Модуль 3. Построение реалистичных изображений

Лекция 3.1. Текстурирование

Большинство материалов реального мира не являются инвариантными к поверхности. Данный факт находит свое отражение в той характерной особенности, что зачастую даже простой, с виду однородный материал, в двух соседних точках будет иметь немного отличные друг от друга свойства. Ситуация усложняется сильнее, имей мы дело с материалами в действительности неоднородными. К примеру, узорчатый рисунок обоев на стене. В данном случае мы будем иметь дело как с изменением характеристик диффузного рассеивания, так, возможно и с разнообразными моделями освещения на различных участках поверхности. Подложка такого материала могла бы быть имитирована простейшей моделью освещения Ламберта, а вот рисунок с участием металлической краски пришлось бы имитировать чем-то более сложным.

Таким образом, мы приходим к пониманию того факта, что для визуализации конечного объекта зачастую будет недостаточно некоторой модели освещения, которая будет использоваться при растеризации. Любая модель освещения является функцией некоторого фиксированного набора параметров. Даже простейшая модель освещения Ламберта имеет один такой параметр - коэффициент диффузного рассеивания. То есть задача визуализации сложных материалов заключается в необходимости варьировать по поверхности треугольников некоторые параметры используемой модели освещения. Вопрос заключается в том, каким образом задавать такую вариацию для полигональных сеток, а вернее - как задать саму поверхность?

Появление текстуры

Подавляющее число материалов реального мира не являются инвариантными к поверхности.

Это означает, что в каждой точке поверхности вычисление финального цвета с учетом освещенности будет производится **по собственному закону**.

Если некому материалу однозначно соответствует некоторая модель освещения, в которой освещенность является функцией параметров

(a, b, c, …)

то, варьируя по поверхности значения этих параметров, можно описать сколь угодно сложный материал.

Основной вопрос – как проводить подобную вариацию.

Ситуация еще больше усложняется, если учесть возможность деформации полигональной сетки при анимации. В таком случае взаимное положение вершин внутри треугольника изменяется, следовательно, необходимо искать такие методы вариации свойств, которые могли бы учитывать столь специфичные ситуации. В данной лекции мы попытаемся дать краткий обзор методов текстурирования, призванных решать подобного рода задачи, а так же наиболее популярные техники, использующие текстуры.

Процедурные текстуры

Простейший из способов вариации свойств материала по поверхности некоторого объекта. Чтобы понять суть метода, необходимо обратиться к природе любого полигонального объекта.

Что, по сути, представляет из себя некоторый полигональный объект? Что, если рассматривать его как некоторый твердотелый объект реального мира? В таком случае в природе любой объект подобного рода представляет из себя твердую конструкцию, целиком составленную из некоторого материала, свойства которого в пространстве варьируются по некоторому закону. Для примера можно взять некоторую статую, высеченную из гранита. Гранит - один из природных материалов, вариация свойств которого в пространстве можно с определенной долей успешности приблизить некоторой математической формулой. Таким образом, имей мы для конечного фрагмента растеризации его координаты в пространстве, мы с легкостью могли бы подставить их в математическую формулу и получить величину некоторых свойств (к примеру, диффузного рассеивания), которые применили бы при расчете модели освещения.

Процедурные текстуры - это текстуры, описываемые математическими формулами. Наиболее часто встречающиеся процедурные текстуры: разные виды шума (например, fractal noise), дерево, вода, лава, дым, мрамор, огонь и т.п., то есть те, которые сравнительно просто можно описать математически. Небольшая модификация формулы позволяет в рамках одного типа материала (к примеру, дерева) получает бесконечное количество вариаций.

На следующем рисунке можно видеть пример вариаций материалов, использующих процедурное текстурирование.

Однако применение процедурных текстур ограничено случаями недеформируемых поверхностей. Действительно, если некоторая полигональная поверхность будет деформироваться, то соответствующим образом должна деформироваться и соответствующая математическая формула. Реализация же подобного явления с точки зрения математики может быть весьма неочевидна.

