiasing', ACM SIGGRAPH Comput. Graph., vol. 17, no. 3, pp. 157–162, Jul. 1983.

[3] T. Isshiki and H. Kunieda, 'Efficient anti-aliasing algorithm for computer generated images', в *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems ISCAS '99*, Orlando, FL, 1999, vol. 4, pp. 532–535.

[4] J. Jimenez, J. I. Echevarria, T. Sousa, and D. Gutierrez, 'SMAA: Enhanced Subpixel Morphological Antialiasing', *Comput. Graph. Forum*, vol. 31, no. 2pt1, pp. 355–364, May 2012.

[5] R. Barringer and T. Akenine-Möller, 'A ^{4}: asynchronous adaptive anti-aliasing using shared memory',*ACM Trans. Graph.*, vol. 32, no. 4, pp. 100:1–100:10, Jul. 2013.

Метки: антиалиасинг, аляисинг, сглаживание графики

+75 | 172 | 12k | 11

Подписчики
475,5 Карма | 582,5 Рейтинг | 628

@PatientZero
Переводчик-фрилансер

Комментарии 11

PapaBubaDiop 06.12.17 в 11:53 +6

Хороший обзор. Одно замечание — существуют не только ортогональные 2Д решетки, на которые дискретизируются линии и полигоны. Есть устройства с вокселями и, например, гексогональными пикселями.

Кстати, автор первой ссылки — Александр Решетов (кличка Решето) — выпускник МФТИ, мы с ним в Интел в 1996-2000 году 3Д библиотеку лабали.

lleff 06.12.17 в 12:43 0

Спасибо за перевод. А нету ли в современном железе аналога CSAA, только чтобы вместо кучи сэмплов рассчитывало одно единственное число перекрытости пикселя треугольником? Для векторной графики это было бы идеально, глубина тут не нужна.

T-362 06.12.17 в 16:53 0

Даже кроме теоретической части как какой тип работает — получился отличный справочник как какой тип ВЫГЛЯДИТ.

Jamdaze 06.12.17 в 17:00 0

Насчёт будущего. Что вы скажите про необходимость АА в разрешении 4к+?

WST 06.12.17 в 17:22 0

4К 4К рознь, на достаточно большом экране и при 4К будут прекрасно видны все артефакты, тут от плотности пикселей исключительно зависит. Но чаще всего алиасинг прекрасно заметен даже на современных мобильных устройствах, у которых плотность пикселей просто огромная, так что тема антиалиасинга себя не изжила.

creker 06.12.17 в 18:14 0

На современных устройствах в играх с хорошей картинкой нередкость использовать не нативное разрешение. Естественно алиазинг в этом случае будет многократно заметнее.

creker 06.12.17 в 17:21 +1

Собственно, будущее уже наступило. В этом поколении во главе комбинированные техники. В основном временная реконструкция с каким-то аналитическим. Дает удивительные результаты.

stychos 06.12.17 в 18:08 0

Хороший стиль изложения.

Sinatr 07.12.17 в 11:51 +1

Спасибо, интересно. Пара «претензий».

В оригинале картинки находятся в одном месте и меняются кликом на ссылках ниже, очень удобно сравнивать. Вывалянные друг за другом картинки здесь не очень удобно сравнивать, гифка — так себе заменитель, на хабре чтоли технически нельзя добиться того же, что в и оригинале? Возможно имело смысл вообще не постить картинки, только гифки, и подставить ссылки на оригинал? Или это заодно и печатная версия? У оригинала кстате дебильная навигация выпадающим списком, в этом смысле на хабре текст читается приятнее.

> … Алиасинг прозрачности возникает… типом геометрического аляисинга, подписельный аляисинг…

Возможно это только у меня глаза спотыкаются, при чтении «русифицированных» таким образом терминов в разных падежах? Я бы оставил aliasing, anti-aliasing в тексте, просто при первом упоминание термина добавил бы в скобочках перевод, сравните

> (оригинал) **Transparency aliasing**, on the other hand, happens within textured primitives that are partially transparent.
> (перевод) **Алиасинг прозрачности** возникает в текстурированных примитивах с частичной прозрачностью.
> (вариант) **Transparency aliasing** (ступенчатость при прозрачности), с другой стороны, проявляется при использовании текстур с прозрачностью.

Мне лично такой вариант кажется более естественным, но возможно я слишком долго живу за рубежом? Покритикуйте.

kosmos89 08.12.17 в 22:37 0

Мда, термин алиасинг стали некорректно использовать...
Ни Transparency Aliasing, ни Geometry Aliasing (2D) (за исключением окрестности центра круга), ни Geometry Aliasing (3D), ни Shader Aliasing, на самом деле алиасингом не являются. Точнее могут являться, если отдалить изображения подальше.
«Лесенка» не является алиасингом и это совершенно нормальное явление в теории обработки сигналов. Чтобы появился алиасинг, надо, чтобы исходном сигнале появились частоты выше половины частоты семплирования. В нашем случае это означает наличие фич, размером меньше, чем два пикселя.

AlexTOPMAN 12.12.17 в 11:06 0

По поводу невозможности получения рисунка 1-4 из рисунка 1 осталось непонятным «почему?». % наложения треугольника на пиксел понятен и легко вычисляем (ровно также как и для случая 50/50 красить или не красить). Далее, для ускорения создаём из этих данных массив сразу по всем точкам изображения, требующим подобной edge-AA обработки и применяем сразу ко всему массиву следующую единственную формулу: % наложения треугольника на пиксел делим на 100% и умножаем на 256 (это число градаций серого от полного черного к полному белому по схеме 32 бит на цвет для нашего чёрно-белого рисунка) и получаем номер (дробную часть отбрасываем) градации серого для каждого конкретного пиксела, в который его нужно покрасить. Всё. Или я что-то упустил?

ЧТО ОБСУЖДАЮТ

Сейчас | Вчера | Неделя

Parcel — очень быстрый бандлер, не требующий настройки
9k | 29

Ускорение сборки C и C++ проектов
5,9k | 22

Rust vs. C++ на алгоритмических задачах
17,3k | 201

Чем страдает бизнес?
950 | 1

Конструирование сайта, защищенного от блокировок
7k | 28

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

ИНТЕРЕСНЫЕ ПУБЛИКАЦИИ

Apple специально замедляет старые iPhone [GT]

+7 1.7k 9

Двоичный поиск в графах

+6 414 12 0

Дмитрий Муромцев (ИТМО) — об онтологическом моделировании и формировании разговорного интеллекта [GT]

+9 267 0

Заметки к самостоятельному изучению французского языка [GT]

+5 345 9 2

Уязвимость из 1998 года снова в строю – встречайте ROBOT

+5 1.3k 4 0