Современные технологии улучшения качества 3D-изображений

Автор: Алексей Игнатенко (ignatenko@graphics.cs.msu.su)

В последнее время 3D игры стали выглядеть значительно лучше - более качественные изображения, более гладкие объекты, более красивые текстуры. И достигается это не в последнюю очередь за счет улучшения качества изображений, генерируемых играми, а не только за счет увеличения числа треугольников или повышения детализации текстур. В лексикон разработчиков игр уже твердо вошли американизмы типа "антиалиасинг", "трилинейная фильтрация" и "бамп-мэппинг". Все это - аппаратные технологии, позволяющие улучшать качество игрового видеоряда без существенных усилий со стороны сценаристов и дизайнеров. Данная статья посвящена наиболее популярным алгоритмам из этой области.

Содержание

Устранение ступенчатости

Одна из наиболее впечатляющих аппаратных технологий, все чаще находящая применение в игровых приложениях, - это устранение ступенчатости, или антиалиасинг (antialiasing).

Эта технология предназначена для устранения одной из ключевых проблем качества синтезированных изображений - лестничного эффекта (также часто называют алиасингом, aliasing). Часто этот эффект можно заметить на границах объектов или на линиях, близких к вертикальным или горизонтальным, но не строго вертикальных или, соответственно, горизонтальных. Несомненно, читатель много раз встречался с подобными эффектами, например, в трехмерных играх.

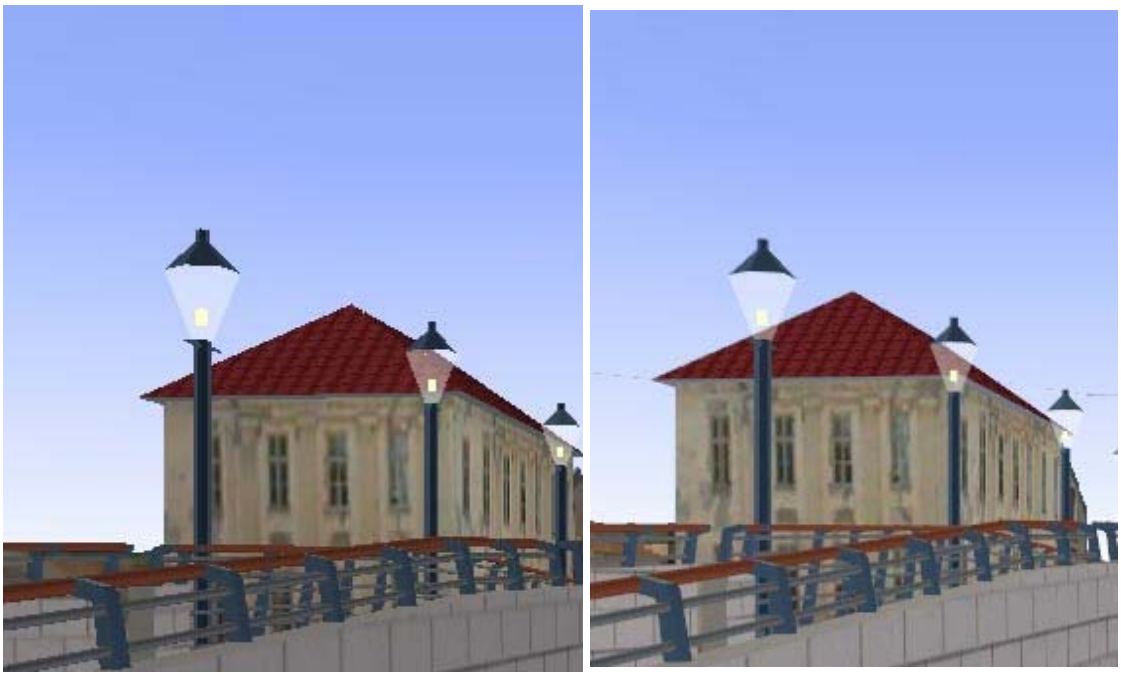


Рисунок 1. Левое изображение получено в стандартном режиме, правое - со включенным устранением ступенчатости. Обратите внимание на передачу мелких деталей и границы объектов (перила на мосту, фонари и т.п.)

Что является причиной лестничного эффекта? Опуская подробности, можно сказать, что главная причина - "сетка" пикселей на компьютерном мониторе. Эта сетка имеет конечное и весьма небольшое разрешение, а пиксели располагаются в строго фиксированных местах.

Ступенчатость возникает, когда точки на линиях пересекают строки или столбцы пикселей под небольшим углом. Часть линии шириной в один пиксель может попасть на один пиксель экрана, а часть - на другой. Таким образом, получается неопределенность: можно рисовать эту часть как один пиксель на одном ряду, один пиксель на другом ряду или закрашивать оба пикселя. К сожалению, все три способа вносят хорошо заметные дефекты в изображение. Если закрашивать только один пиксель, линия может получиться тоньше чем нужно, и будет хорошо заметен разрыв в том месте, где линия переходит с одного ряда пикселей на другой. Если закрашивать оба - ширина линии в этом месте будет два пикселя. Аналогичные артефакты возникают не только при рисовании линий, а практически на всех границах, встречающихся в изображении.

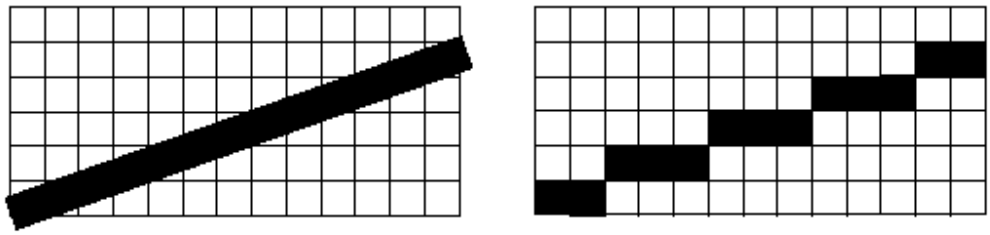


Рисунок 2.