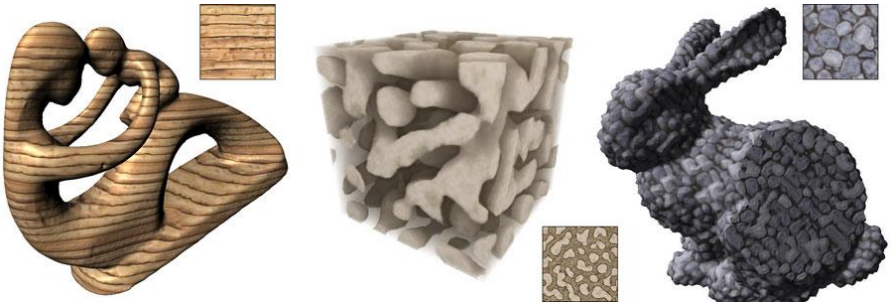
Вторым недостатком процедурных текстур является их крайняя ограниченность. Дело в том, что математическим формулам поддаются только простые материалы, обладающие свойством регулярности. Весьма сложно было бы, к примеру, задать вариации свойств диффузного рассеивания для картины да Винчи. Подобная ограниченность требует применения иных более сложных методов задания вариации свойств материалов.

Объемные текстуры (Volume textures)

Представляют собой трехмерную матрицу, содержащую в себе определенные заранее рассчитанные, либо полученные иным способом значения вариации свойств материала.

Каждый элемент матрицы соответствует некой точке пространства с известными координатами.

Таким образом, определение свойства материала конкретной точки на поверхности сводится к выбору той или иной ячейки трехмерной матрицы.



Объемные текстуры

Процедурные текстуры представляют собой весьма простой способ имитации сложных материалов, между тем имея некоторую ограниченность, связанною с требованием к математической регулярности материала. Что, если данное ограничение мы могли бы обойти? В таком случае мы могли бы имитировать любые сколь угодно сложные материалы в пространстве. Вопрос в том, как это сделать?

Именно для решения подобной задачей были изобретены объемные текстуры. Каждая из таких текстур представляет собой регулярную растровую сетку в трех измерениях. Проще говоря — это трехмерный массив заданной размерности, в ячейках которого находятся свойства материала в данной точке пространства.

Как и в случае с процедурными текстурами, задача по определению свойств фрагмента сводится к извлечению свойств по трехмерным его координатам. Только если в предыдущем случае нам приходилось подставлять координаты в некоторую математическую формулу, в данном случае нам придется извлекать *сэмпл* (от sample - образец).

Суть *семплирования* заключается в следующем: для некоторых координат необходимо определить свойства текстуры по известным значениям, зафиксированным в ячейках растра. Как не трудно заметить, координаты фрагмента не всегда будут точно совпадать с координатами некоторой ячейки растра. В общем случае мы будем иметь дело с четырьмя соседними ячейками растра, и задача по определения влияния каждой из этих ячеек на точку пространства может быть не так проста, как кажется на первый взгляд. Задача по определению свойств материала при известных узлах растровой сетки

является задачей интерполяции, а в контексте текстурирования - задачей *ресемплинга.*

Итак, для произвольных координат в пространстве, проведя семплирование (получение сэмпла), мы можем определить свойства материала. Это позволяет нам имитировать весьма сложные материалы в пространстве, как обладающие регулярностью, так и нет. Однако данный способ, имея некоторые плюсы по сравнению с процедурным текстурированием, имеет ряд весьма весомых минусов.

Первый огромнейший недостаток - это объемы памяти, необходимые для представления такой текстуры. Представим себе ситуацию, когда нам необходимо имитировать свойства кирпичного здания, размерами 10x10x10 метров. При этом мы будем использовать уровень детализации растра в ячейку на миллиметр (что, вообще говоря, может оказаться недостаточным при детальном рассмотрении). Тогда, используя простейшую арифметику, мы получаем количество ячеек, равное 1000000000000. При том, что каждая ячейка потребует более одного байта, объемы памяти, необходимые для хранения подобной текстуры будут нам предоставлены еще очень не скоро даже при нынешней тенденции к увеличению ее объемов.

