LOBACHEVSKY STATE UNIVERSITY
of NIZHNI NOVGOROD
National Research University

Computing Mathematics and Cybernetics faculty

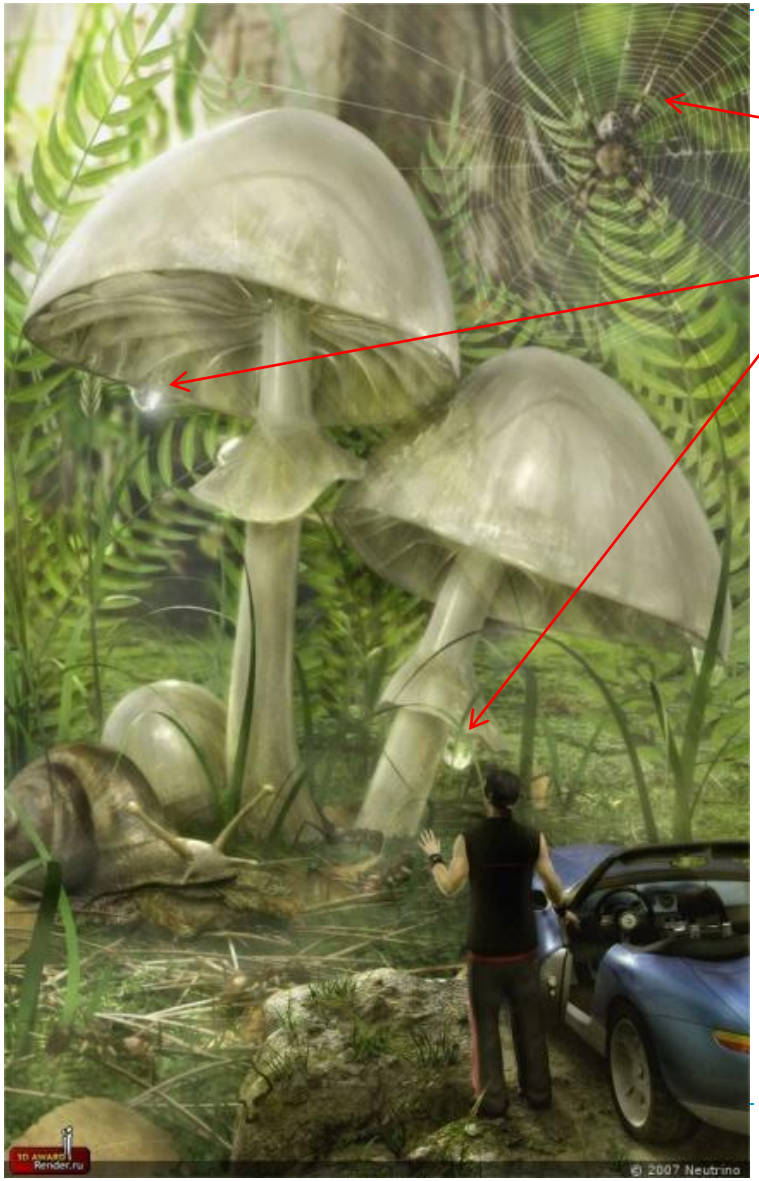
Software department

CS255. **Computer Graphics** Introduction Course

Texture mapping & anti-aliasing

Prof. Vadim E. Turlapov
TA Alexandra Belokamenskaya
TA Svetlana Nosova

Additional scene details are achieved without complicating the object's geometry



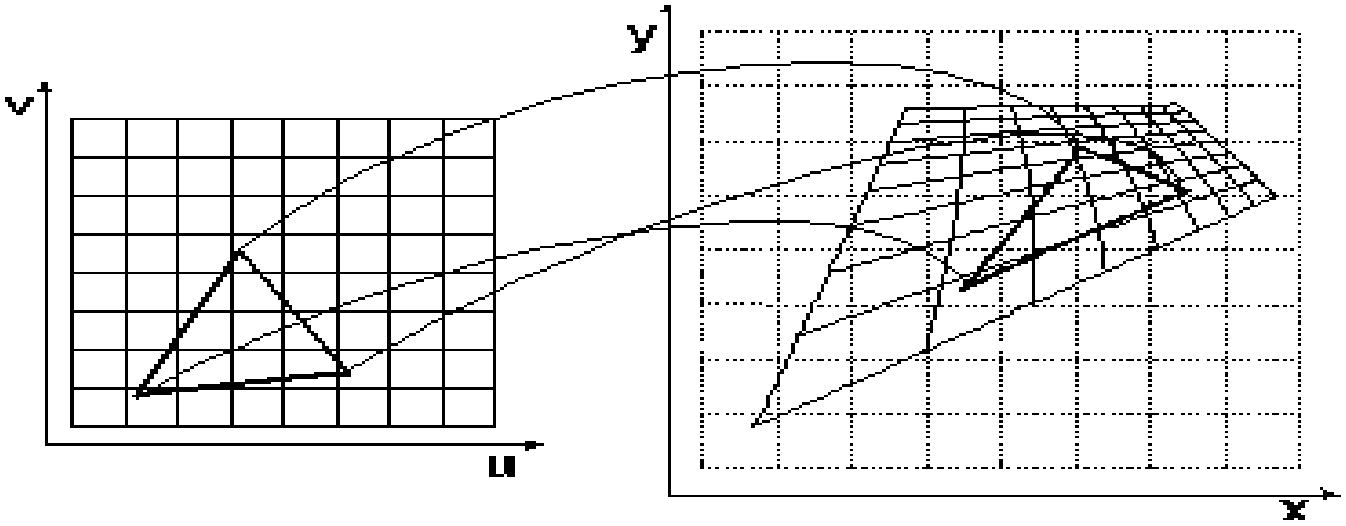
Background, Spiderweb

Water drops

Методы текстурирования (MIP-mapping, bump-mapping), фильтрации, сглаживания

Текстурирование. Наложение текстуры или текстурирование (texture mapping) - это метод, посредством которого на поверхность объекта накладывается некоторое изображение, называемое изображением текстуры.
В общем контексте графического конвейера этот метод открывает огромные возможности, но простота идеи метода наложения текстуры весьма обманчива.

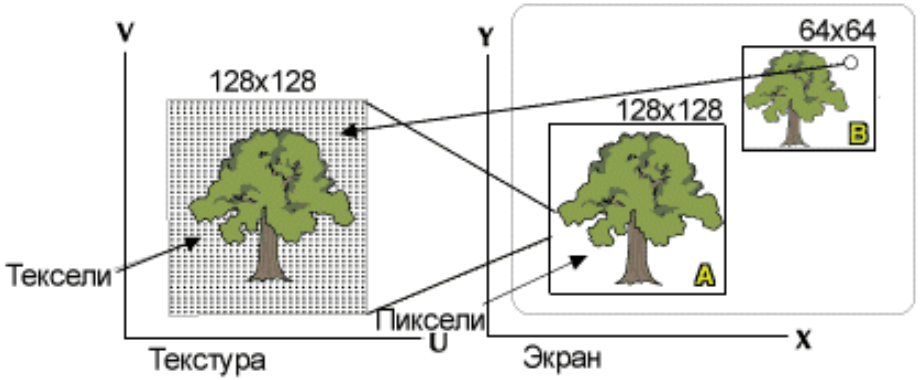
Технология текстурирования заключается в проецировании изображения (текстуры) на трехмерную поверхность. Таким образом обеспечивается дополнительная **детализация 3D объекта без усложнения его геометрии.**



Когда изображение используется в качестве текстуры, накладываемой на 3D примитив, проявляется множество разнообразных ошибок визуализации, называемых артефактами.

Текстурирование. Сэмплинг и его артефакты

Сэмплинг (point-sampling) – простейший метод текстурирования, в котором текселы непосредственно переносятся в пикселы изображения с учетом масштаба. Методу присущ серьезный артефакт: когда наблюдатель приближается вплотную к текстурированной поверхности, происходит пикселизация. Для избежания этого артефакта используют методы текстурирования, основанные на фильтрации текстур.