Как же бороться с этим неприятным эффектом? Можно заметить, что размер одной "ступеньки" никогда не бывает больше, чем один пиксель. Поэтому наиболее естественный и простой путь - это увеличение экранного разрешения. Т.е. увеличение разрешения уменьшает размер пикселя и, следовательно, заметность артефактов. Однако увеличить разрешение можно не всегда. Например, оно может быть ограничено максимальным разрешением монитора или его может ограничивать приложение.

Другой путь - увеличение так называемого эффективного разрешения, т.е. использование специального алгоритма для получения цвета пикселя таким образом, чтобы эмулировать несколько пикселей. Такие методы и называются методами устранения ступенчатости.

Эти методы вычисляют значение цвета как бы внутри экранного пикселя в нескольких точках. Эти точки называются сэмплами (sample). Сэмплы представляют собой те самые дополнительные пиксели изображения, которые увеличивают эффективное разрешение. Значение сэмплов используются для вычисления конечного цвета пикселя.

Суперсэмплинг

Суперсэмплинг - это технология устранения ступенчатости, которая используется практически во всех современных аппаратных ускорителях. Графический процессор, который использует суперсэмплинг, визуализирует экранное изображение с разрешением, значительно большим чем текущее разрешение дисплея. Существует достаточно много методов выполнения этой операции, при этом их всех можно охарактеризуются числом используемых дополнительных пикселей. После рисования изображения с высоким разрешением, процессор уменьшает размер картинки до разрешения дисплея, причем эта операция производится с соответствующей фильтрацией.

Степень изменения размера изображения определяется отношением числа пикселей в исходном изображении (высокого разрешения) к числу пикселей в выходном изображении. Например, 2x суперсэмплинг пишет в буфер кадра в два раза больше пикселей, 4x - в четыре и т.д. Чем больше это число, тем более качественное устранение ступенчатости можно получить.

Как нетрудно догадаться, использование суперсэмплинга вызывает значительное ухудшение скорости визуализации. Если графический процессор рисует в четыре раза больше пикселей, скорость будет в четыре раза меньше по сравнение со стандартным режимом. И даже хуже, потому что необходимо дополнительно фильтровать изображение высокого разрешения. Вот почему после включения устранения ступенчатости в драйверах видеокарт уровня Nvidia GeForce 2, скорость рисования сильно падает.

Мультисэмплинг

Современные графические процессоры используют более сложную схему для устранения эффекта ступенчатости, так называемый мультисэмплинг. Хотя схема и более сложная, работает она значительно быстрее.

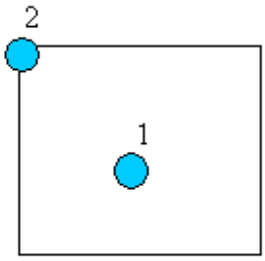
Суперсэмплинг работает таким образом, что на этапе создания изображения графический процессор не делает различий между стандартным режимом и режим устранения ступенчатости. Как говорилось выше, это работает медленно.

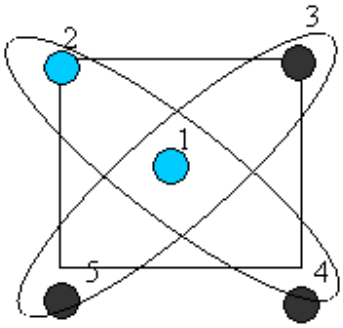
Основная идея мультисэмплинга - сделать генерацию и обработку дополнительных сэмплов интеллектуальной, и встроить ее в основной графический конвейер. Это, конечно, усложняет сам графический процессор, но в то же время позволяет значительно ускорить процесс устранения ступенчатости. Например, значение текстуры ищется только один раз для "главного" пикселя, делается ряд других оптимизаций.

Другими словами, вместо генерации изображения высокого разрешения, графический процессор для каждого пикселя создает несколько сэмплов, лежащих внутри него. Конечный цвет этого пикселя определяется из цветов сэмплов, обычно некоторым вариантом усреднения.

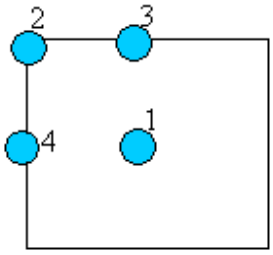
Алгоритмы мультисэмплинга различаются количеством сэмплов и их расположением (или, как говорят, шаблоном сэмплирования, sampling pattern), а также некоторыми приемами оптимизации.

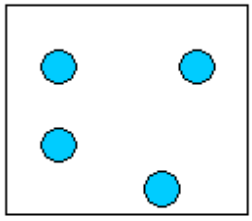
Различные производители в своих аппаратных реализациях используют различные шаблоны сэмплирования. Примеры даны на рисунке:





NVidia 2x NVidia Quincunx (tm)





NVidia 4x ATI Smoothvision (tm)

NVidia 2x

В этом режиме изображение последовательно рисуется сразу в два буфера. Между двумя проходами происходит сдвиг изображения на половину пикселя. Затем два изображения усредняются. Таким образом, это чистая 2x схема мультисэмплинга, которая работает быстро, но качество получается не очень хорошее.

NVidia 4x

Этот режим работает аналогично 2x, за исключением того, что используются соответственно 4 буфера.