Вторым недостатком, характерным для объемных текстур является неочевидность способа создания таких текстур. Если в случае процедурных текстур достаточно привлечения гениального математика, то в случае текстур объемных даже гениальный художник может встать в тупик, поставь перед ним задачу описания свойств материала в пространстве при помощи растра.

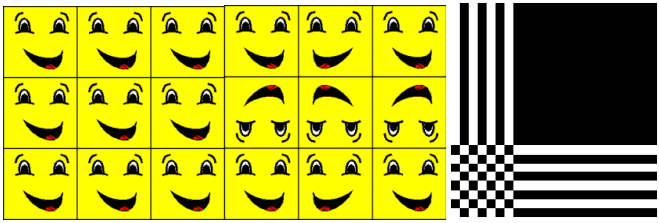
Повторение (Tiling)

Подобный подход призван снизить нагрузку на память, используя небольшую область текстуры в качестве базы и повторяя ее при необходимости в пространстве. Разумеется, что данный подход будет работать только в том случае, когда некоторый объект будет проявлять свойство регулярности.

Предположим, мы имеем дело с кирпичной стеной. На локальном участке текстура может имитировать весьма разнообразные свойства

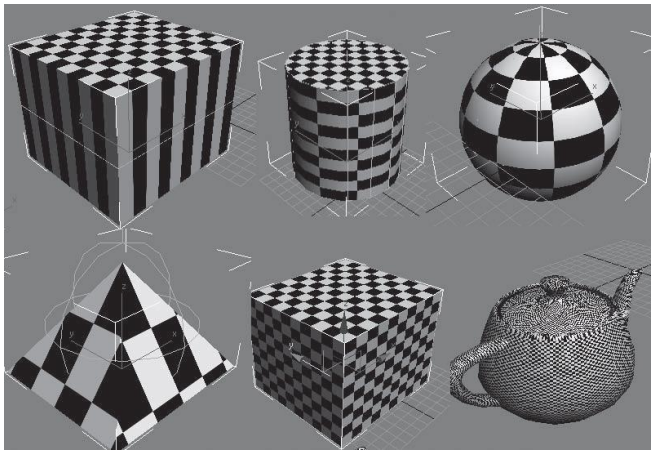
материала без учета регулярности структуры, однако подобная регулярность может прослеживаться в более глобальном масштабе. К примеру, мы могли бы имитировать свойства материала на участке 1x1x1 метр, а затем распространить методом повторения свойства материала на соседние участки. Разумеется, противоположные края имитируемого участка должны точно повторять друг друга. Пример такого подхода продемонстрирован на рисунке ниже.

Режимы повторения



Выделяют три популярных режима повторения:

– Wrap
– Mirror
– Clamp



Плоское текстурирование

Объемное и процедурное текстурирование в пространстве обладают весьма неприятным недостатком, происходящим из попытки описать свойства материала в объеме. Ведь конечная полигональная модель не имеет объема. Вдобавок, нас интересуют свойства материала лишь на плоской поверхности полигонов. Имитация материала внутри модели, равно как и за ее пределами является задачей избыточной. Вдобавок, пространственная функция не может учитывать деформации объекта. С другой стороны, Каждый растеризуемый фрагмент принадлежит соответствующему полигону. Если бы мы могли увязать свойства данного фрагмента с вариацией свойств только на поверхности данного полигона, мы одновременно решили бы и проблемы деформации объекта, и проблему избыточности данных.

Плоское или двумерное текстурирование решает задачу вариации свойств материала только на поверхности объекта. Однако, полигоны трехмерны. В вышеописанных методах текстурирования свойства конкретного фрагмента были функцией трехмерной системы координат. В нашем же случае, свойства фрагмента так же должны быть функцией некоторой системы координат.