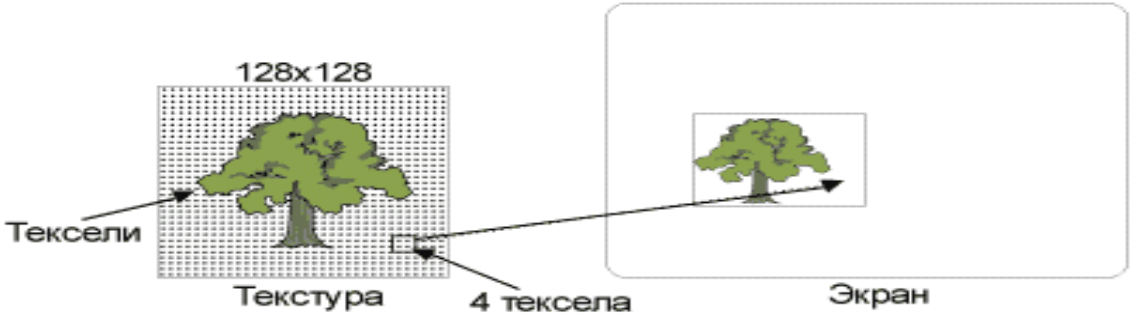


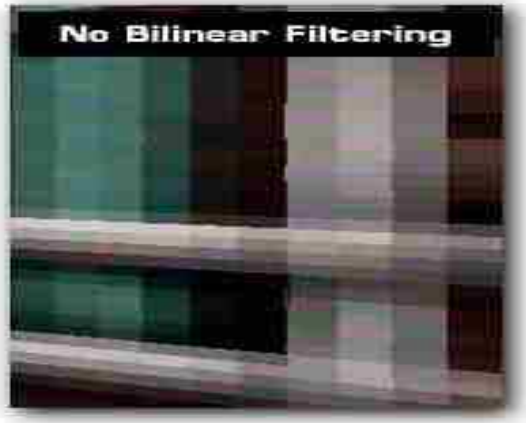
A: однозначное соответствие между текселами и пикселами
B: pixel 0:texel 0; pixel 1:texel 2; pixel 2:texel 4...

Артефакты и билинейная фильтрация

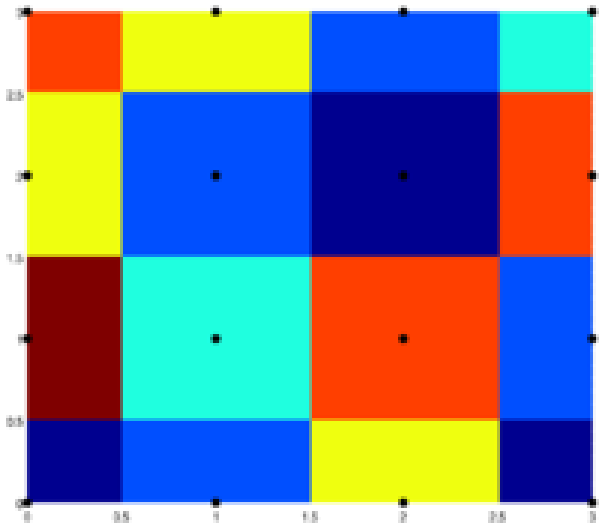
Bi-linear filtering - это техника устранения искажений изображения (фильтрация), таких, как "блочность" текстур при их увеличении. При медленном вращении или приближении/ удалении объекта могут быть заметны "перескакивания" пикселов с одного места на другое, т.е. появляется блочность.

Во избежании этого эффекта применяют билинейную фильтрацию, при использовании которой в качестве цвета каждого пиксела берется взвешенное среднее значение (линейная интерполяция) цвета четырех смежных текселов (верхний рис.). Результирующий цвет пиксела определяется в результате трех операций смешивания: сначала смешиваются цвета двух пар текселов по x, а потом смешиваются два полученных цвета по y. Результат показан на нижнем рис.

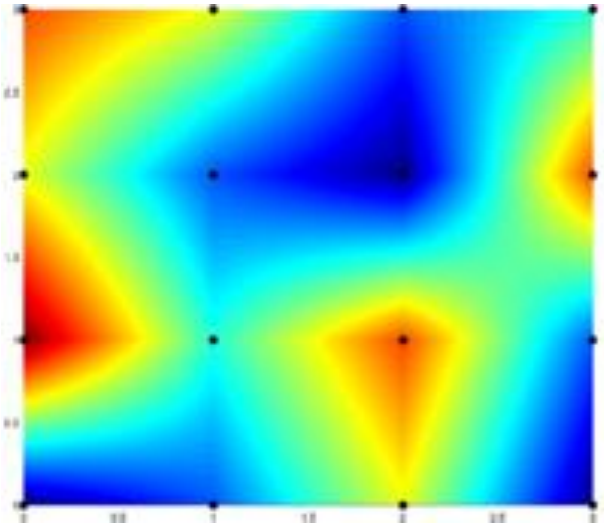




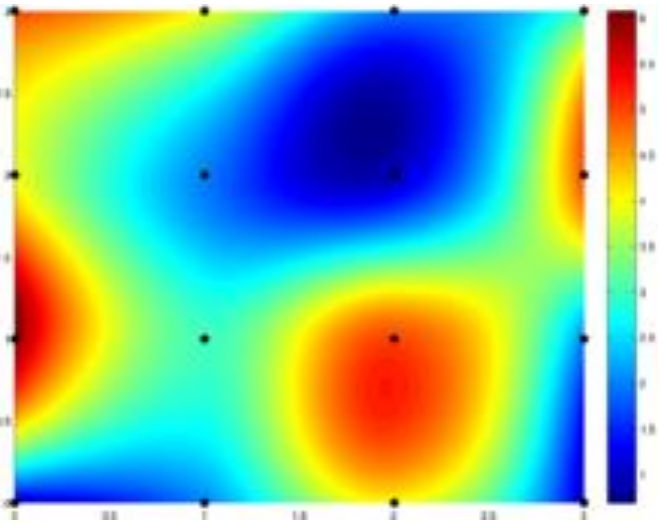
Свойства билинейной интерполяции



Интерполяция методом ближайшего соседа

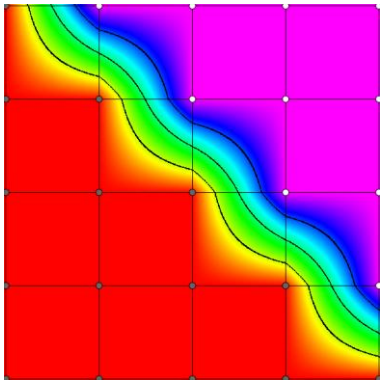


Билинейная интерполяция. Частные производные терпят разрыв на границах квадратов

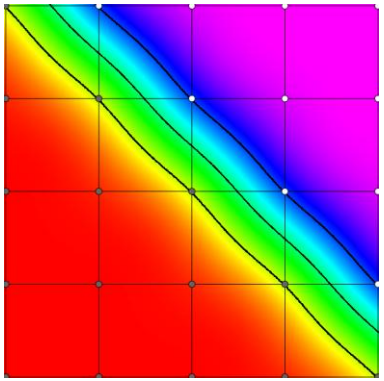


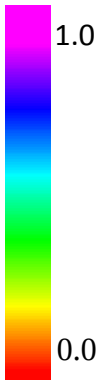
Бикубическая интерполяция (ru.wikipedia.org)

Билинейная интерполяция



Бикубическая интерполяция



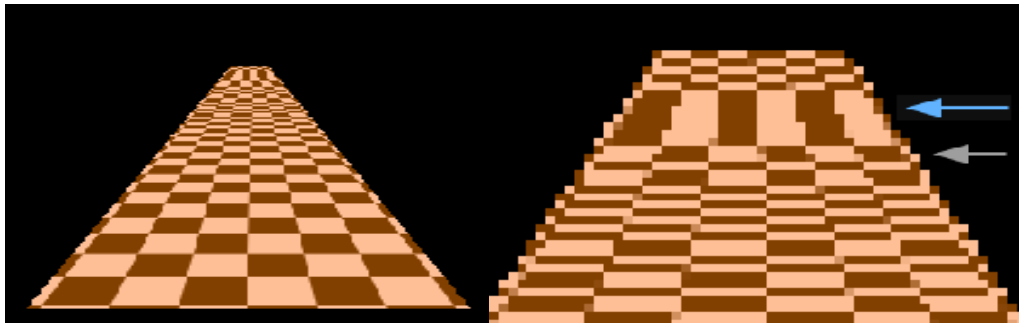


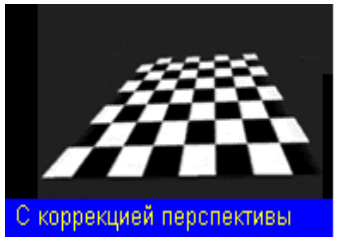
Артефакты depth aliasing и перспективная коррекция

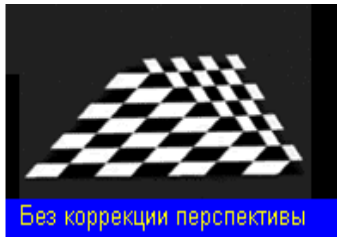
Существует класс артефактов наложения текстур известный под названием "**depth aliasing**" (ошибки глубины сцены или Z-aliasing), от которых билинейная фильтрация не избавляет и не может избавить.