NVidia Quincunx (tm)

В режиме QuincunxT (читается - "квинканкс") каждый сэмпл из каждого прохода дополнительно фильтруется с четырьмя соседними из второго изображения. NVidia заявляет, что такая схема дает качество, практически равное 4x. Однако отметим, что улучшение происходит не столько за счет увеличения количества сэмплов (используется два сэмпла, аналогично 2x-шаблону), а размывания изображения за счет смешивания с соседними пикселями. Тем не менее, на практике это дает хороший результат. Существует также NVidia AccuviewT - модифицированная технология, в которой положения сэмплов слегка сдвигаются внутрь основного пикселя для улучшения качества.

ATI Smoothvision (tm)

Эта технология реализована в графических процессорах ATI. В ней используются группы из 16 сэмплов, которые распределяются между различным количеством точек, в зависимости от желаемого качества устранения ступенчатости. В 2x-режиме сэмплы покрывают 8 точек, в 4x режиме - 4 точки и так далее. Каждый пиксель имеет 8 фиксированных псевдослучайных положений, в которые могут попасть сэмплы. Известно, что человеческий глаз хорошо замечает регулярные структуры, что в данном случае является

нежелательным. Использование нерегулярного шаблона позволяет создавать более приятные глазу картинки.

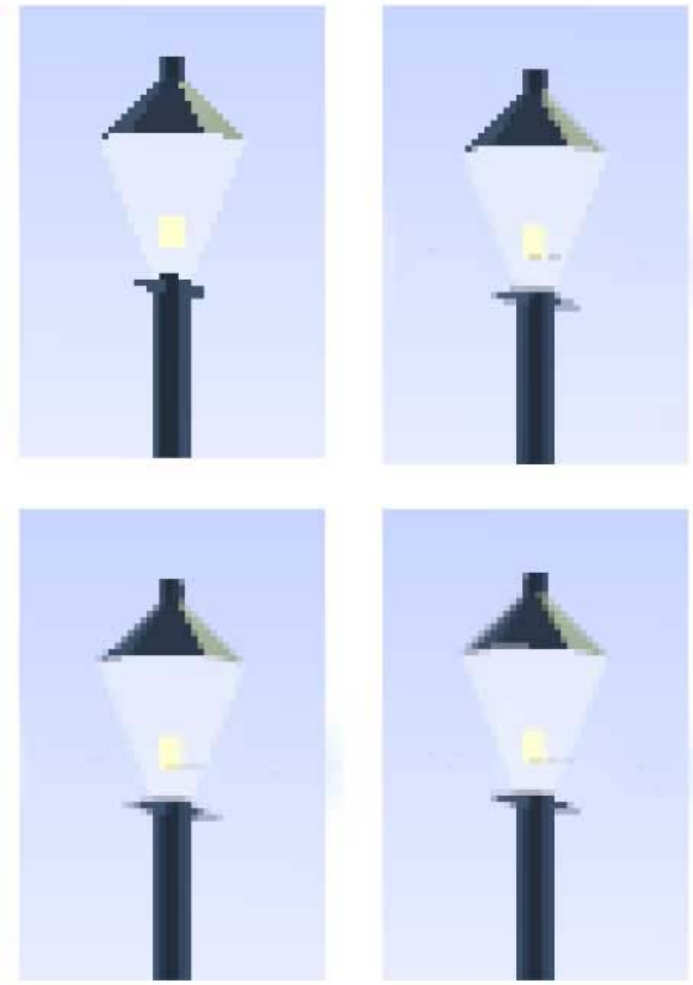


Рисунок 4. Результат применения мультисэмплинга для устранения ступенчатости. а) без мультисэмплинга б) 2x в) 4x г) 6x

Вместо итога заметим, что использование режимов устранения ступенчатости возможно даже без ведома приложения, обычно его можно включить через драйвера видеокарты.

Эффект размывания при движении

Размывание при движении (motion blur) - еще одна из множества технологий, которые используются для того, чтобы сделать картинки в 3D играх более реалистичными. Если камерой снимается реальный мир, можно заметить, что быстро движущиеся объекты как бы расплываются, размазываются на изображении. Этот эффект и называется размыванием при движении (иногда мы будем называть его размыванием движения). Его использование может заставить любую игру выглядеть "как в реальном мире".

На самом деле, размывание при движении пришло к нам не из реального мира (у людей объекты в принципе не должны размываться перед глазами), а из кинематографа. Этот эффект есть практически в любом фильме, но вы скорее всего его не замечаете. Как и многие другие артефакты фотографии, он воспринимается как дополнительный эффект, придающий ощущения реализма. В то время как фотографы стараюсь избавиться от артефактов, программисты игр старательно воссоздают их в своих

играх. Мы все так привыкли видеть размывание при движении в телевизионных программах и фильмах, что анимация без него выглядит нереалистично.

Во время работы камеры затвор открывает чувствительная пленку на короткое время, в течение которого на нее падает свет. Это вызывает химические реакции, пленка темнеет и в конце концов на ней образуется изображение сцены. Этот процесс известен как экспозиция. Если во время экспозиции сцена меняется, на пленке образуется размытое изображение.

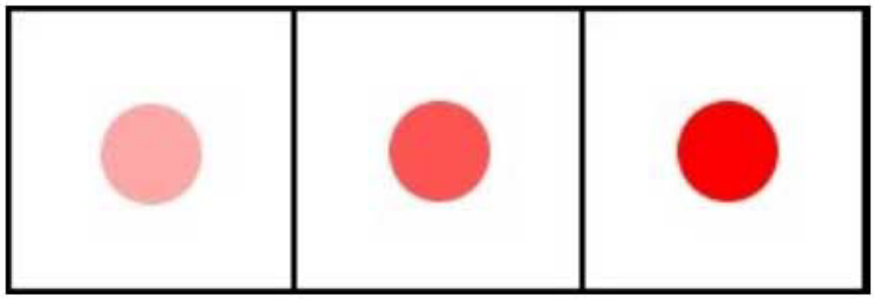


Рисунок 5. Чем дольше длится экспозиция, тем темнее объект.