Логично было бы предположить, что каждый фрагмент внутри полигона описывался бы некоторыми двумерными координатами, но что это за координаты еще предстоит решить.

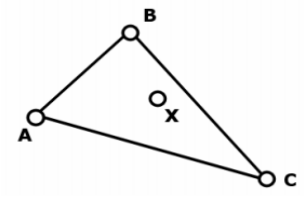
Барицентрические координаты

Пусть некоторый треугольник в пространстве представляется как набор вершин A, B и C. Тогда координаты любого фрагмента X внутри треугольника можно получить как линейную комбинацию вершин

$$aA + bB + cC, \text{ где } a+b+c=1.$$

Так как три коэффициента всегда дают единицу, для каждого фрагмента треугольника мы можем перейти к всего двум координатам (a,b).

Таким образом, мы перешли к двумерной функции, что делает возможным наличие некоторой плоской функции или плоского двумерного растра для вариации свойств на поверхности треугольника.



Однако подобный подход, очевидно, породит следующие проблемы:

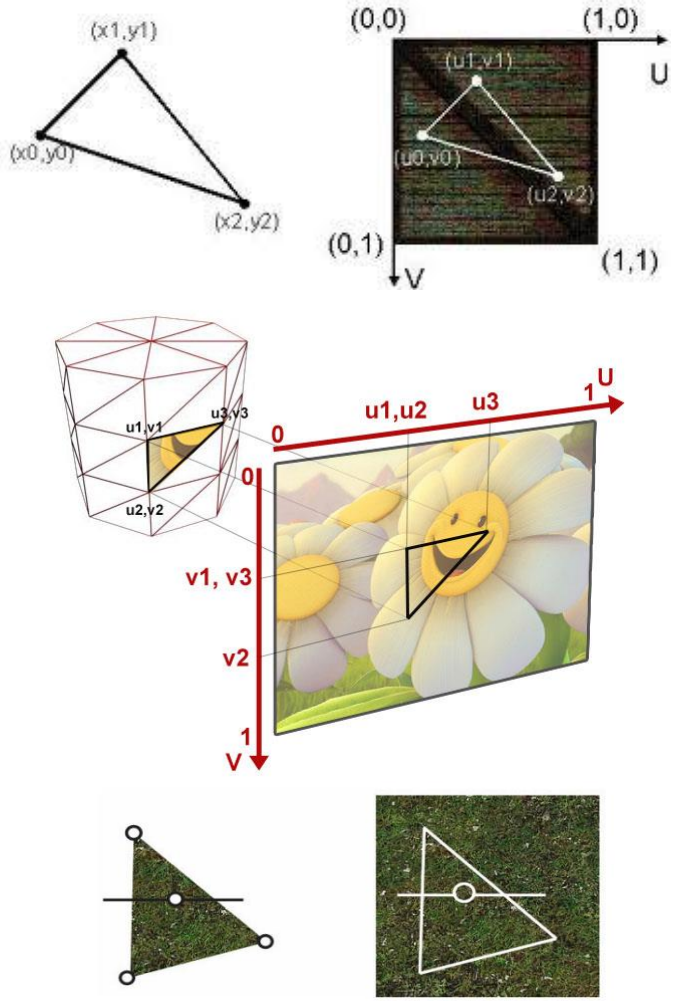
Представим себе два треугольника, имеющих общее ребро. Представим себе некоторый фрагмент, располагающийся на этом ребре. Данный фрагмент будет одновременно принадлежать обоим треугольникам, но при этом внутри каждого будет иметь две разных системы координат. При этом мы помним, что свойства этого фрагмента вне зависимости от систем координат должны быть одинаковыми в обоих случаях. Таким образом, мы приходим к необходимости совмещения двух математических функций, базирующихся на различных системах координат. Задача усложняется еще и тем, что в конечной полигональной сетке множество треугольников будут вступать друг с другом во множественные связи разделения ребра, а таких треугольников может быть несколько тысяч и даже миллионов.