Ошибки "depth aliasing" возникают из-за того, что объекты более отдаленные от наблюдателя, выглядят более маленькими на экране. Если объект двигается и удаляется, то наложенное текстурное изображение становится все более и более сжатым. В конечном счете появляются ошибки визуализации. Эти ошибки визуализации **особенно нежелательны в анимации,** где такие артефакты становятся причиной мерцания и эффекта медленного движения в той части изображения, которая должна быть неподвижной. Как только вертикальная сторона квадрата (высота) сокращается до двух пикселей (см. напротив голубой стрелки), появляются артефакты "depth-aliasing" - несколько квадратов сливаются в один.

Одним из способов устранения depth aliasing, имеющим и самостоятельное значение, является **перспективная коррекция**. Перспективная коррекция – ресурсоемкая процедура (одна операция деления на каждый пиксел), поэтому 3D-ускорители реализуют ее аппаратно. Но разные ускорители достигают разного качества перспективной коррекции.



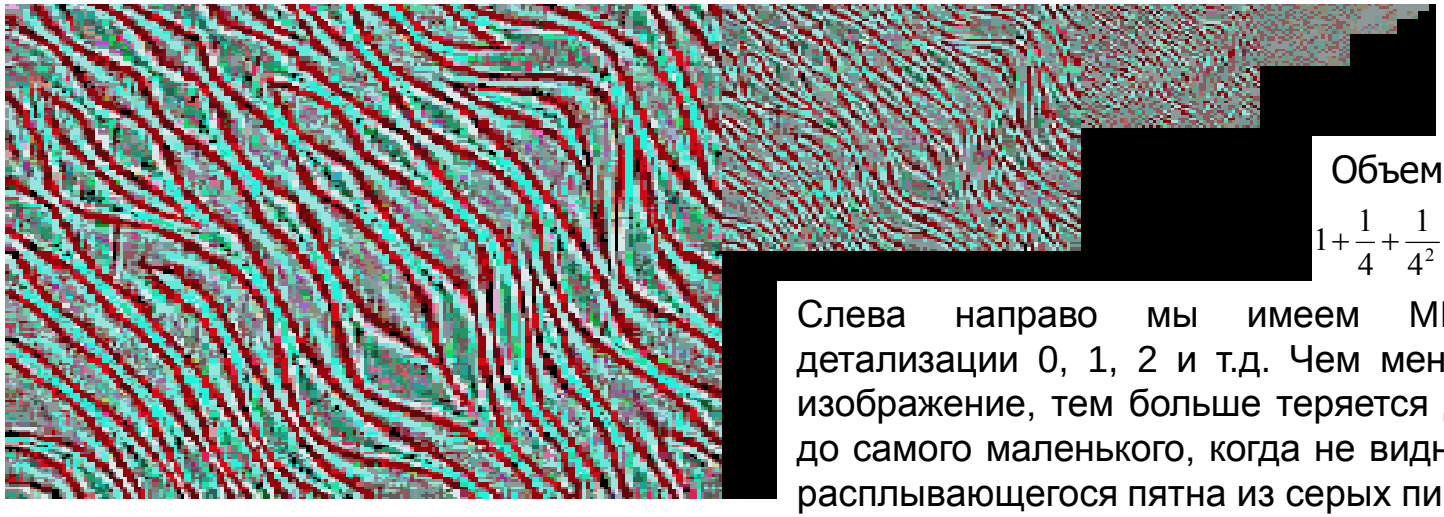




Алгоритм

Depth aliasing и MIP-mapping

Для избежания ошибок "depth aliasing" и имитации того факта, что объекты на расстоянии выглядят менее детализированными, чем те, что находятся ближе к точке наблюдения, используется техника, известная как ***MIP-mapping***. Mip-mapping - наложение текстур, имеющих разную степень или уровень детализации, когда в зависимости от расстояния до точки наблюдения выбирается текстура с необходимой детализацией. Mip-текстура (mip-map) состоит из набора заранее отфильтрованных и масштабированных изображений. В изображении, связанном с уровнем mip-map, пиксель представляется в виде среднего четырех пикселей из предыдущего уровня с более высоким разрешением. Отсюда, изображение связанное с каждым уровнем mip-текстуры в четыре раза меньше по размеру предыдущего mip-map уровня.



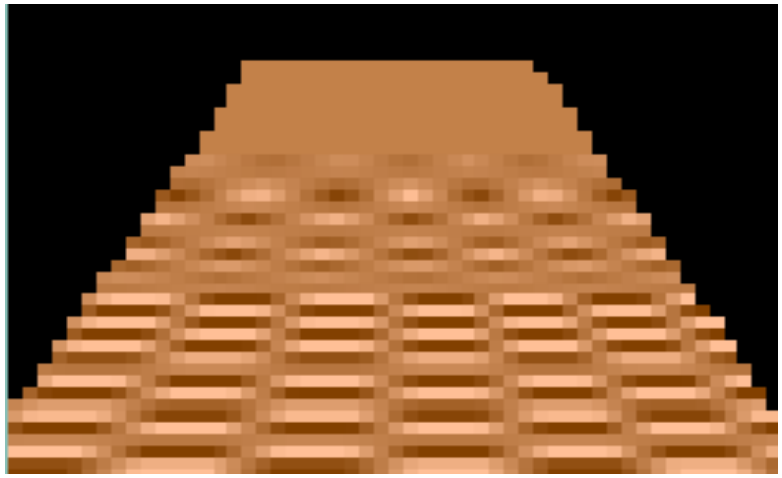
Объем MIP-текстуры:

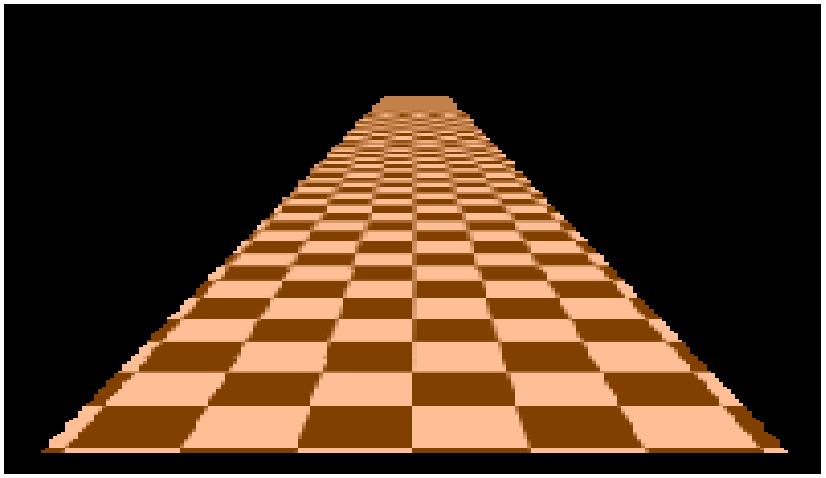
$$1+\frac{1}{4}+\frac{1}{4^2}+\frac{1}{4^3}+\ldots\leq\frac{1}{1-1/4}=\frac{4}{3}$$

Слева направо мы имеем MIP-map уровни детализации 0, 1, 2 и т.д. Чем меньше становится изображение, тем больше теряется деталей, вплоть до самого маленького, когда не видно ничего, кроме расплывающегося пятна из серых пикселей.