Природа компьютерной анимации такова, что объекты не движутся плавно, глаз может замечать скачки при движении объекта. Предположим, что игра работает со скорость 60 FPS, т.е. каждый новый кадр рисуется через 1/60 секунды. Однако время непрерывно и не обновляется только 60 раз в секунду. Также и монитор, и человеческий глаз не могут "обрабатывать" изображения непрерывно, им нужен некоторый интервал времени для обновления изображения. Эффект анимации "скачками", который неизбежен даже при самой быстрой аппаратуре, называется временной ступенчатостью (по аналогии с эффектом ступенчатости изображения, который мы уже рассматривали).

Основная идея технология устранения этого эффекта подобна той, которая использовалась для устранения ступенчатости изображения - суперсэмплинг. Только в данном случае это так называемый временной суперсэмплинг. Для того, чтобы представить непрерывное время в одном кадре, необходимо дополнительно нарисовать несколько кадров-сэмплов, затем усреднить их и таким образом получить нужный эффект. Это усреднение, кстати, и вызовет эффект размывания, причем чем быстрее движется объект, тем сильнее он сдвинется на кадрах-сэмплах и поэтому его размывание будет сильнее.

Например, для размывания движения с использованием 4-х сэмплов необходимо рисовать со скоростью 240 FPS, чтобы получить вывод со скоростью 60 FPS. Чем больше сэмплов используется, тем более качественное изображение получится.

Однако такой метод работает медленно. Для современной игры даже на самой лучшей аппаратуре 60 FPS - предел мечтаний, а про 240 можно и не говорить. Таким образом необходимы быть может менее качественные, но более быстрые решения.

В некоторых компьютерных играх для эмуляции размывания при движении используются "следы" от объектов. Изображение текущего кадра смешивается с изображением предыдущего, т.е. получается что несколько последних кадров все время видны на экране. Это вызывает эффект "расплывания" объектов по экрану. Это, конечно, не настоящие устранение временной ступенчатости, но все же оно дает некоторый эффект.

Появление пиксельных шейдеров в современной аппаратуре позволило разработать новые алгоритмы для размывания движущихся объектов. Эти алгоритмы основываются на вычислении "скорости" движения каждого пикселя из кадра анимации и пост-обработки кадра таким образом, чтобы эмулировать эффект усреднения промежуточных кадров. На первом этапе сцена просто рисуется в буфер кадра и сохраняется в текстуре. Затем для каждой вершины находится разница между ее текущим положением и положением на предыдущем кадре - это и будет вектор скорости движения данной точки. Чем он длиннее, тем быстрее движется точка. Далее для каждого пикселя сцены выбираются несколько соседних точек текстуры (в которой, напомним, хранится изображение текущего кадра) и их значения

усредняются и записываются как цвет текущей точки. Соседние точки выбираются по информации о скорости основного пикселя.

Такие технологии работают достаточно быстро и качественно, хотя и требуют самую современную аппаратуру.

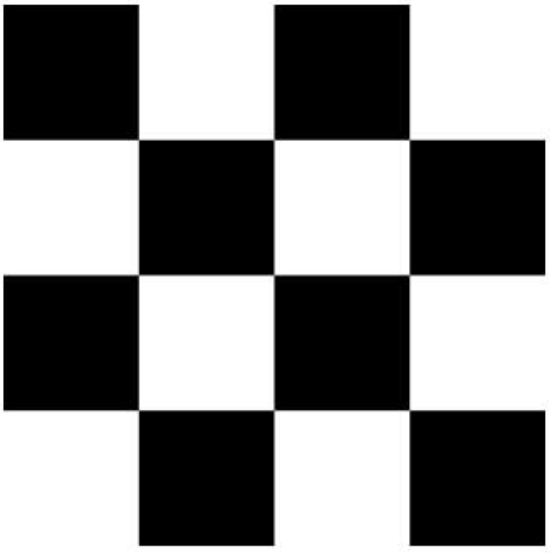
Фильтрация текстур

Текстурирование является важнейшим элементом сегодняшних 3D приложений, без него многие трехмерные модели теряют значительную часть своей визуальной привлекательности. Однако процесс нанесения текстур на поверхности не обходится без артефактов и соответствующих методов их подавления. В мире трехмерных игр то и дело встречаются специализированные термины типа "мип-мэппинг", "трилинейная фильтрация" и т.п., которые как раз и относятся к этим методам.

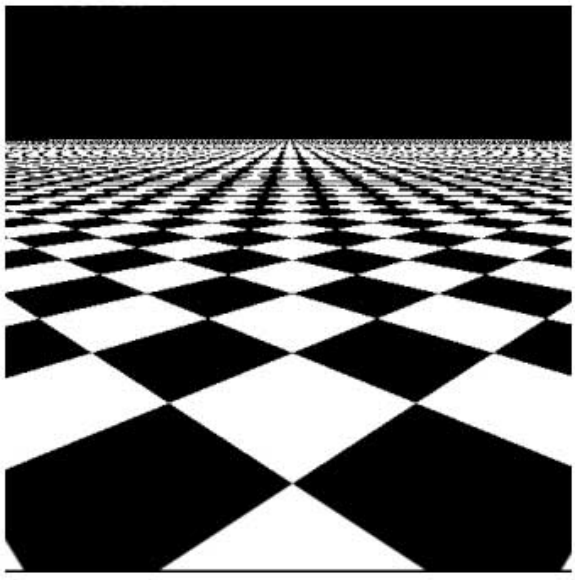
Частным случаем эффекта ступенчатости, рассмотренным ранее, является эффект ступенчатости текстурированных поверхностей, который, к сожалению, нельзя убрать методами мульти- или суперсэмплинга, описанными выше.