Из-за того, что мы практически не имеем возможности создать одну функцию для двух соседствующих треугольников, которая решала бы проблемы стыков, мы приходим к необходимости задания независимых функций для каждого отдельного треугольника, либо задания независимых растров, что является задачей *крайне* трудоемкой, если не неразрешимой.

Текстурные координаты

Вспомним, что нам известно о фрагментах треугольника. Каждый фрагмент генерируется на этапе растеризации. При этом сами координаты конечного фрагмента G являются линейной функцией координат E и F. При этом E - линейная функция A и B, а F - A и C. Линейные закономерности позволяют интерполировать любые величины. То есть, если мы назначим вершинам A, B и C значения некоторой величины веса, то вес фрагмента G будет вычислен как последовательность линейных интерполяций.

Теперь представим себе, что у нас есть некоторая поверхность, на которой существует система координат (u,v). На этой поверхности при помощи математической формулы либо при помощи растровой сетки мы можем задать вариацию свойств некоторого материала. Тогда для задания вариаций свойств на поверхности некоторого треугольника в пространстве нам достаточно однозначно ассоциировать с этим треугольником некоторый треугольник на поверхности материала (текстуре). Сделать это можно, задав каждой вершине треугольника пару текстурных координат (u,v). Тогда задача по определению свойств конкретного фрагмента заключается в определении его координат через последовательность линейных интерполяций и последующем извлечении сэмпла по этим координатам с поверхности материала.



Традиционно для плоского текстурирования используются растровые сетки, а не математические формулы. И хотя процедурное текстурирование в двумерном пространстве так же представляется возможным, из-за ограниченности материалов, удовлетворяющих свойству регулярности, используется оно только в частных случаях. В связи с этим понятие *текстуры* обычно связывается именно с двумерным растром.

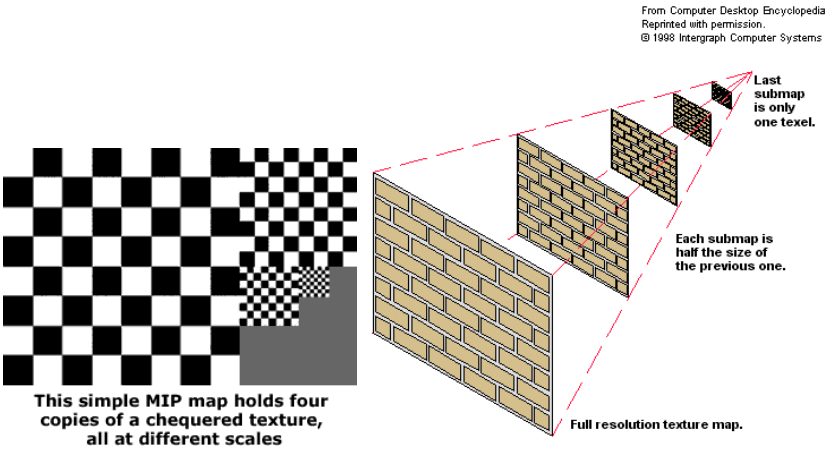
Традиционно координаты на растре принято именовать как (u,v). При это ось u отвечает за горизонтальное направление на растре, а ось v - за вертикальное. При этом тектстура любого размера в координатах (u,v) имеет размер 1x1, а отсчет по осям производится в диапазоне [0; 1].

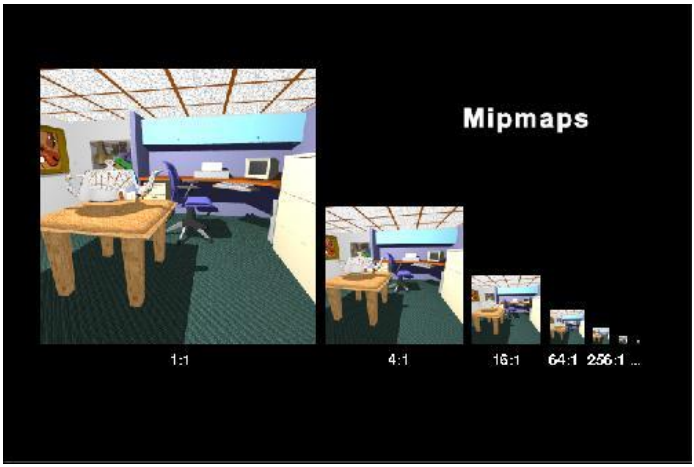
Mip mapping

MIP-текстурирование (англ. *MIP mapping)* — метод текстурирования, использующий несколько копий одной текстуры с разной детализацией. Название происходит от лат. *multum inparvo* — «много в одном».