MIP-mapping и уровни детализации (LOD)

Степень или уровень детализации - Level of Detail или просто LOD, используются для определения, какой mip-map уровень (или какую степень детализации) следует выбрать для наложения текстуры на объект.



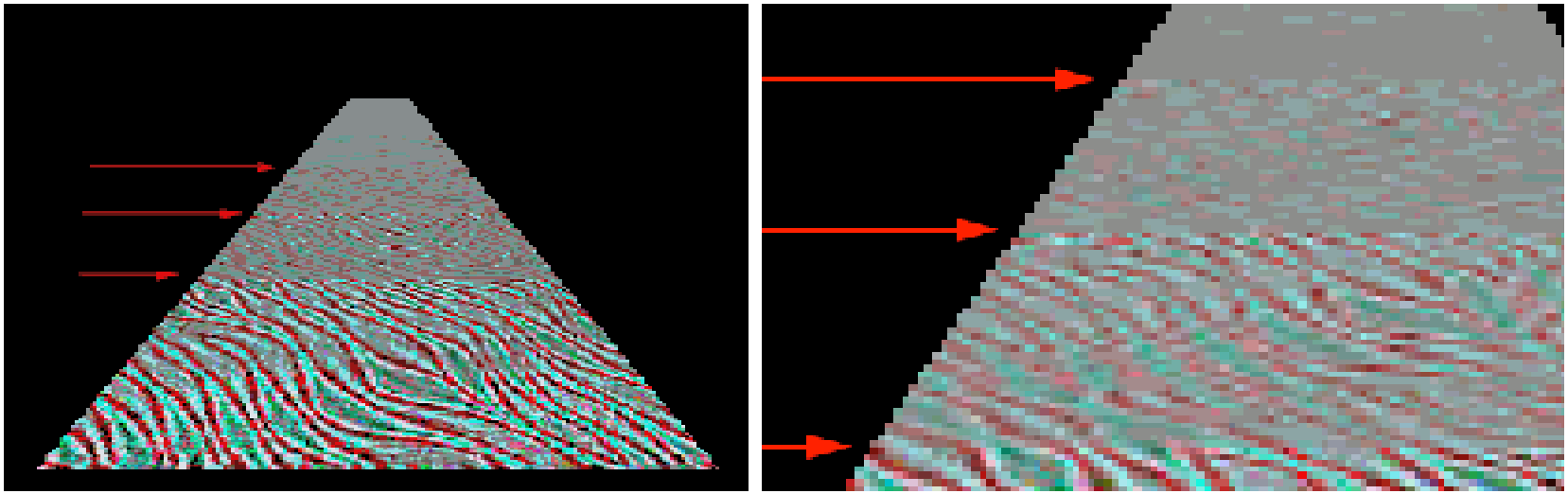


Per-polygon mip-mapping (mip-текстурирование по каждому полигону). LOD вычисляется лишь раз для всего треугольника, следствием использования этого значения для всех пикселов треугольника становится эффект растрескивания (на рис. слева), когда некоторые треугольники, из которых состоит анимированный объект, вдруг внезапно становятся чрезмерно размытыми или с неровностями.

Per-pixel mip-mapping (попиксельное mip-текстурирование). Значение LOD вычисляется для каждого пиксела. В результате предотвращается появление ошибок визуализации и излишней размытости (на рис. справа).

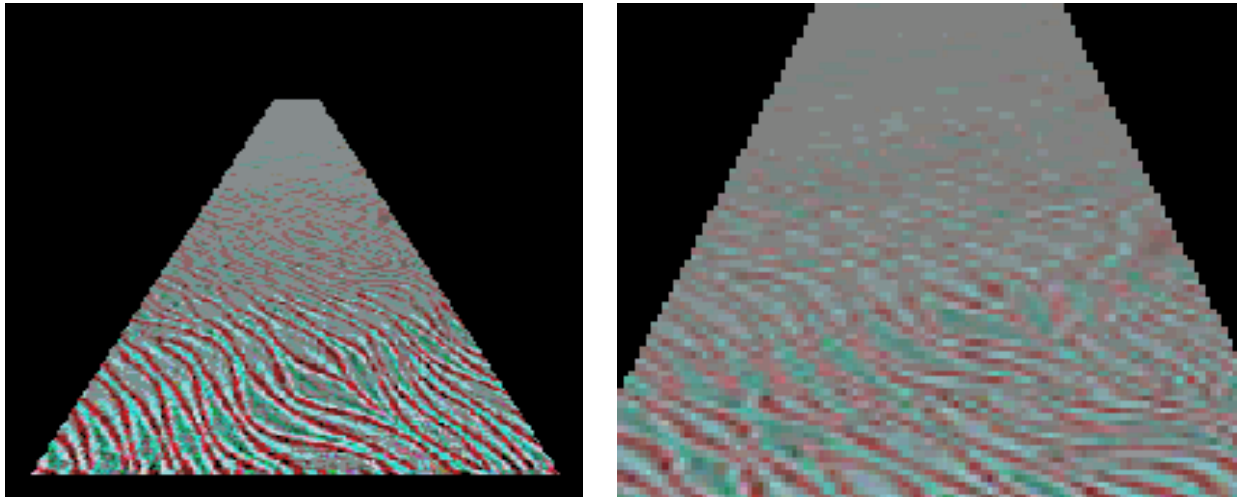
MIP-текстурирование и mip-banding (полосатость)

В то время как mip-текстурирование решает проблему ошибок "depth-aliasing", его использование может стать причиной появления других артефактов. При удалении объекта от точки наблюдения, происходит переход от низкого mip-map уровня (соответствующего изображению с высокой детализацией) к высокому mip-map уровню (соответствующего изображению с высокой степенью фильтрации и низкой детализацией). В момент нахождения объекта в переходном состоянии от одного mip-map уровня к другому, появляется особый тип ошибок визуализации, известных под названием "mip-banding" (мип-бендинг) – полосатость или слоеность, т.е. явно различимые границы перехода от одного mip-map уровня к другому.



MIP-banding и трилинейная фильтрация

Трилинейная фильтрация (tri-linear filtering) представляет собой технику, которая удаляет артефакты "mip-banding", возникающие при использовании mip-текстурирования. При трилинейной фильтрации берется блок из четырех текселей и находится среднее (интерполированное) значение цвета, как при билинейной, затем берется такой же четырехтексельный блок из соседнего mip-map уровня и так же усредняется. И после всего этого находится среднее значение между двумя полученными, которое и будет цветом обрабатываемого пикселя. Эффективно удаляются линии, которые возникают на стыках mip-map уровней при билинейной фильтрации (см. рис). Трилинейная фильтрация требует дополнительных ресурсов, т.к. реализуется одновременной обработкой двух уровней текстур.



MIP-текстурирование. Расчет LOD и фильтрация

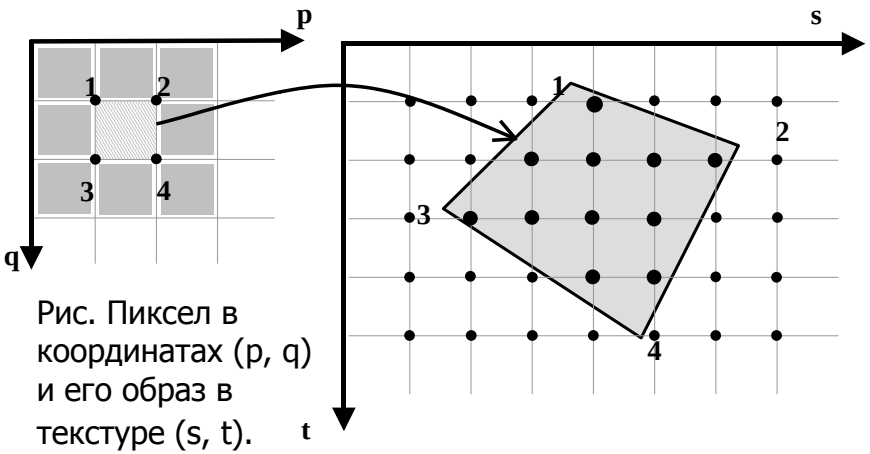


Рис. Пиксел в координатах (p, q) и его образ в текстуре (s, t).

При наложении текстуры методом mipmapping'а первая задача – определить подходящий уровень детализации LOD, чтобы размер образа экранного пиксела на текстуре и размер текселя текстуры были максимально близки.

Рассмотрим пиксел как квадрат со стороной=1. Его образ в координатах (s, t) произвольный четырех-угольник. Изменение s вдоль образа горизонтальной стороны пиксела определяется величиной $\partial s/\partial p$ рассчитанной в центре пиксела.

Изменение t вдоль образа горизонтальной стороны пиксела аналогично приблизительно равно $\partial t/\partial p$.

Заменив p на q получим соотношения для вертикальной стороны пиксела ($\partial s/\partial q$, $\partial t/\partial q$).

Производные обычно приближаются конечными разностями значений на сторонах пикселов.

В результате – варианты оценки размера пиксела (для комбинирования):

- $BLUR: LOD = MAX\left(\sqrt{\frac{\partial s}{\partial p}^2 + \frac{\partial t}{\partial p}^2}, \sqrt{\frac{\partial s}{\partial q}^2 + \frac{\partial t}{\partial q}^2}\right)$ – слегка размытое изображение;
- $SHARP: LOD = MIN\left(\sqrt{\frac{\partial s}{\partial p}^2 + \frac{\partial t}{\partial p}^2}, \sqrt{\frac{\partial s}{\partial q}^2 + \frac{\partial t}{\partial q}^2}\right)$ – изображение резкое, контрастное и

MIP-текстурирование. Расчет LOD и фильтрация

$$LOCHEED : LOD = MAX\left(\sqrt{\frac{\partial s}{\partial p}^2 + \frac{\partial s}{\partial q}^2}, \sqrt{\frac{\partial t}{\partial p}^2 + \frac{\partial t}{\partial q}^2}\right)$$

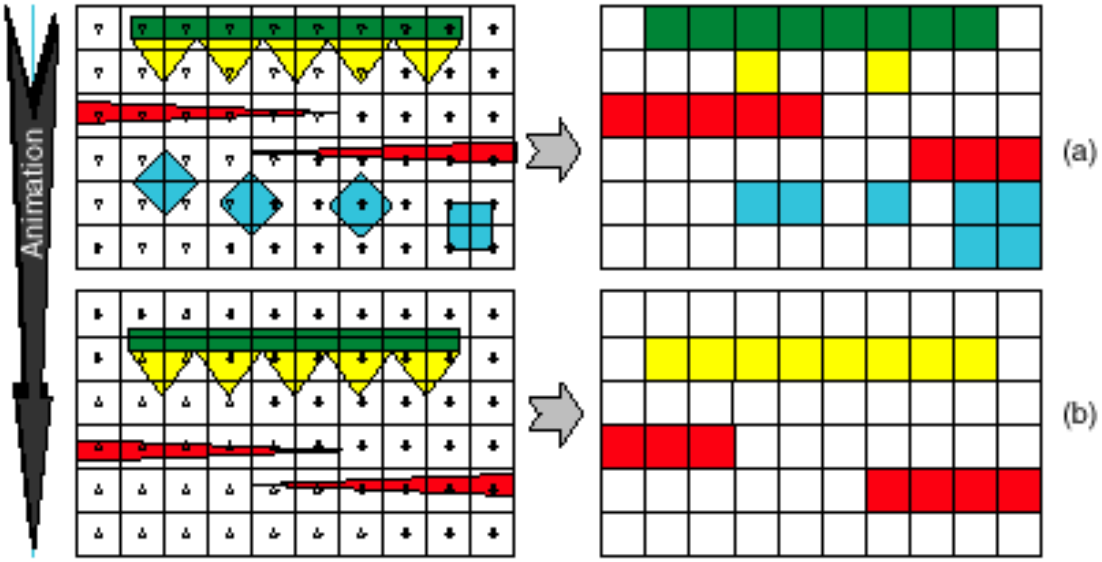
Получившиеся значения LOD и координат (s, t) вещественны. Существует три метода выбора цвета пиксела, основанные на данных оценках LOD:
1. Nearest (см. рис.): округляем LOD до ближайшего уровня, а (s, t) до ближайшего тексела
2.Linear: уровни детализации [LOD] и [LOD+1], на каждом берем цвет точки (s, t) округляя до ближайшего тексела, цвет пиксела определяем линейной интерполяцией цветов [LOD] и [LOD+1] по вещественному значению уровня LOD.
3.Trilinear: аналогичен Linear, но значения цвета на обоих уровнях, рассчитываются по методу Bilinear.

Метод Trilinear позволяет полностью исключить артефакты межуровнего перехода при MIP-mapping-е. Необходимый объем вычислений пока не позволяет реализовать его в реальном времени на слабых ускорителях:

Артефакты. Влияние анимации

Иллюстрация исчезновения полигонов

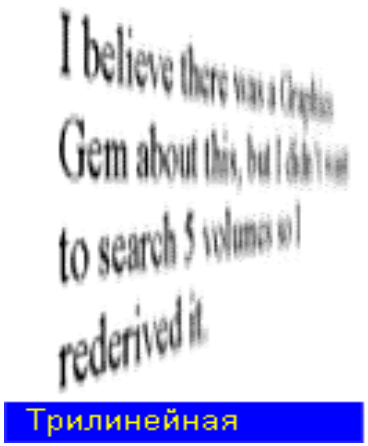
- (а) хорошо детализированное изображение слева,
- (б) грубая версия с отсутствующими полигонами и деталями справа



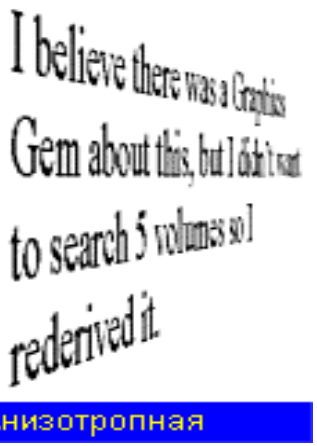
Анизотропная фильтрация (Anisotropic filtering)

Так как билинейная и трилинейная фильтрации являются изотропными - изображение фильтруется в определенной форме - в форме квадрата. Большинство же формируемых объектов из-за проекционных искажений не подходят под эту форму: для их качественной обработки необходимо использовать другой тип фильтрации - **анизотропный**. Анизотропная фильтрация обычно оперирует не менее чем 8 текселями, во все стороны mip-map уровней, при этом используется модель неопределенной заранее формы. В результате убираются шумы и искажения объектов, а изображение в целом получается более качественным. Анизотропная фильтрация работает с текселами как с эллипсами и для получения одного пиксела обрабатывает до 32 (2x16) текселов.

Качество текстуры при анизотропной ➔ фильтрации даже на дальних дистанциях остается схожей с оригинальным; при изотропной фильтрации же видна тенденция в "сглаживанию" изображения, в результате теряется качество. Анизотропная фильтрация, как и трилинейная, уменьшает неровность текстур. Но при использовании анизотропной фильтрации качество получается лучшим.