Представьте себе черно-белую шахматную доску большого, практически бесконечного размера. Допустим, мы рисуем эту доску на экране и смотрим на нее под небольшим углом. Для достаточно удаленных участков доски размеры клеток неизбежно начнут уменьшаться до размера одного пикселя и меньше. Это так называемое оптическое уменьшение текстуры (minification). Между пикселями текстуры начнется "борьба" за обладание пикселями экрана, что приведет к неприятному мельтешению, что является одной из разновидностей эффекта ступенчатости. Увеличение экранного разрешения (реального или эффективного) помогает только немного, потому что для достаточно удаленных объектов детали текстур все равно становятся меньше пикселей.

С другой стороны, наиболее ближние к нам части доски занимают большую экранную площадь, и можно наблюдать огромные пиксели текстуры. Это называется оптическим увеличением текстуры (magnification). Хотя эта проблема стоит не так остро, для уменьшения негативного эффекта с ней тоже необходимо бороться.



Это текстура шахматной доски



Плоскость с наложенной текстурой. Обратите внимание на искажения, возникающие при уменьшении клеток

Рисунок 6

Для решения проблем текстурирования применяется так называемая фильтрация текстур. Если разобраться в процессе рисования трехмерного объекта с наложенной текстурой, можно увидеть, что вычисление цвета пикселя идет как бы "наоборот", - сначала находится пиксель экрана, куда будет спроецирована некоторая точка объекта, а затем для этой точки находятся все пиксели текстуры, попадающие в нее. Выбор пикселей текстуры и их комбинация (усреднение) для получения финального цвета пикселя экрана и называется фильтрацией текстуры.

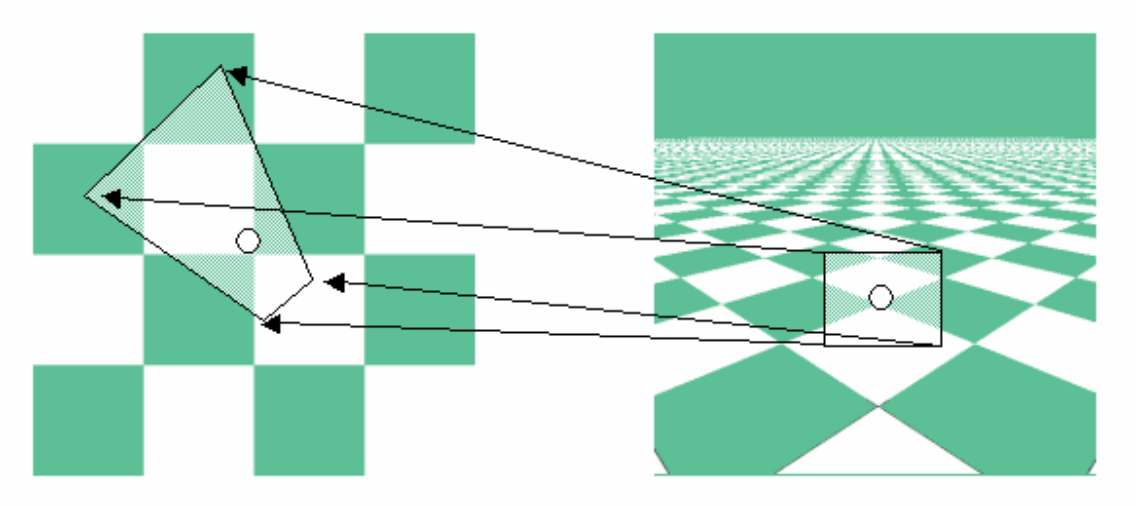


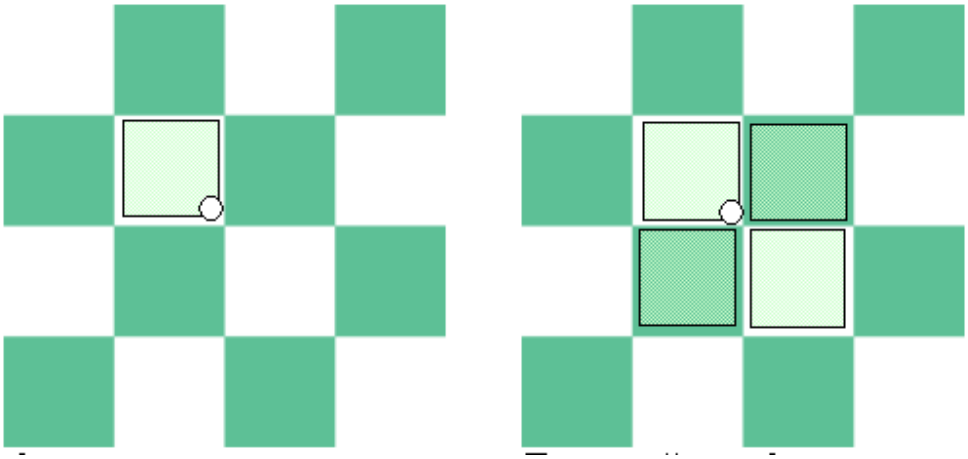
Рисунок 7. Область экрана и ее образ в текстуре

В процессе текстурирования каждому пикселю экрана ставится в соответствие координата внутри текстуры, причем эта координата не обязательно целочисленная. Более того, пикселю соответствует некоторая область в изображении текстуры, в которую могут попадать несколько пикселей из текстуры. Будем называть эту область образом пикселя в текстуре. Для ближних частей нашей доски пиксель экрана становится значительно меньше пикселя текстуры и как бы находится внутри него (образ содержится внутри пикселя текстуры). Для удаленных, наоборот, в каждый пиксель попадает большое количество точек текстуры (образ содержит в себе несколько точек текстуры). Образ пикселя может иметь различную форму и в общем случае представляет собой произвольный четырехугольник.