Изображение лучше всего выглядит, когда детализация текстуры близка к разрешению экрана. Если разрешение экрана высокое (текстура слишком маленькая/объект очень близко), получается размытое изображение. Если же разрешение текстуры слишком высокое (текстура слишком велика/объект очень далеко), получаем случайные пиксели — а значит, потерю мелких деталей. Получается, что лучше иметь несколько текстур разной детализации и накладывать на объект ту, которая наиболее подходит в данной ситуации.

Создаётся так называемая *MIP-пирамида* — последовательность текстур с разрешением от максимального до 1^x1. Например: 1^x1, 2^x2, 4^x4, 8^x8, 16^x16, 32^x32, 64^x64, 128^x128, 256^x256, 512^x512 и 1024^x1024. Каждая из этих текстур называется *MIP-уровнем* (англ. *MIP level)* или *уровнем детализации* (англ. *level of detail*).





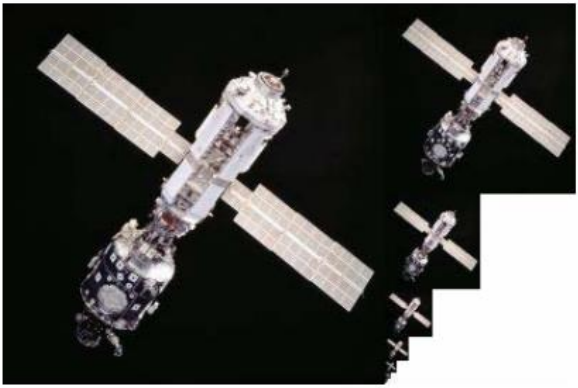
На всех этих текстурах находится одно и то же изображение. Таким образом, MIP- текстурирование увеличивает расход памяти на треть.

При наложении текстур вычисляется расстояние до объекта, соответственно находится номер текстуры как

$$miplevel = \log_2\left(\frac{scale \cdot dist}{resolution}\right) + mipbias$$

где *resolution* — разрешение виртуальной камеры (количество пикселей, которое будет в объекте размером в 1 ед., расположенном в 1

ед. от камеры), *scale* — масштаб текстуры (размер текселя в единицах трёхмерного мира), *dist* — расстояние до объекта в тех же единицах, *mip bias* — число, позволяющее выбирать более или менее детальную текстуру, чем даёт формула. Эта цифра округляется до целого, и текстура с соответствующим номером (нулевая — самая детальная, первая — вдвое меньшая и т.д.) накладывается на объект.



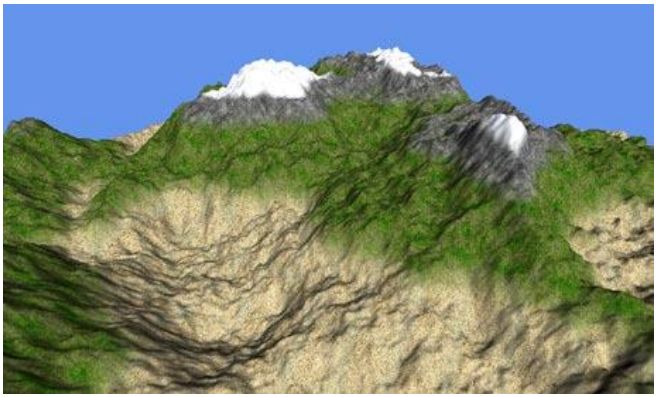
MIP-текстурирование не решает проблему текстур, находящихся под острым углом к зрителю. У таких текстур разрешение по одной оси сильно отличается от разрешения по другой — и, например, по оси X изображение явно размыто, в то время как по оси Y видны мерцания, свойственные завышенному разрешению текстуры. Наконец, видна чёткая граница между MIP-уровнями. Это решается трилинейной фильтрацией.