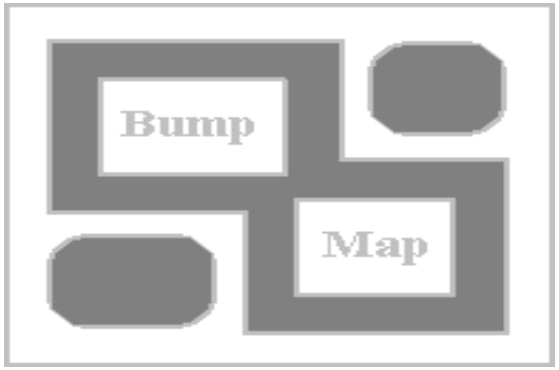
Трилинейная

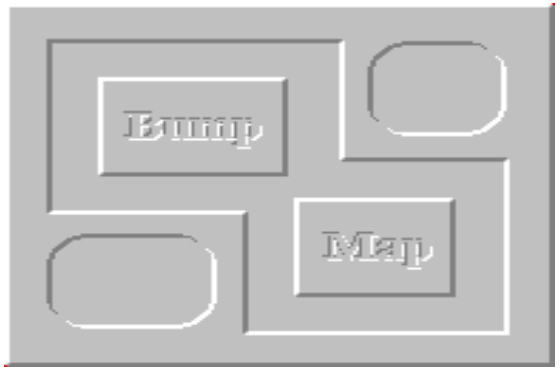


Анизотропная

Наложение рельефа (bump-mapping)

Для того чтобы подчеркнуть бугорки и впадины рельефа с помощью светотени, надо затемнить либо осветлить стенки этих бугорков и впадин. Другой метод состоит в симуляции рельефности, глянцевой или зеркальной поверхности, отражением окружающей среды. Это и делает техника наложения рельефа. Рельефное текстурирование напоминает обычный процесс наложения текстуры на полигон, но текстуры (на рис. слева), предназначенной для придания плоскому полигону зрительного ощущения рельефа и объемности. Эта техника может добавить детализацию сцене без создания дополнительных полигонов. Сам полигон остается плоским.

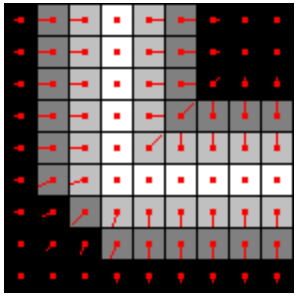




Для реализации рельефного текстурирования в настоящий момент существует два подхода. Первый -- реализация эффекта **выдавливания** (**embossed effect**) в процессе bump mapping, и второй – использование карт окружающей среды (**environment-map bump mapping**).

Метод выдавливания (embossing)

Метод выдавливания (embossing) использует карту высот для затенения и осветления определенных участков поверхности так, чтобы создавалось впечатление, что выпуклости отбрасывают тени. Метод реализован в современных 3D акселераторах.



Красные векторы указывают на уменьшение высоты.

Способ преобразования информации о высоте неровностей на карте высот в информацию о величине подстройки вектора нормали

1.Для каждого пиксела по высотам карты (bump map) рассчитывается вектор, указывающий направление уклона поверхности, т.е. вектор градиента:

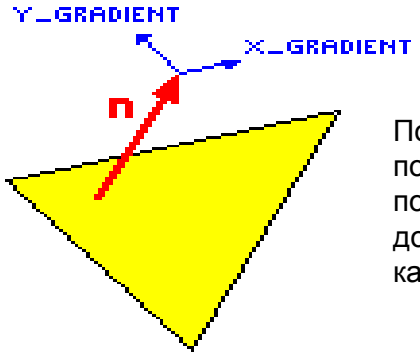
x_gradient = pixel(x-1, y) - pixel(x+1, y), y_gradient = pixel(x, y-1) - pixel(x, y+1),

где x и y -- координаты соответствующего пиксела

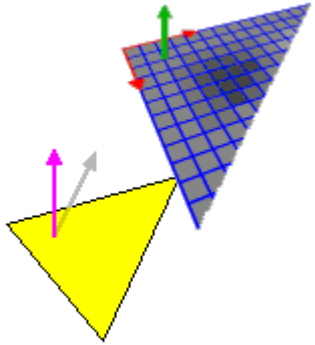
2.Вектор нормали к поверхности корректируется для каждого пиксела на величину градиента

New_Normal = Normal + (U * x_gradient) + (V * y_gradient)

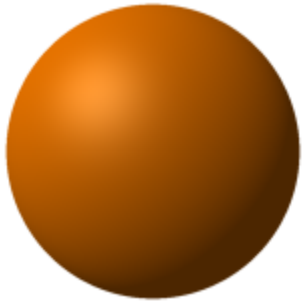
Получив новый вектор нормали, мы можете просчитать яркость данного пикселя.



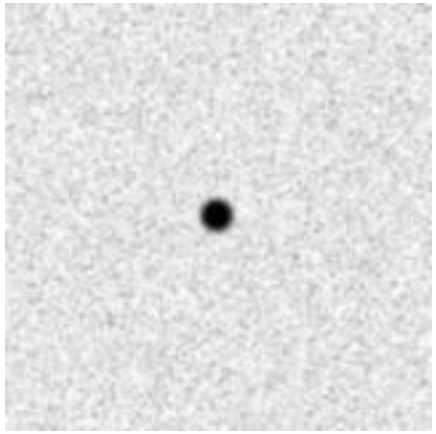
Полигон с изначальным вектором нормали, обозначенным **n**. Также показаны два вектора, которые будут использованы для изменения положения (направления) нормали к пикселю под ним. Оба вектора должны располагаться параллельно осям координат применяемой карты высот.



Bump-mapping. Example



A sphere without bump mapping



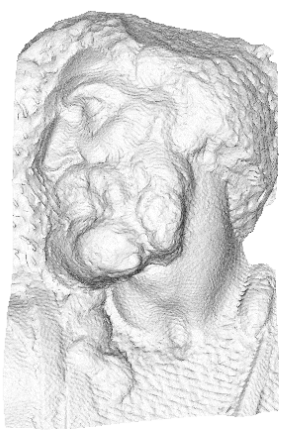
The bump map that is applied to the image

This sphere is geometrically the same as the first, but has a bump map applied. This changes giving it the appearance of a bumpy texture like an orange.

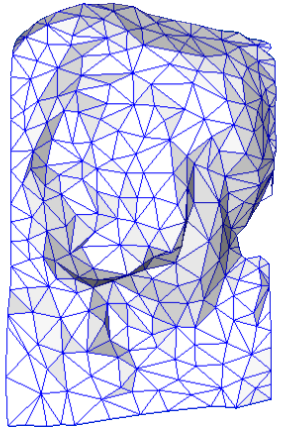
Normal mapping

In 3D computer graphics, normal mapping is an application of the technique known as bump mapping. Normal mapping is sometimes referred to as "Dot3 bump mapping".

While bump mapping perturbs the existing normal of a model, **normal mapping replaces the normal entirely**. Like bump mapping, it is used to add details to shading without using more polygons. But where a bump map is usually calculated based on a single-channel (interpreted as grayscale) image, the source for the normals in normal mapping is usually a multichannel image (x, y and z channels) derived from a set of more detailed versions of the objects. The values of each channel usually represent the xyz coordinates of the normal in the point corresponding to that texel.

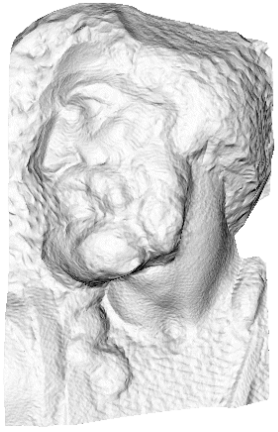


original mesh
4M triangles

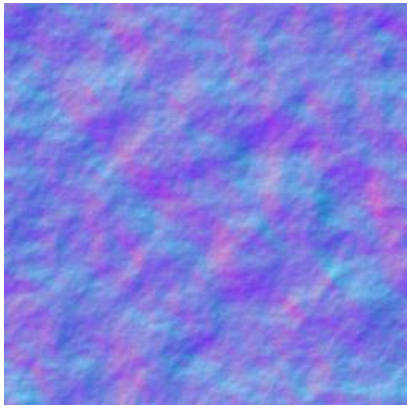


simplified mesh
500 triangles

Normal mapping used to re-detail simplified meshes



simplified mesh
and normal mapping
500 triangles

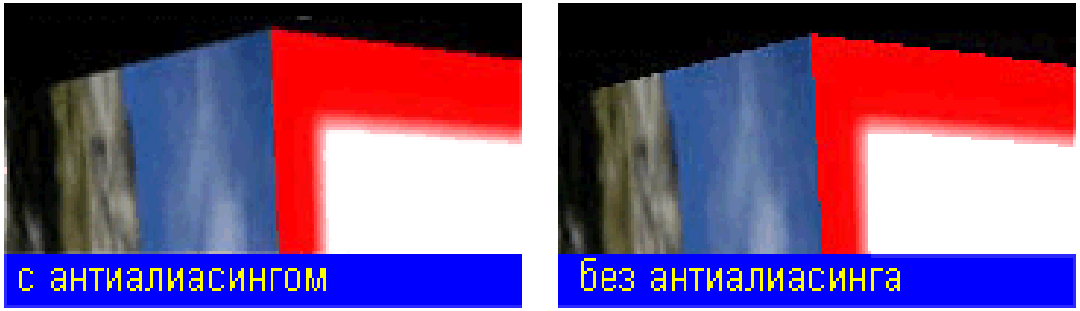


Example of a multichannel normal map

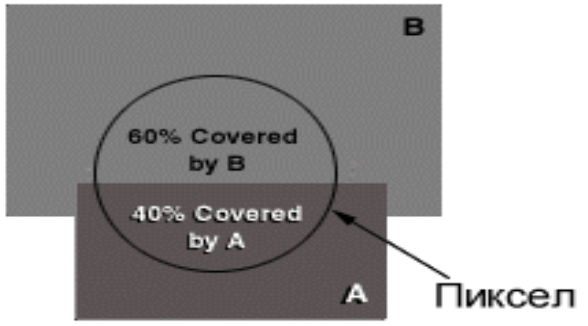
Сглаживание (anti-aliasing).
Краевой антиалиасинг

Алиасинг – результат сэмплинга, то есть преобразования непрерывного изображения в дискретное. Алиасинг ухудшает качество изображения, вызывая разнообразные артефакты: такие, как лестничный эффект. Процедура *сглаживания* может помочь улучшить (или, как минимум, не ухудшить) качество графического изображения. В некоторых программах автоматическое применение сглаживания даже является одной из опций функции масштабирования. По предназначению антиалиасинг делится на краевой и полный.

Краевой антиалиасинг (КАА) – механизм борьбы с лестничным эффектом. КАА сглаживает края полигонов и диагональные линии. Для реализации краевого антиалиасинга чаще всего используют технику усреднения по площади (area averaging).

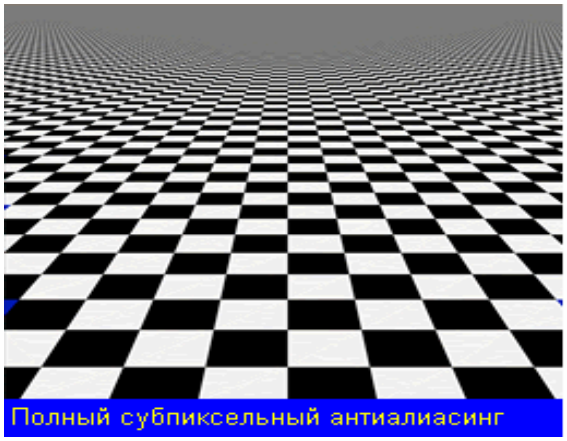


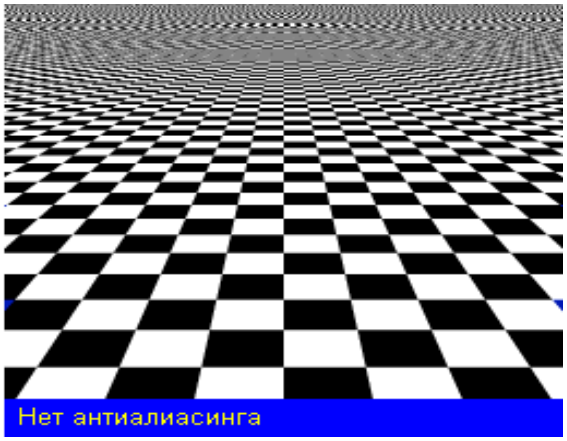
Цвет пиксела определяется на основании того, насколько каждый полигон перекрывает данный пиксел (см. рис). ➔

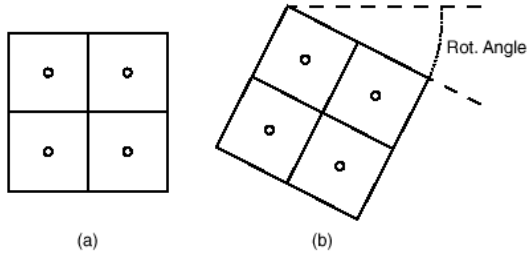


Anti-aliasing. Полный антиалиасинг

Полный антиалиасинг, в отличие от краевого, направлен на полную нейтрализацию алиасинга, в том числе bleeding-а (просачивания цветов фона). Представителем полного антиалиасинга является **субпиксельный антиалиасинг**, который реализован в 3D-ускорителях.
Субпиксельный антиалиасинг базируется на технике **суперсэмплинга**. Суперсэмплинг означает, что вся сцена рендерится в каком-то большом виртуальном разрешении, а затем сжимается до фактического разрешения. В общем случае виртуальное и фактическое разрешения могут быть некратными. Техника суперсэмплинга эффективно реализуется благодаря тому, что ускорители используют tile-based архитектуру. Ускорителю традиционной архитектуры потребовался бы большой объем памяти (для виртуального разрешения 1600x1200 – более 8 MB). Ускоритель tile-based архитектуры работает не с целым фреймбуфером, а с отдельными фрагментами (tile - плитка или фрагмент изображения, тайл может быть и размером 32 на 32 пикселя). И все данные о субпикселах он хранит только для фрагмента, который рендерится в данный момент.



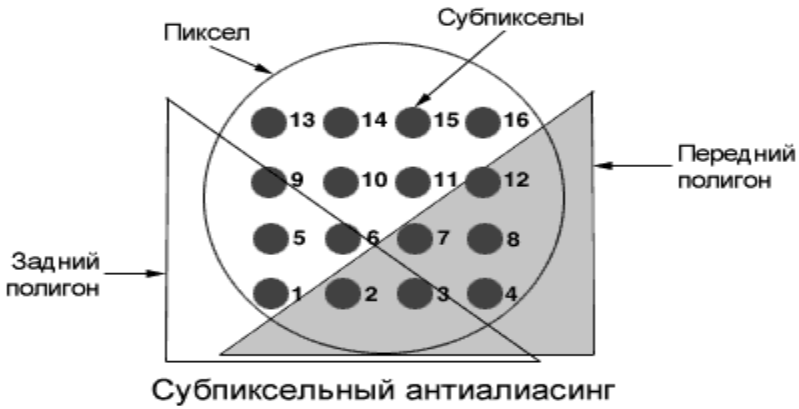




a)Упорядоченная решетка (Ordered Grid Super-sampling, OGSS);
b)Повернутая или флуктуирующая (Rotated Grid Super-sampling, RGSS)

Полный антиалиасинг.
Мультисэмплинг и маски

В некоторых 3D-ускорителях используется другой метод, основанный на хранении **маски**. Рассмотрим случай, когда 1 пиксел разбивается на 16 (4x4) субпикселов, а полигоны рендерятся front-to-back (эта техника называется **мультисэмплингом**).



Субпиксельный антиалиасинг

Когда рендерится полигон не переднем плане, субпикселы 2,3,4,7,8,12 окрашиваются в цвет переднего полигона и запоминаются как маска. Когда рендерится задний полигон, в его цвет окрашиваются субпикселы 1,5,6,9. Субпикселы 2,3, маскируются и остаются с цветом переднего полигона. В результате – никакого **bleeding**-а.

Недостаток такого антиалиасинга – это необходимость хранения маски для каждого пиксела и требование сортировки полигонов front-to-back. Второе требование можно обойти, сохраняя z-координату для каждого субпиксела. Хранить z-координаты для всех субпикселей на экране невозможно, так как это требует гигантского объема видеопамяти. Поддержку субпиксельного антиалиасинга с z-буферизацией реализует техника **аккумулятора**. В памяти аккумулятора последовательно проходят обработку субпиксельные фрагменты. Как следствие, - уменьшение производительности в число раз, равное числу субпикселов в пикселе. Например субпиксельный антиалиасинг 4x4 снижает производительность в 16 раз.

Фотореалистическая визуализация.
Отображение отражения окружающей среды

Развитие методов

1) **Хромированное отображение** (chrome mapping) – случайные блики.
2) **Отображение среды** (environment mapping):

Метод окружающей сферы

Авторы – Блинн (Blinn,1976) и Ньюэлл

Для каждого направления луча $\boldsymbol{u}$, соответствующего вертекса $\boldsymbol{P}$ и нормали $\boldsymbol{n}$ поверхности строится направление отраженного поверхностью луча $\boldsymbol{r}$

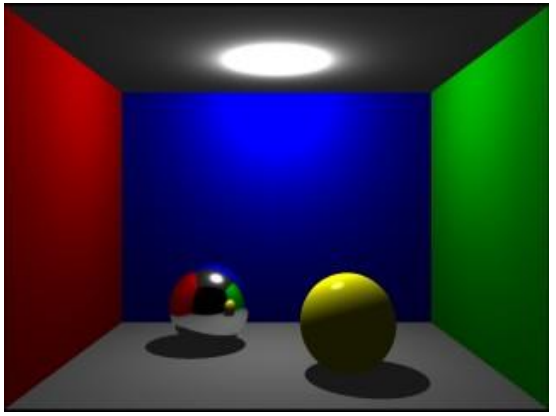
$$\boldsymbol{r}=\boldsymbol{u}-2(\boldsymbol{u},\boldsymbol{n})\boldsymbol{n}\ , \qquad \{\boldsymbol{s},\boldsymbol{t}\}=\boldsymbol{F}(\boldsymbol{r})$$

Пересечение направления луча с окружающей сферой, на которую "натянута" текстура, дает конкретный тексел текстуры, присваиваемый вершине.

Используется текстурное отображение бликов, попавших между вершинами.

Метод окружающего куба ➔

Автор – Грин (Green, 1986)



Отображение отражения окружающей среды.
Метод окружающего куба

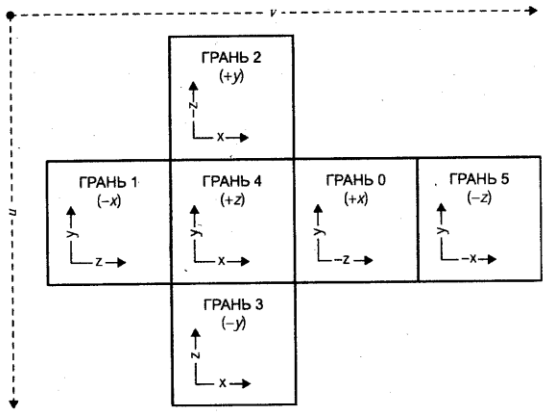


Рис. Индексы и системы координат граней кубической карты

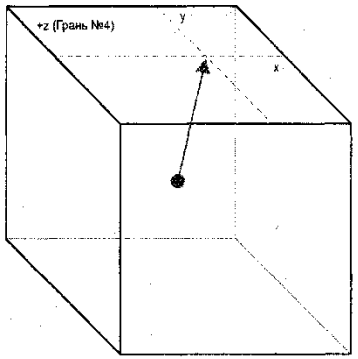


Рис.Отображение углов наблюдения на единичной кубической карте в 3D-координаты текстуры

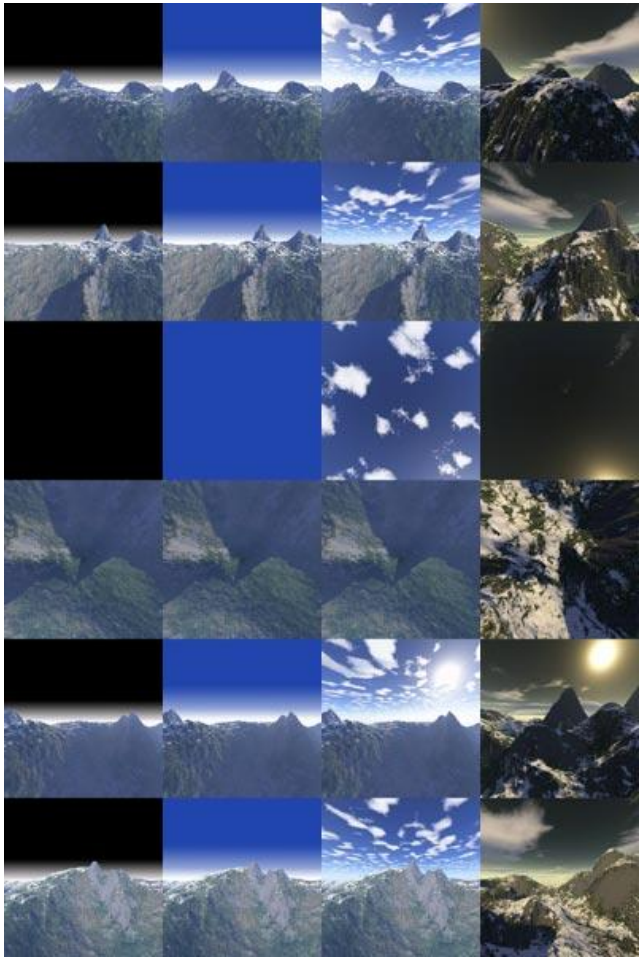


Рис. Примеры кубических карт из …\media\textures (из кн. Снука)

Инициализация текстур в OpenGL

```
photo_image = auxDIBImageLoad("photo.bmp");
space_image = auxDIBImageLoad("space.bmp");
glGenTextures(1, &photo_tex);
glGenTextures(1, &space_tex);

glBindTexture(GL_TEXTURE_2D, photo_tex);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
glTexImage2D(GL_TEXTURE_2D, 0, 3, photo_image->sizeX,  photo_image->sizeY,
        0, GL_RGB, GL_UNSIGNED_BYTE, photo_image->data);
```

Текстурирование в OpenGL. Нижний уровень

```
glEnable(GL_TEXTURE_2D);
  glColor3d(1,1,1);

    glBindTexture(GL_TEXTURE_2D, space_tex );  // дальний фон - капли на синем
       glBegin(GL_QUADS);
        glTexCoord2d(0,0); glVertex3d(-5,-5,-4.6);
        glTexCoord2d(0,1); glVertex3d(-5, 5,-4.6);
        glTexCoord2d(1,1); glVertex3d( 5, 5,-4.6);
        glTexCoord2d(1,0); glVertex3d( 5,-5,-4.6);
       glEnd();

    glBindTexture(GL_TEXTURE_2D, photo_tex);    // ближний фон - заяц тайлинг 4x4
       glBegin(GL_QUADS);
        glTexCoord2d(0,0); glVertex3d(-4.5,-4.5,-4.5);
        glTexCoord2d(0,4); glVertex3d(-4.5, 4.5,-4.5);
        glTexCoord2d(4,4); glVertex3d( 4.5, 4.5,-4.5);
        glTexCoord2d(4,0); glVertex3d( 4.5,-4.5,-4.5);
       glEnd();

  glDisable(GL_TEXTURE_2D);
```

Текстурирование в OpenGL. Генерация текстуры

```
glPushMatrix();          // сохраняем текущую систему координат
  //glTranslated(0,0,0); glRotated(time*10,0.2,1,0.2);
  glRotated(alpha, 0,1,0);   // вращение подсцены мышью
  glRotated(beta, -1,0,0);

   glEnable(GL_TEXTURE_2D);
   glEnable(GL_TEXTURE_GEN_S); //разрешить генерацию текстуры по S
   glEnable(GL_TEXTURE_GEN_T);

   glBindTexture(GL_TEXTURE_2D, photo_tex ); //привязка конкретной текстуры
   glColor3d(1,1,0); //фоновый цвет объекта
   glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP); //методом окружающей сферы
   glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
   //glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
   //в координатах объекта
   auxSolidTeapot(1.2);        // для чайника

   glDisable(GL_TEXTURE_GEN_S);
   glDisable(GL_TEXTURE_GEN_T);
   glDisable(GL_TEXTURE_2D);
glPopMatrix();          // возвращаем систему координат
```

Источники

1. **Курсы лаборатории компьютерной графики ВМК МГУ**
http://courses.graphicon.ru/ (открытые),
http://courses.graphics.cs.msu.ru/(закрытые)