Рассмотрим различные методы фильтрации текстур и их вариации

Ближайший сосед (nearest neighbor)

В этом, наиболее простом, методе в качестве цвета пикселя просто выбирается цвет ближайшего соответствующего пикселя текстуры. Этот метод самый быстрый, но и наименее качественный. По сути, это даже не специальный метод фильтрации, а просто способ выбрать хоть какой-то пиксель текстуры, соответствующий экранному пикселю. Он широко применялся до появления аппаратных ускорителей, вместе с широким распространением которых появилась возможность использовать более качественные методы.



Фильтрация методом ближайшего соседа Билинейная фильтрация

Рисунок 8

Билинейная фильтрация (bilinear)

Билинейная фильтрация находит четыре пикселя текстуры, ближайшие к текущей точке экрана и результирующий цвет определяется как результат смешения цветов этих пикселей в некоторой пропорции.

Фильтрация методом ближайшего соседа и билинейная фильтрация работают достаточно хорошо когда, во-первых, степень уменьшения текстуры невелика, а во-вторых, когда мы видим текстуру под прямым углом, т.е. фронтально. С чем это связано?

Если рассмотреть, как описывалось выше, "образ" пикселя экрана в текстуре, то для случая сильного уменьшения он будет включать в себя очень много пикселей текстуры (вплоть до всех пикселей!). Кроме того, если мы смотрим на текстуру под углом, этот образ будет сильно вытянут. В обоих случаях описанные методы будут работать плохо, поскольку фильтр не будет "захватывать" соответствующие пиксели текстуры.

Для решения этих проблем применяют так называемый мип-мэппинг и анизотропную фильтрацию.

Мип-мэппинг

При значительном оптическом уменьшении точке экрана может соответствовать достаточно много пикселей текстуры. Это значит, что реализация даже самого хорошего фильтра будет требовать достаточно много времени для усреднения всех точек. Однако проблему можно решить, если создавать и хранить версии текстуры, в которых значения будут усреднены заранее. А на этапе визуализации для пикселя искать нужную версию исходной текстуры и брать значение из нее.

Термин mipmap произошел от латинского multum in parvo - многое в малом. При использовании этой технологии в памяти графического ускорителя в дополнение к изображению текстуры хранится набор ее уменьшенных копий, причем каждая новая ровно в два раза меньше предыдущей. Т.е. для текстуры размером 256x256 дополнительно хранятся изображения 128x128, 64x64 и т.д, вплоть до 1x1.

Далее для каждого пикселя выбирается подходящий уровень мипмапа (чем больше размер "образа" пикселя в текстуре, тем меньший мипмап берется). Далее значения в мипмапе могут усредняться билинейно или методом ближайшего соседа (как описано выше) и дополнительно происходит фильтрация между соседними уровнями мипмапа. Такая фильтрация называется трилинейной. Она дает весьма качественные результаты и широко используется на практике.

Рисунок 9. Уровни мипмапа

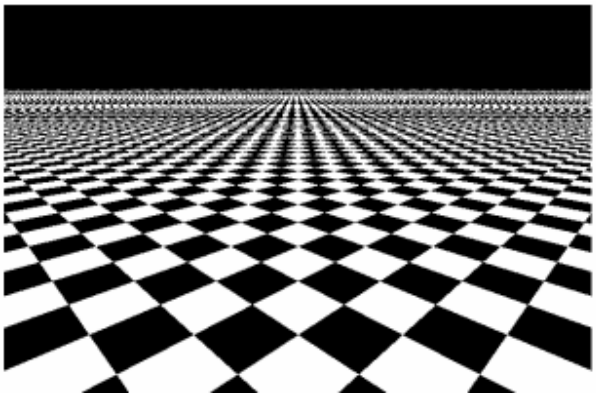
Однако проблема с "вытянутым" образом пикселя в текстуре остается. Как раз по этой причине наша доска на большом расстоянии выглядит очень нечеткой.

Анизотропная фильтрация

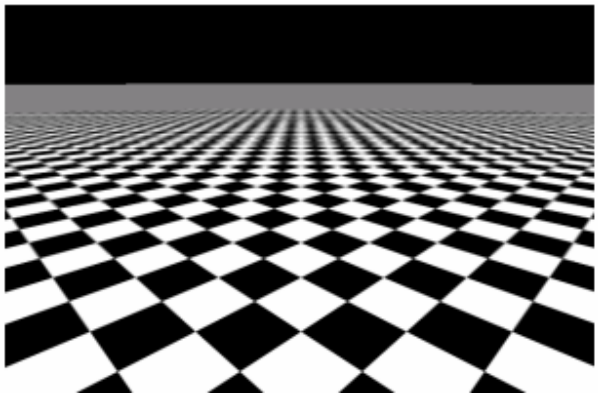
Анизотропная фильтрация - это процесс фильтрации текстуры, специально учитывающий случай вытянутого образа пикселя в текстуре. Фактически, вместо квадратного фильтра (как в билинейной фильтрации), используется вытянутый, что позволяет более качественно выбрать нужный цвет для экранного пикселя. Такая фильтрация используется вместе с мипмэппингом и дает весьма качественные результаты. Однако, существуют и недостатки: реализация анизотропной фильтрации достаточно сложна и при ее включении скорость рисования значительно падает. Анизотропная фильтрация поддерживается последними поколениями графических процессоров NVidia и ATI. Причем с различным уровнем анизотропии - чем больше этот уровень, чем более "вытянутые" образы пикселей можно корректно обрабатывать и тем лучше качество.

Сравнение фильтраций

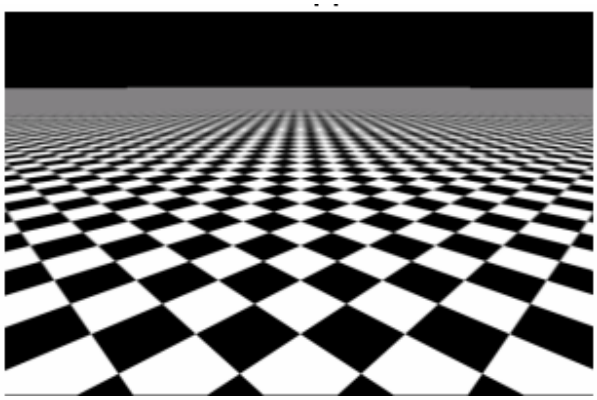
Итог следующий: для подавления артефактов алиасинга текстур аппаратно поддерживаются несколько методов фильтрации, различающиеся по своему качеству и скорости работы. Наиболее простой метод фильтрации - метод ближайшего соседа (который фактически не борется с артефактами, а просто заполняет пиксели). Сейчас чаще всего используется билинейная фильтрация вместе с мип-мэппингом или трилинейная фильтрация. В последнее время графические процессоры начали поддерживать наиболее качественный режим фильтрации - анизотропную фильтрацию.

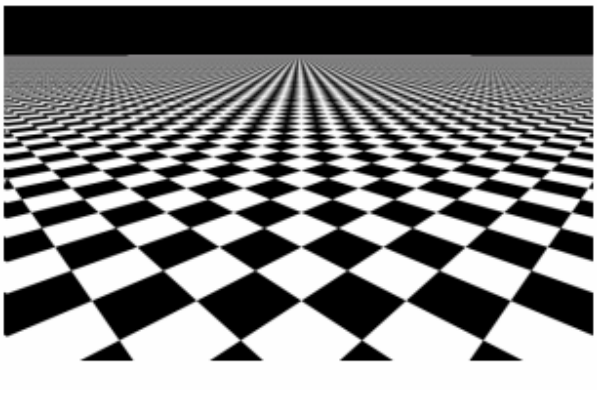


Ближайший сосед



Билинейная





Трилинейная Анизотропная

Рисунок 10.

Бамп-мэппинг

Бамп-мэппинг (bump mapping) - это тип графических спецэффектов, который призван создавать впечатление "шершавых" или бугристых поверхностей. В последнее время использование бамп-мэппинга стало чуть ли не стандартом игровых приложений.

Основная идея бамп-мэппинга - использование текстур для управления взаимодействием света с поверхностью объекта. Это позволяет добавлять мелкие детали без увеличения количества треугольников. В природе мы различаем мелкие неровности поверхностей по теням: любой бугорок будет с одной стороны светлым, а с другой - темным. Фактически, глаз может и не различать изменения в форме поверхности. Этот эффект и используется в технологии бамп-мэппинга. Одна или несколько дополнительных текстур накладываются на поверхность объекта и используются для вычисления освещенности точек объекта. Т.е. поверхность объекта не меняется вовсе, только создается иллюзия неровностей.

Существует несколько методов бамп-мэппинга, но прежде чем мы перейдем к их рассмотрению, необходимо выяснить, собственно как задать неровности на поверхности. Как уже говорилось выше, для этого используются дополнительные текстуры, причем они могут быть разных видов:

Карта нормалей. В этом случае каждый пиксель дополнительной текстуры хранит вектор, перпендикулярный поверхности (нормаль), закодированный в виде цвета. Нормали используются для вычисления освещенности.

Карта смещений. Карта смещений представляет собой текстуру в градациях серого, в каждом пикселе которой хранится смещение от оригинальной поверхности.

Эти текстуры готовятся дизайнерами трехмерных моделей вместе с геометрией и основными текстурами. Существуют и программы, позволяющие получать карты нормалей или смещений автоматически

Препроцессированный бамп-мэппинг (Pre-calculated bump mapping)

Текстуры, которые будут хранить информацию о поверхности объекта, создаются заранее, до этапа визуализации, путем затемнения некоторых точек текстуры (и, следовательно, самой поверхности) объекта и высветления других. Далее во время рисования используется обычная текстура.

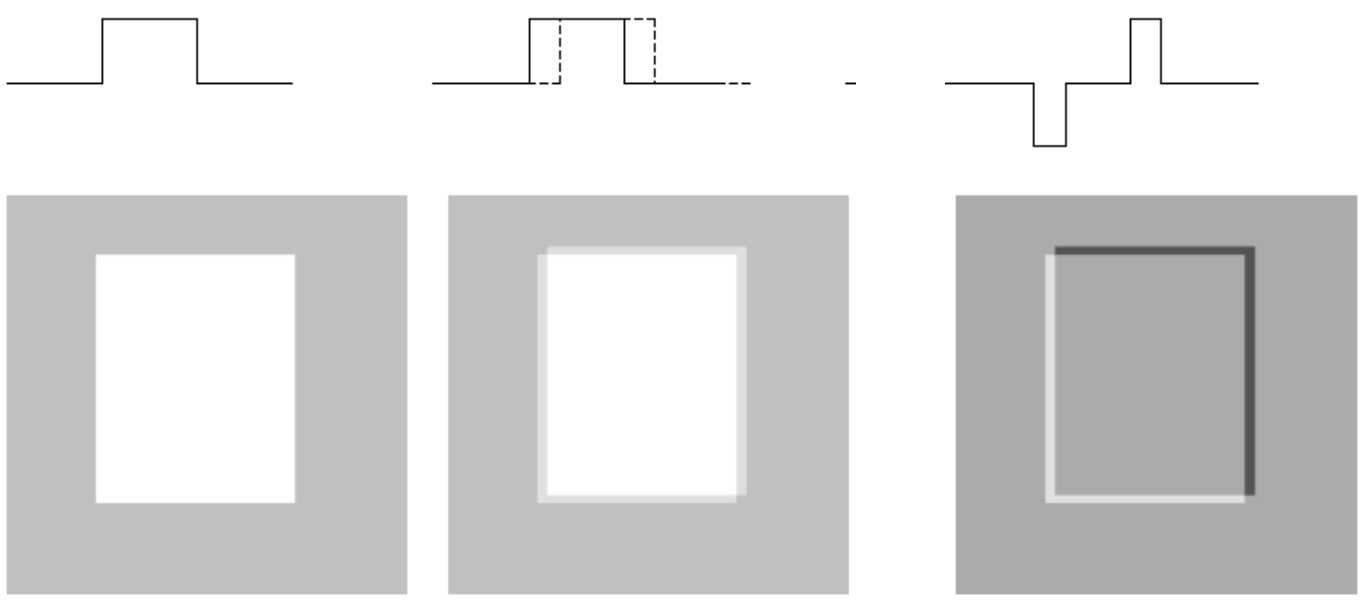
Этот метод не требует никаких алгоритмических ухищрений во время рисования, но, к сожалению, изменений в освещении поверхностей при изменении положений источников света или движения объекта не происходит. А без этого действительно успешной симуляции неровной поверхности не

создать. Подобные методы используются для статических частей сцены, часто для архитектуры уровней и т.п

Бамп-мэппинг с помощью тиснения (Emboss bump mapping)

Эта технология применялась на первых графических процессорах (NVidia TNT, TNT2, GeForce). Для объекта создается карта смещений. Рисование происходит в два этапа. На первом этапе карта смещений попиксельно складывается сама с собой. При этом вторая копия сдвигается на небольшое расстояние в направлении источника света. При этом получается следующий эффект: положительные значения разницы определяют освещенные пиксели, отрицательные - пиксели в тени. Эта информация используется для соответствующего изменения цвета пикселей основной текстуры.

Бамп-мэппинг с помощью тиснения не требует аппаратуры, поддерживающей пиксельные шейдеры, однако он плохо работает для относительно крупных неровностей поверхности. Также объекты не всегда выглядят убедительно, это сильно зависит от того, под каким углом смотреть на поверхность.



Карта смещений

Карта смещений складывается сама с собой, но сдвинутой на небольшое расстояние

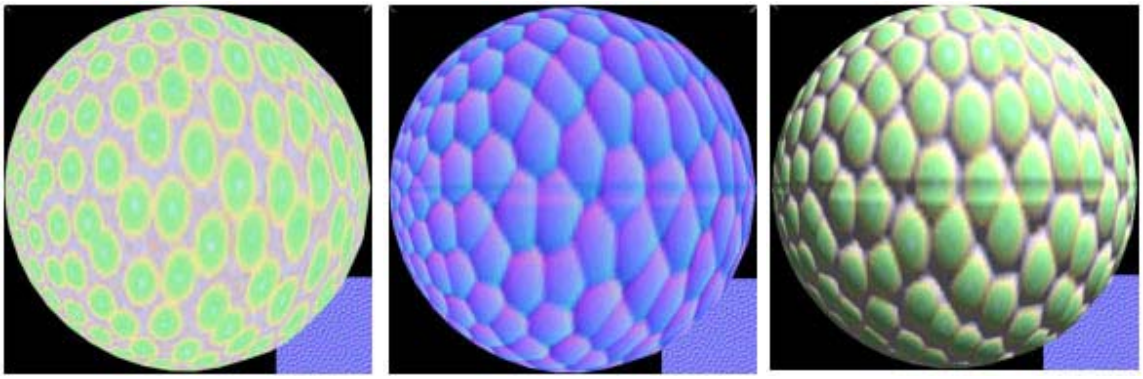
Конечная текстура с эффектом тиснения

Рисунок 11

Пиксельный бамп-мэппинг (pixel bump mapping)

Пиксельный бамп-мэппинг - на данный момент вершина развития подобных технологий. В этой технологии все вычисляется максимально честно. На вход пиксельному шейдеру дается карта нормалей, из которой берутся значения нормали для каждой точки объекта. Затем значение нормали сравнивается с направлением на источник света и вычисляется значение цвета.

Эта технология поддерживается в аппаратуре начиная с видеокарт уровня GeForce2.



Объект с основной текстурой Карта нормалей Результат бамп-мэппинга

Рисунок 11

Итак, мы увидели, каким образом можно использовать особенности человеческого восприятия мира для улучшения качества изображений, создаваемый 3D-играми. Счастливые обладатели последнего поколения видеокарт NVidia GeForce, ATI Radeon (впрочем, и не только последнего) могут самостоятельно поиграть с некоторыми их описанных эффектов, благо настройки устранения ступенчатости и анизотропной фильтрации доступны из опций драйверов. Эти и другие методы, оставшиеся за рамками данной статьи, успешно внедряются разработчиками игр в новые продукты. В общем, жизнь становится лучше. То-то еще будет!