Мультитекстурирование

Как мы уже убедились, при помощи текстур можно задавать вариацию достаточно большого числа свойств на поверхности. На данный момент необходимо четко определиться: текстура не обязательно должна являться некоторым изображением, которое будет накладываться на объект. И хотя такой подход является частным случаем применения текстуры, изначально текстуры созданы именно для вариации каких-либо свойств, а таких свойств может быть достаточно много.

Ярким примером имитации множества независимых свойств на поверхности материала являются эффекты мультитекстурирования. Суть всех эффектов сводится к тому, что для некоторой поверхности наличествует определенное количество свойств, вариацию которых нельзя описать единой текстурой. Примером такой ситуации могут являться случаи, когда для двух различных свойств поверхности необходимо использовать различные системы координат.

Рассмотрим три наиболее распространенных эффекта, достигаемых мультитекстурированием.



Декали (Decals)

Данный эффект используется для нанесения на поверхность некоторого объекта дополнительного рисунка. В видеоиграх это могут быть всевозможные напыления, граффити, следы крови или дыры от выстрелов.

Суть заключается в том, что для полигонального объекта одновременно используются две текстуры с различными системами координат - текстура самой поверхности и текстура рисунка. Предположим, что нам необходимо нанести на стену пятна крови.

В качестве основной текстуры поверхности возьмем текстуру кирпичной стены. Тогда текстура рисунка (декали) должна представлять из себя растровое изображение с прозрачными областями. В тех местах, где тексель (texel - texture element) декали будет прозрачным, мы будем считать конечным свойством свойство

основной текстуры. В случае непрозрачного текселя декали мы будем брать за основное свойство - свойство декали.

Таким образом, для вычисления свойства поверхности в некоторой точке достаточно применить следующую формулу:

Result = Decal * alpha + Base * (1 - alpha),

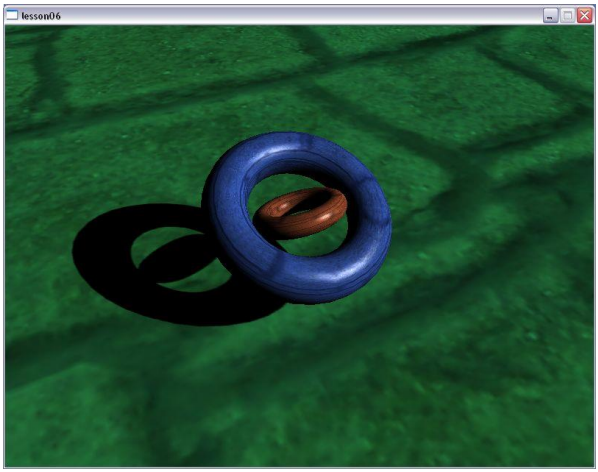
Где Decal - это свойство декали в данном фрагмента, Base - свойство основного материала стены, а alpha - прозрачность материала декали.

Теневые карты (Shadow Maps)

Предположим, что мы имеем ту же кирпичную стену и вариацию материала на ней, выраженную текстурой. При этом для всей поверхность мы имеем карту освещенности, в которой записаны величины яркости, достигающие каждого из фрагментов стены. В таком случае мы имеем дело со следующей ситуацией:

Result = Base * Shadow,

Где Base - цвет, полученный для данного фрагмента с использованием текстуры стены, а Shadow - величина в диапазоне [0;1], описывающая достижимость световыми лучами конкретного фрагмента.



Детали (Detail)

Часто возникают ситуации, при которых одна текстура используется для покрытия большой площади полигональной модели. В случае приближения наблюдателя к такой поверхности, из-за низкой детализации текстуры поверхность начинает выглядеть размытой. Однако существует сравнительно простой способ искусственного повышения детализации за счет использования карты деталей. Такая карта является текстурой, в которую записаны отклонения в большую или меньшую сторону от результата, полученного при семплирования

свойств основного материала. Следующая иллюстрация демонстрирует получаемый эффект:

Result = Base + Detail

Детализация геометрии за счет текстур

Часто возможна такая ситуация, когда конечный полигональный объект должен обладать высоким уровнем детализации, но количество полигонов, которое потребовалось бы в этом случае, было бы настолько велико, что это неизбежно повлекло бы за собой потери производительности на операциях с вершинами.

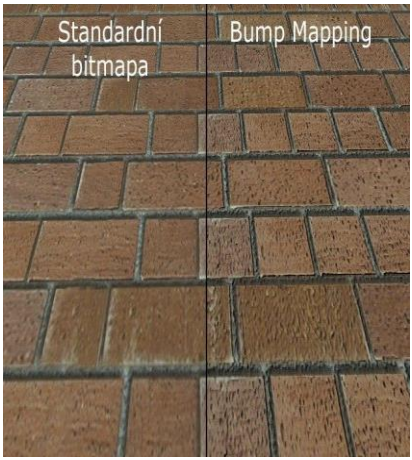
На ранних этапах развития машинной графики такие ситуации обходились весьма просто. Мелкие детали рельефа поверхности заносили в текстуру художниками. Каждый из них, создавая текстуру, прорисовывал на тени от глаз, складки на одежде и мелкие детали в виде пуговиц. Подобный подход обладает достаточной эффективностью, однако не трудно заметить, что он практически не интерактивен к изменению условий освещения. Действительно, если объем мелкого элемента создается за счет его взаимодействия со светом, условия освещения изменились, а сама освещенность осталась неизменной, это приводит к падению уровня реализма визуализации.

На данный момент существует три основных способа борьбы с повышением полигональной детализации за счет текстур, которые до недавнего времени были недоступны по причине ограниченности производительных мощностей. Однако на сей день аппаратное обеспечение стало достаточно эффективным, чтобы воплотить эти методы в жизнь в том числе в реальном времени.

Высотные карты (Heightmaps)

Все эффекты повышения детализации за счет текстур основываются на наличии некой карты высот. По сути, это такая же текстура, как и та, что описывает вариацию свойств по поверхности, за тем исключением, что в качестве варьируемого ею свойства выступает изменение высоты на поверхности в направлении нормали. То есть, таким образом будут достигаться эффекты выдавливания и вдавливания поверхности, с помощью которых можно достичь эффекта повышения детализации.

Типичная высотная карта представляется текстурой в серых тонах. При этом чем светлее какой-либо пиксель этой текстуры, тем выше будет сдвигаться в направлении нормали соответствующий ему фрагмент. Карта высот может быть сгенерирована различными средствами, и она является входным параметром для трех основных семейств алгоритмов повышения детализации за счет текстур.



Parallax mapping

Самый прогрессивный метод детализации, ставший доступным лишь последние несколько лет. Идея заключается в том, что, имея рельефную поверхность и определенный угол наблюдения, в точке A соответствующей плоской поверхности, мы в действительности будем видеть не ее, а точку B. Зная угол наблюдения, а так же все параметры поверхности, мы можем вычислить смещение и взять сэмпл в других координатах.

Подобный подход менее производителен, нежели bump mapping, однако он интерактивен не только к изменению условий освещения, но и так же к изменению угла наблюдения. Уровень реализма, достигаемый подобным алгоритмом, делает его наиболее привлекательным с точки зрения повышения детализации за счет текстур.

Впервые в видеоиграх данный алгоритм был использован в игре F.E.A.R. для имитации углублений в стенах от выстрелов. Так же на настоящий момент он активно используется всеми производителями видеоигр. Эффекты, достигаемые им, продемонстрированы на следующем рисунке: