Компютърна графика

Визуализиране на текстури

доц. Милена Лазарова, кат. КС, ФКСУ

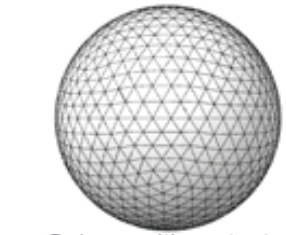
Текстуриране

- Texture Mapping → създаване на “красиви” повърхности на обектите
  - “облепване” на повърхност с изображение
  - създават се детайли без да се добавят повече полигони
    - примерни изображения за генериране на текстура: тухли, дървесина, небе с облаци

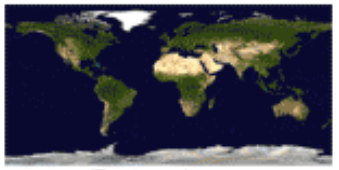
Microsoft Flight Simulator X

Текстуриране

- Текстуриране
  - текстурното изображение може да се разглежда като направено от разтегателна материя
  - поставя се върху повърхността за да ***замести*** или ***промени*** оригиналния цвят
    - оригиналният интензитет може да се използва за да се ***модифицира текстурата***
  - хардуерно имплементирано в GPU



Sphere with no texture



Texture image

Sphere with texture

Текстуриране

- Мотиви за използване на текстури
  - да се увеличи количеството детайли в изображението
  - изчислително “скъпо” решение
    - добавяне на детайли към модела
  - ефективно решение
    - използване на текстура върху модела

Добавяне на детайли към модел

- Детайли се добавят като части от геометрията на обекта
  - повече и по-малки триъгълници
- ***Предимства***
  - реалистично представяне на осветяване
- ***Недостатъци***
  - трудно се генерира
    - моделиращите програми не винаги успяват да добавят желаното ниво на детайли
  - изчислително “скъпо” решение
    - модели с много детайли изискват много време за визуализиране
    - модели с много детайли изискват много памет за съхраняване
  - сложните детайли не могат да се пре-използват

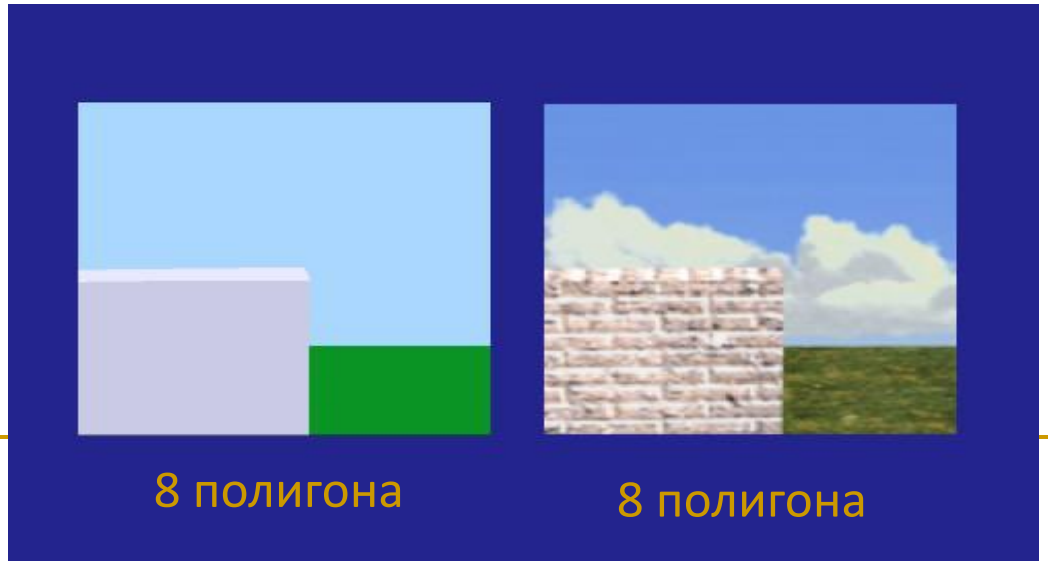
Наслагване на текстура

- ***Предимства***
  - текстурата може да се пре-използва
    - съхранява се веднъж, използва се многократно
    - може да се компресира за намаляване на използваната памет
  - бързо се създава и се визуализира текстуриране
  - използва се интуитивно
  - не променя геометрията на обектите
  - особено полезна за отдалечени обекти, терен, небе

- ***Недостатъци***
  - много груба апроксимация на реалността
    - само наложена, но непроменена текстура може да направи повърхността нереалистична

Текстуриране

- Детайли, които текстурирането добавя към обектите
  - дифузни, фонови и отразени цветове
  - огледални компоненти
  - прозрачност, рефлективност
  - фини детайли към повърхността (грапавини)
  - проектирано осветяване и сенки



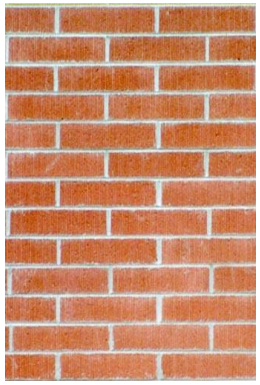
Текстуриране

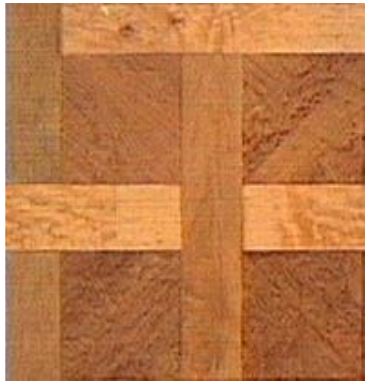
- За интерактивни приложения в реално време не могат да се използват обекти със сложна геометрична форма на моделите
  - текстурирането осигурява “достатъчно добро” решение
- Пример: сцена съставена само от текстурно изобразени полигони
  - ***ниска сложност на сцената***

Gears of War

Текстуриране

- ***Видове текстури***
  - случайни
  - регулярни





  - ***Всяко изображение може да се използва като текстура***

Текстуриране

- Източници на текстура
  - ***процедурни***
    - текстурата се задава с изчисляване на 2D функция
  - ***изображение***
    - текстурата се задава с bitmap изображение
      - изображението може да е в стандартен графичен файлов формат: jpg, bmp, gif, pbm, …
      - трябва да се определят височина и ширина на текстурата (брой редове/колони на изображението)
      - трябва да се зададе указател към данните за пикселите на текстурата

Изобразяване

- Изобразяване (Mapping)
  - функцията е изобразяване
    - функциите изобразяват стойности от подобласт на една или няколко входни променливи в подобласт от стойности на зависима изходна променлива
    - едно пространство може да се преобразува (изобрази) в друго пространство чрез функция

Изобразяване

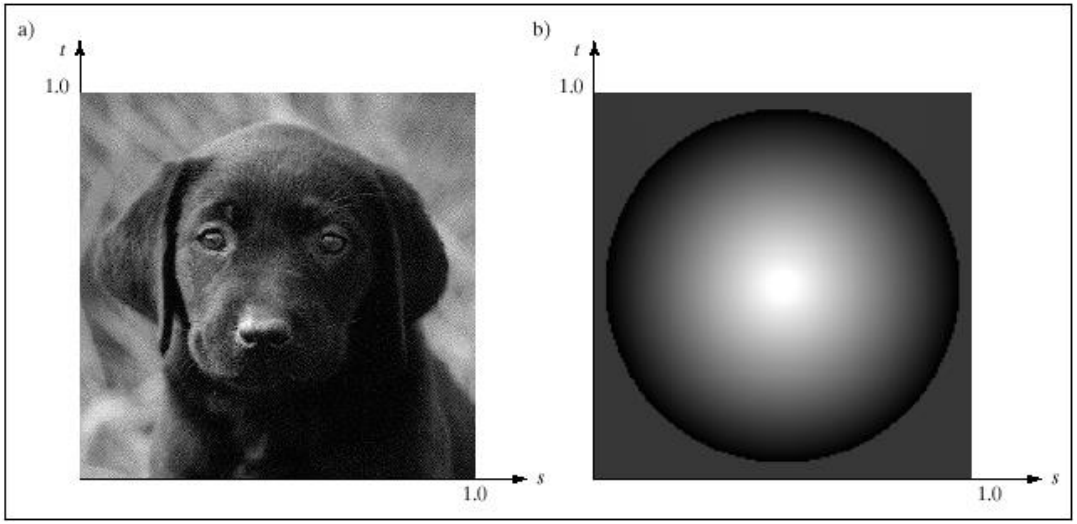
- Изобразяване чрез определяне на пресечни точки
  - чрез линейни трансформации точки и вектори се преместват в пространството
    - ***изобразяване на точки от екрана за визуализиране*** в точки от нормализирано пространство на камерата на наблюдение
    - ***изобразяване на лъчи от нормализираното пространство на камерата*** на наблюдение в лъчи от ненормализираното пространство на световния изглед
    - ***изобразяване на ненормализирани лъчи от световния изглед*** в нетрансформирани точки в пространството на изображението и изчисляване на точки на пресичане в пространството на изображението
    - ***изобразяване на точки и нормали от пространството на обекта*** в световни координати за определяне на осветеност

Текстуриране

- Изобразяване на текстура
  - точки от ***повърхността на обекта***

    се изобразяват в
  - точки от ***текстурата***

  - ***Каква е функцията за изобразяването?***

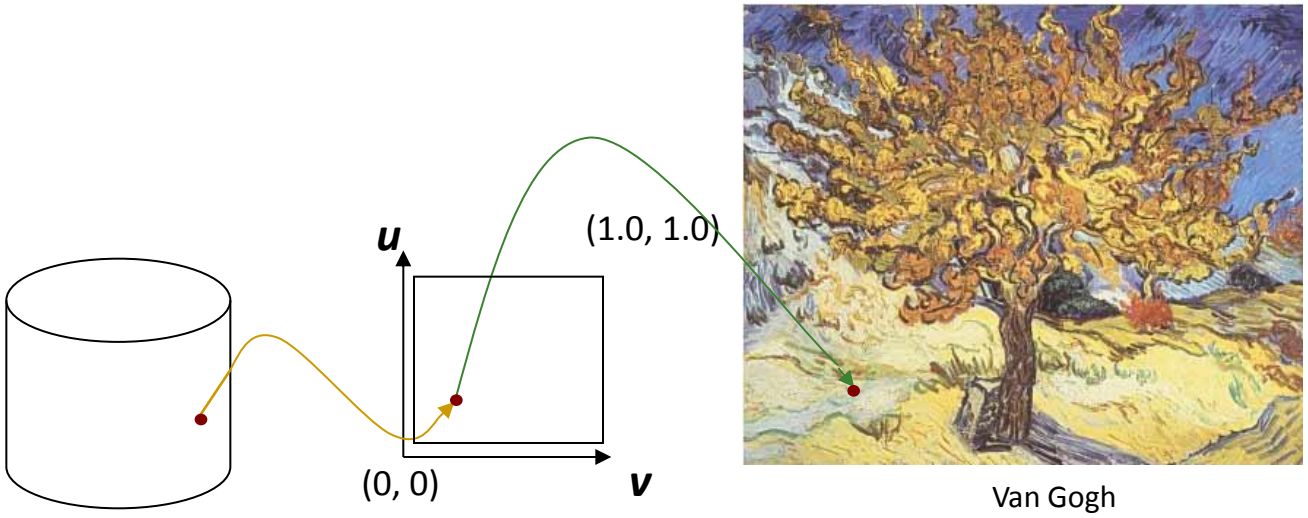


Текстуриране

- ***Основна идея***
  - ***Дефиниция***
    - текстуриране (texture mapping) е процеса на изобразяване на геометрична точка в пространството в стойност (цвят, нормала, ...) от текстурно изображение
  - ***Цел***
    - изобразяване на произволна геометрична форма в текстура с произволна размерност
  - ***Две стъпки***
    - изобразяване на точка от произволна геометрична форма в точка от абстрактен единичен квадрат представящ конкретно изображение
    - изобразяване на точка от абстрактен единичен квадрат в точка от конкретно изображение с произволни размери

Текстуриране

- ***Две стъпки на текстурно изобразяване***



Van Gogh

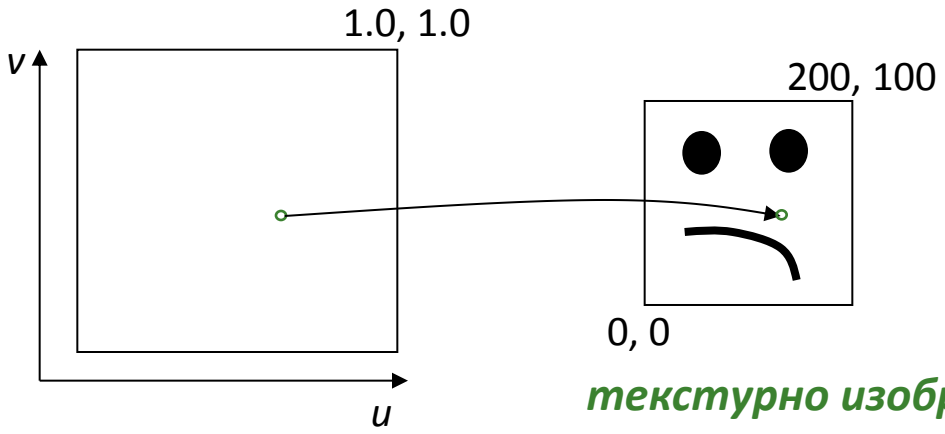
  - първата стъпка е по-сложна
  - втората стъпка е по-лесна

пространството *u,v* не е свързано с UV пространството на проекционната равнина

Текстуриране

- ***Стъпка 2. Изобразяване от единичен квадрат в текстурно изображение с произволни размери***

  1. Трансформиране на точка от абстрактна непрекъсната текстурна равнина в точка от дискретно текстурно изображение
  2. Определяне на цвета на трансформираната точка в текстурното изображение



единична равнина на текстурата

текстурно изображение
(произволна pixmap)

Пример:
(0.0, 0.0) $\Rightarrow$ (0, 0)
(1.0, 1.0) $\Rightarrow$ (200, 100)
(0.7, 0.45) $\Rightarrow$ (140, 45)

Текстуриране

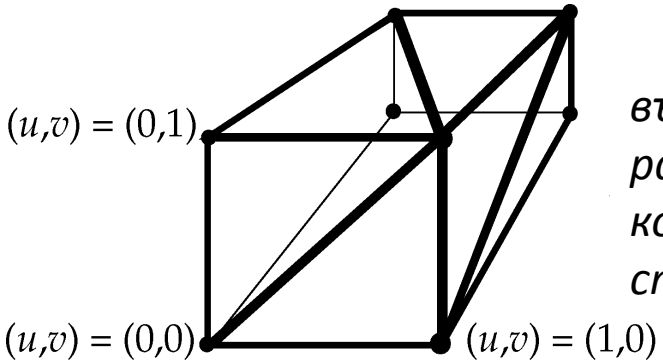
- ***Стъпка 2. Изобразяване от единичен квадрат в текстурно изображение с произволни размери***

  1. За всяка точка (*u*, *v*) от единичния квадрат съответната точка в текстура с дължина *l* пиксела и височина *h* пиксела е (*u*∗*l*, *v*∗*h*)
     - не винаги получените координати са дискретни точки в текстурата
       - получават се грешки от дискретизацията (sampling errors)
       - може да се наложи осредняване на съседните текстурни пиксели (филтрация)
  2. Определяне на цвета на трансформираната точка в текстурното изображение

- Единичният квадрат *uv* се разглежда като разтягащ се лист (rubber sheet), който обгръща обекта (wrap)

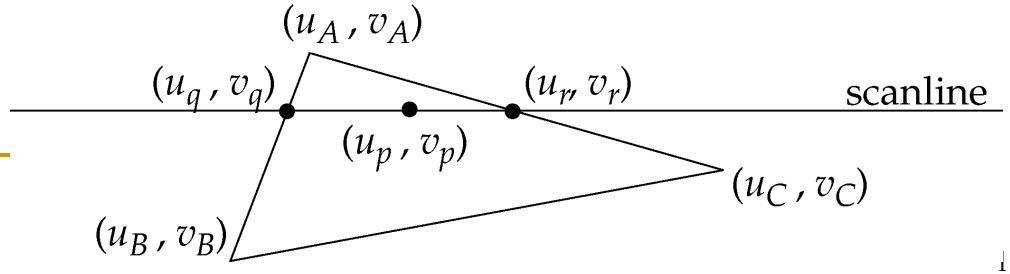
Текстуриране на полигони

- ***Стъпка 1. Интерполиране на координатите на текстурата***
  - (*u*,*v*) координатите предварително се изчисляват и се определят за всеки от възлите на триъгълниците по време на триангулиране (tessellation)
  - съхраняват се заедно с координатите на възлите в модела

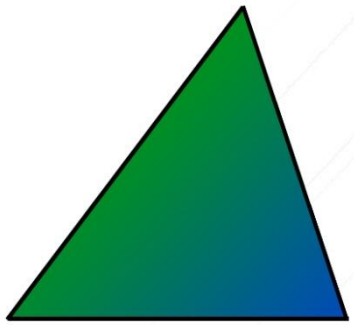


възлите имат различни текстурни координати за всяка страна

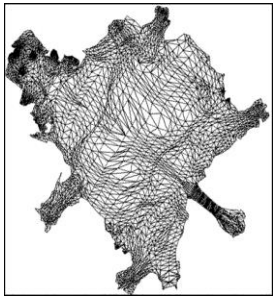
- Линейна интерполация на u,v координатите за всеки триъгълник/регион при определяне на осветеност по модел на Гуро

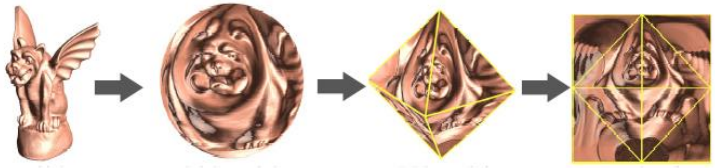


Текстуриране на мрежи



- ***Текстуриране на триъгълник***
  - задават се текстурни координати за всеки възел в зависимост от позицията му в единичния квадрат
  - Текстурните координати се съхраняват заедно с координатите на възела, нормалата и др. при визуализиране
  - Интерполират се стойности в триъгълника на базата на барицентрични координати (***barycentric coordinates***)
- ***Текстуриране на мрежа***
  - няма стандартен метод
  - изравняване със запазване на ъгли (A. Sheffer, E. de Sturler)
    - запазва ъглите
    - много излишно пространство
  - сферична параметризация (E. Praun, H. Hoppe)
    - няма излишно пространство
    - може да доведе до изкривяване
    - трудно се имплементира





Барицентрични координати

- При интерполиране на стойности по протежение на линия
  - по зададени стойностите на два цвята C_a и C_b може да се определи стойността на произволна точка по линията, определена от двете стойности

$$C(t) = (1-t)C_a + tC_b, \quad 0 \le t \le 1$$

- Стойността на цвета *C(t)* за точка с местоположение (1– *t)a*+t*b* може да бъде определена като

$$C(s,t) = sC_a + tC_b, \quad s+t=1, \quad s,t \ge 0$$

- Стойностите на *s* и *t* са ***барицентрични координати*** за линейния сегмент в цветовото пространство между C_a и C_b

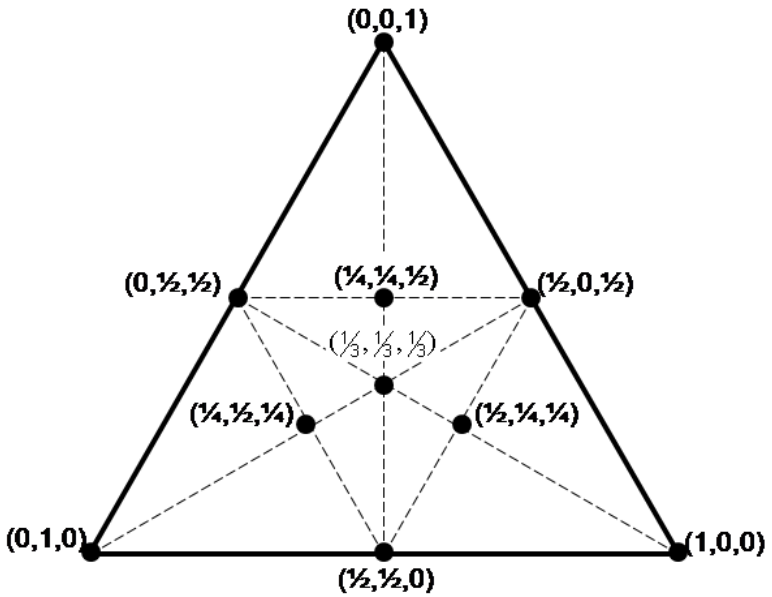
Барицентрични координати

- Барицентричните координати могат да бъдат обобщени за триъгълник

$$C(s,t,u) = sC_a + tC_b + uC_c, \quad s+t+u=1, \quad s,t,u \geq 0$$

  - Аналогично е обобщението на барицентрични координати за *n*-мерна фигура

- ***Приложение на барицентрични координати***
  - при пресичане на лъч с многостенен обект се определят
    - възлите на пресечения обект
    - Барицентрични координати (t_1,t_2,t_3) на пресечната точка
  - използват за интерполиране на цвят, нормала, текстурни координати и др. между възлите на триъгълника



Барицентрични координати

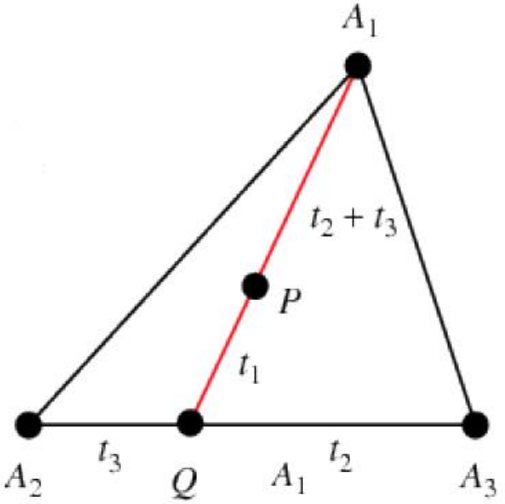
- За определяне на тегла на всеки възел
  - изчислява се пресечната точка Q на линията през A_1 и P и линията през A_2 и A_3

    $t_3' = |Q - A_2|$

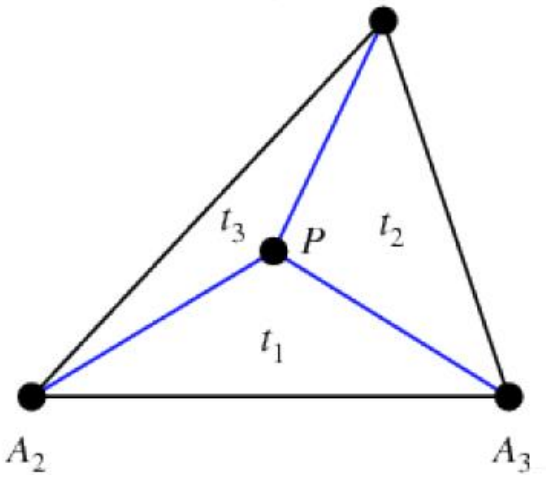
    $t_2' = |Q - A_3|$

    $t_1' = |P - Q|$

    $(t_1, t_2, t_3) = (t_1', t_2', t_3')/(t_1+t_2+t_3)$

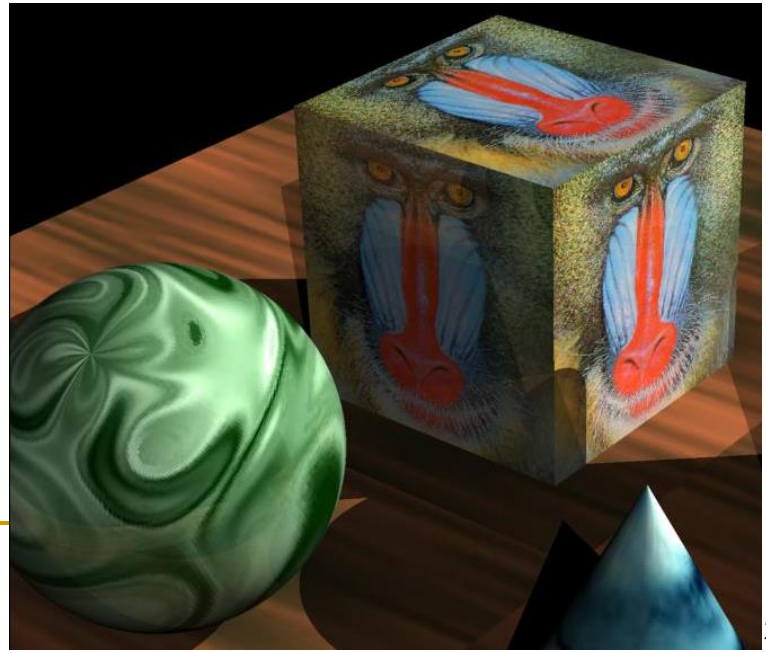


  - теглата във всеки възел са пропорционални на лица на триъгълниците
    - например теглото в A_1 е пропорционално на лицето на триъгълника P, A_2, A_3



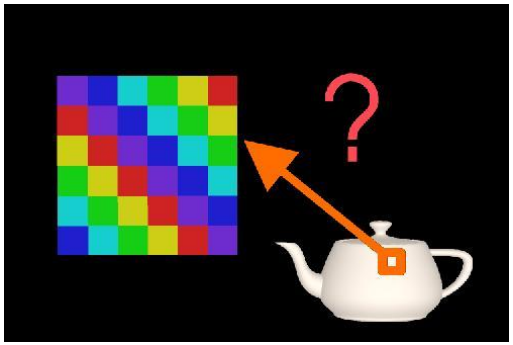
Текстуриране и Ray Tracing

- С трасиране на лъчи се определя пресечна точка на лъч и обект *(x, y, z)*
- ***Цел***
  - да се преобразува пресечната точка *(x, y, z)* в точка от единичния квадрат (*u, v*) за да се определи цвета на пиксела от екрана
  - изобразяване *(x, y, z)* в *(u, v)*
- ***Три основни случая***
  - равнини
  - цилиндри
  - сфери



Текстуриране и Ray Tracing

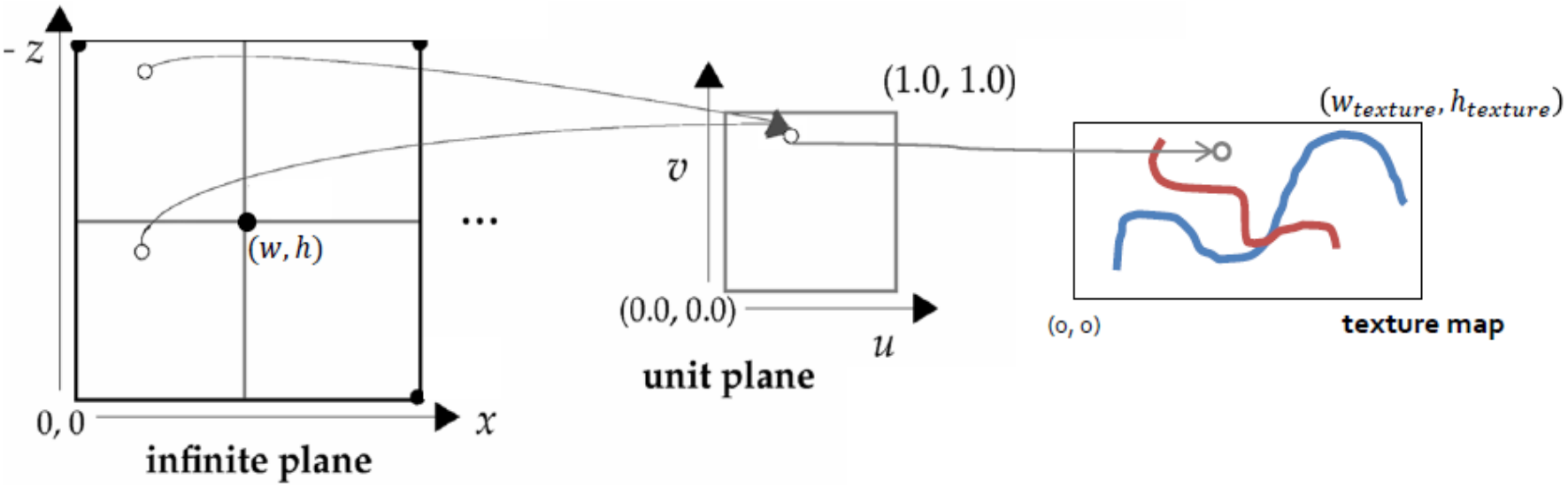
- Изобразяването се изчислява най-лесно за прости, ***нетрансформирани*** обекти
  - т.е. преобразуваме координати в пространството на обекта *(x, y, z)* в *(u, v)*



- ***Недостатък***
  - може да се получи нежелано мащабиране на текстурата при трансформиране на обекта в световни координати
    - например мащабиране на текстурирана сфера с коефициент 2 по y ще мащабира текстурното изображение с коефициент 2 по *y*
  - ***Филтриране*** за да се избегне подобен ефект

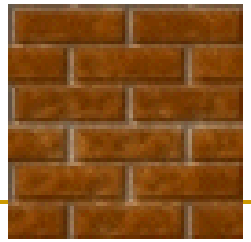
Текстуриране на правоъгълници

- Изобразяване на точка от ***много голям правоъгълник*** в точка от единичния квадрат
  - ***Tiling***: повторение на текстурата в безкрайна равнина
    - при зададени координати (*x*,*y*) на точка от произволно голям правоъгълник, който се покрива с повторения на текстура с размер (*w*,*h*) се определят координатите (*u*,*v*) в единичния квадрат, представящ текстура с произволни размери

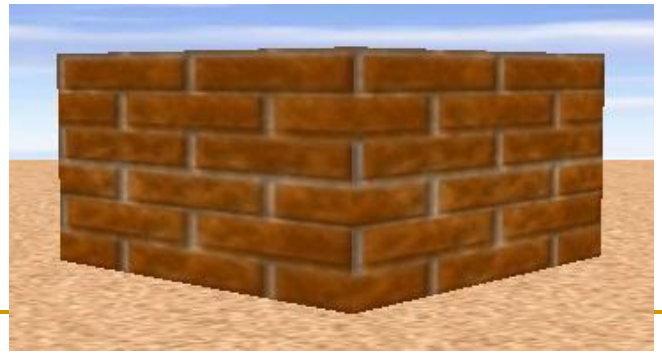


Текстуриране на правоъгълници

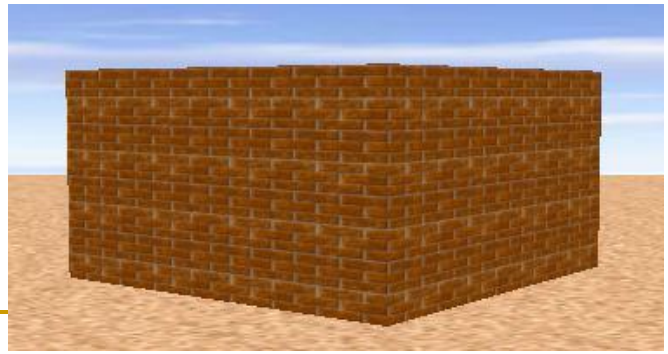
- ***Tiling***
  - създава се текстуриран обект с прилагане на текстура с тухли за равнина
    - получава се реалистично изображение, но с твърде малко тухли в стената
  - с повторение на текстурата се увеличава броя тухли в стената и се постига по-добра реалистичност на текстурирането



текстура



без tiling



с tiling

Текстуриране на правоъгълници

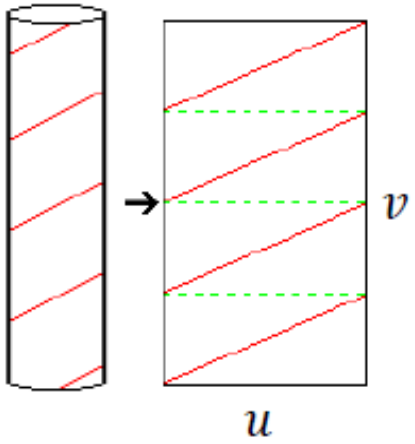
- ***Stretching***
  - с нерегулярни текстури не може да се използва подхода с повторение за текстуриране на голям обект
  - използва се разтягане
- Пример: създаване на небе

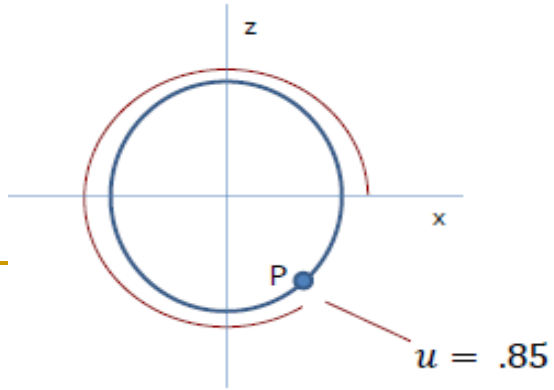
текстура

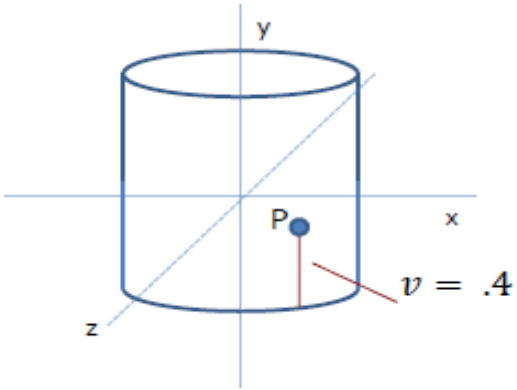
текстуриране с разтегляне

Текстуриране на цилиндри/конуси

- Изобразяване на точка от ***цилиндър или конус*** в точка от единичния квадрат
  - дадена е точка P от повърхността
    - ако лежи в една от страните, то точката се изобразява в равнината на тази страна
    - ако лежи върху крива повърхнина
      - използва се позицията на точка по периметъра за да се определи *u*
      - използва се височината на точката за да се определи *v*

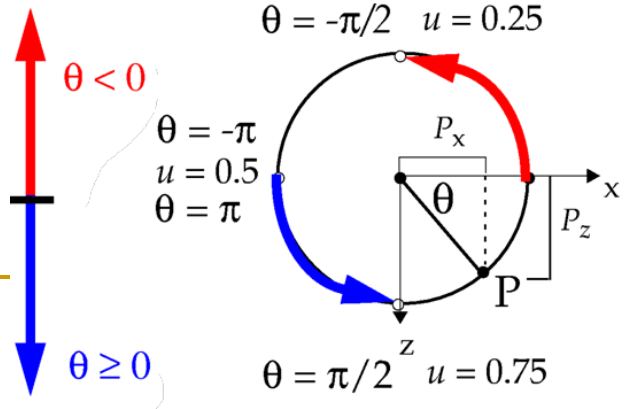






Текстуриране на цилиндри/конуси

- Изчисляване на координатата *u* за конуси и цилиндри
  - всички точки от периметъра на обекта да се изобразят в [0, 1]
    - най-лесно: $u = \theta/(2\pi)$
      - стойността на θ не винаги се определя лесно
    - ако се използва arctg не се изобразява цялата окръжност и се получава неоднозначност – две точки от периметъра се изобразяват в една и съща координата
      - например arctg(1, 1) = arctg(-1, -1) =π/2
    - ако се използва atan2(x, y) се осигурява обхват на стойностите от −π до π
      - но има точка на прекъсване



$$\theta = \text{atan2}\,(P_z, P_x)$$

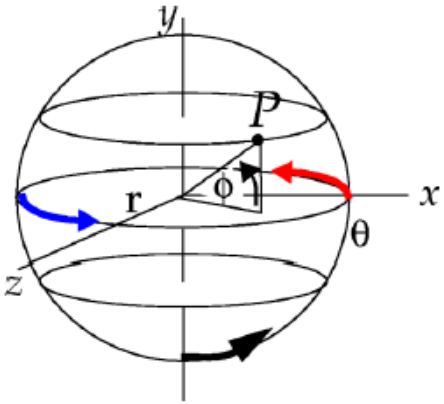
$$\text{if } \theta < 0 \quad u = \frac{-\theta}{2\pi} \qquad [0.0, 0.5]$$

$$\text{if } \theta \geq 0 \quad u = 1 - \frac{\theta}{2\pi} \qquad [0.5, 1.0]$$

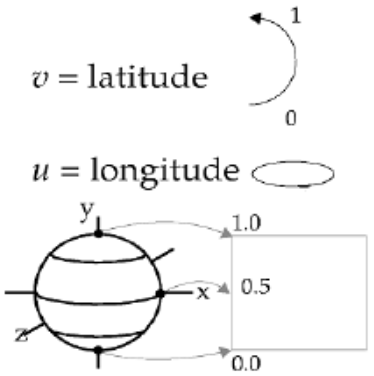
Текстуриране на сфера

- Изобразяване на точка от ***сфера*** в точка от единичния квадрат
  - да се определят координатите (*u*,*v*) на точка P
  - сферата се разглежда като съставена от множество окръжности с различен радиус
    - стойността на координатата *u* се определя както при цилиндър и конус
    - ако *v*=0 или *v*=1 се задава предварително дефинирана стойност на *u* - например 0.5
    - *v* е функция на “географската ширина” на P

$$v = \frac{\phi}{\pi} + 0.5$$



$$\phi = \sin^{-1}\left(\frac{P_y}{r}\right)$$



$$-\frac{\pi}{2} \le \phi < \frac{\pi}{2}$$

Текстуриране на сложни форми

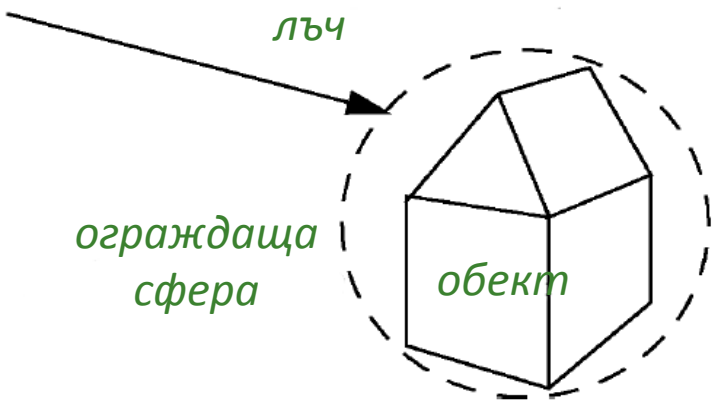
- Представянето на обектите с основни прости геометрични примитиви преди текстуриране не винаги води до очаквания резултат

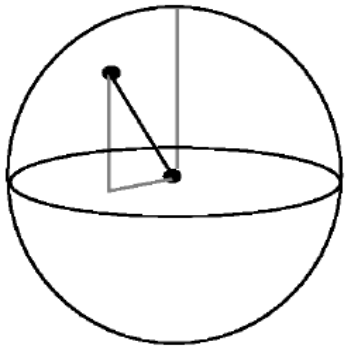
- Пример за относително несложен обект “къща”
  - текстуриране за всяка стена на къщата
    - прилага се текстура за равнинен многоъгълник
  - получават прекъсвания на границите между полигоните

- Интуитивен подход: редуциране на проблема до вече решен
  - за целите на текстурирането ***къщата се разглежда като сфера***

Текстуриране на сложни форми

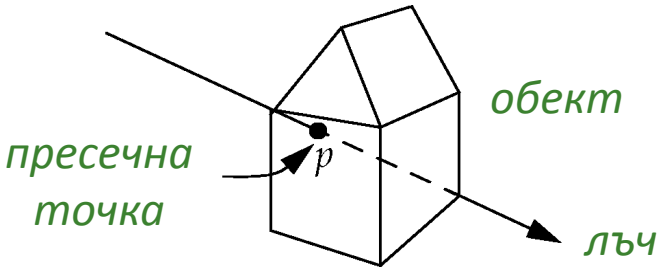
- ***Интуитивен подход***: ***ограждаща сфера на сложни обекти***
  - Текстурно изобразяване в две стъпки
    - Стъпка 1: Определя се пресечна точка на лъч с ограждащата сфера вместо с обекта в пространството на обекта
    - Стъпка 2: Определяне на uv-координати на пресечната точка
- ***Не-интуитивен подход***: точка от сложен обект се разглежда като точка от сфера и се проектира с използване на сферично uv-изобразяване

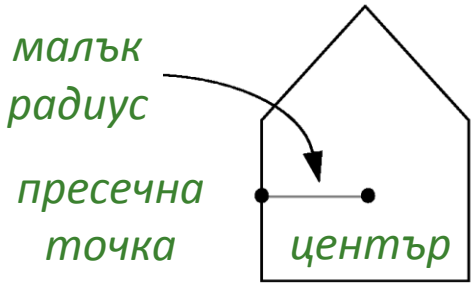


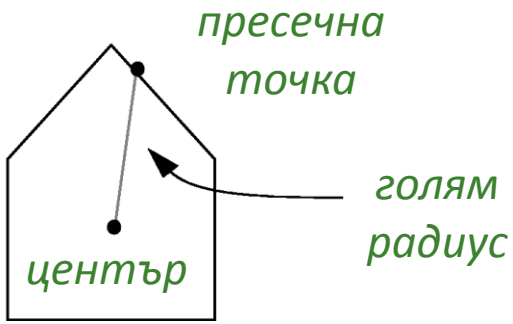


Текстуриране на сложни форми

- ***Интуитивен подход*: *ограждаща сфера на сложни обекти***
- Стъпка 1: Определя се пресечна точка на лъча с обекта
  - най-близката от пресечните точки с всяка стена
- Стъпка 2: Определят се uv координати на пресечната точка като точка от сфера
  - използва се сферична проекция
  - сферата има константен радиус, но обекта (къщата) – не
    - разстоянието от центъра на обекта до пресечната точка се променя с промяна на местоположението на точката

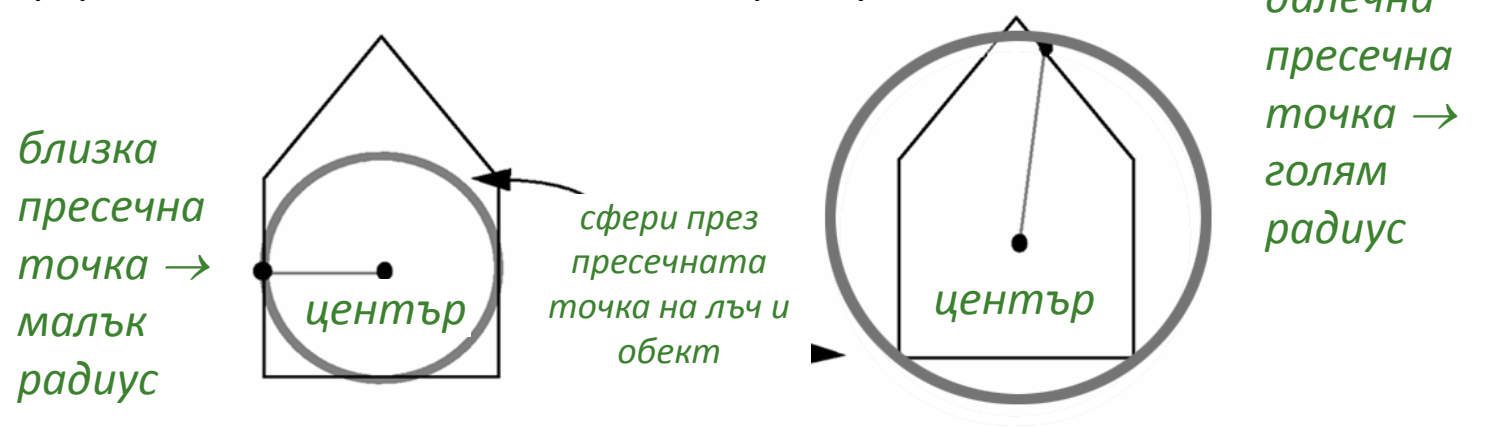






Текстуриране на сложни форми

- ***Интуитивен подход*: *ограждаща сфера на сложни обекти***
- Пресечната точка на лъча и обекта се разглежда като точка от сфера
  - сферата не винаги има един и същи радиус



- Радиус на ограждащата сфера
  - изчислява се радиусът като разстоянието от центъра на сложен обект до текущата пресечна точка
  - използва се тази стойност на радиусът за определяне на uv координати

Текстуриране на сложни форми

- Вместо сферично проектиране може да се използва друга функция за изобразяване на координати от пространство на обекта в uv координати

- Всяка проекция има недостатъци
  - сферична
    - изкривяване в полюсите на сферата
  - цилиндрична
    - несвързаност на контурите при основите
  - планарна
    - игнорира се едното измерение

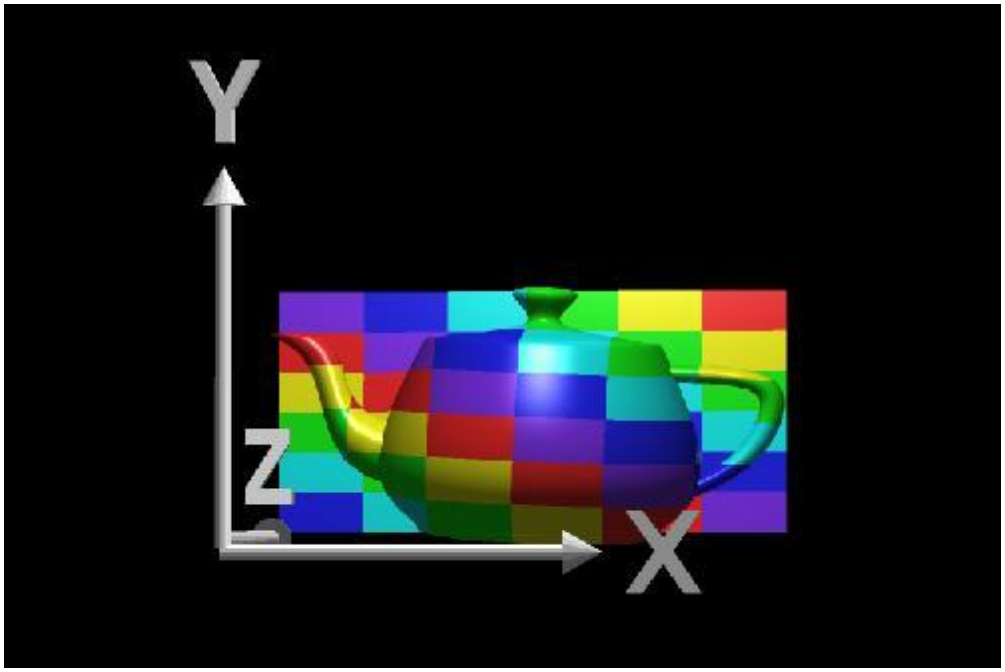
- Най-добри резултати
  - промяна на техниката за *uv*- проектиране
    в зависимост от това кои недостатъци са нежелателни

сферична проекция

планарна проекция

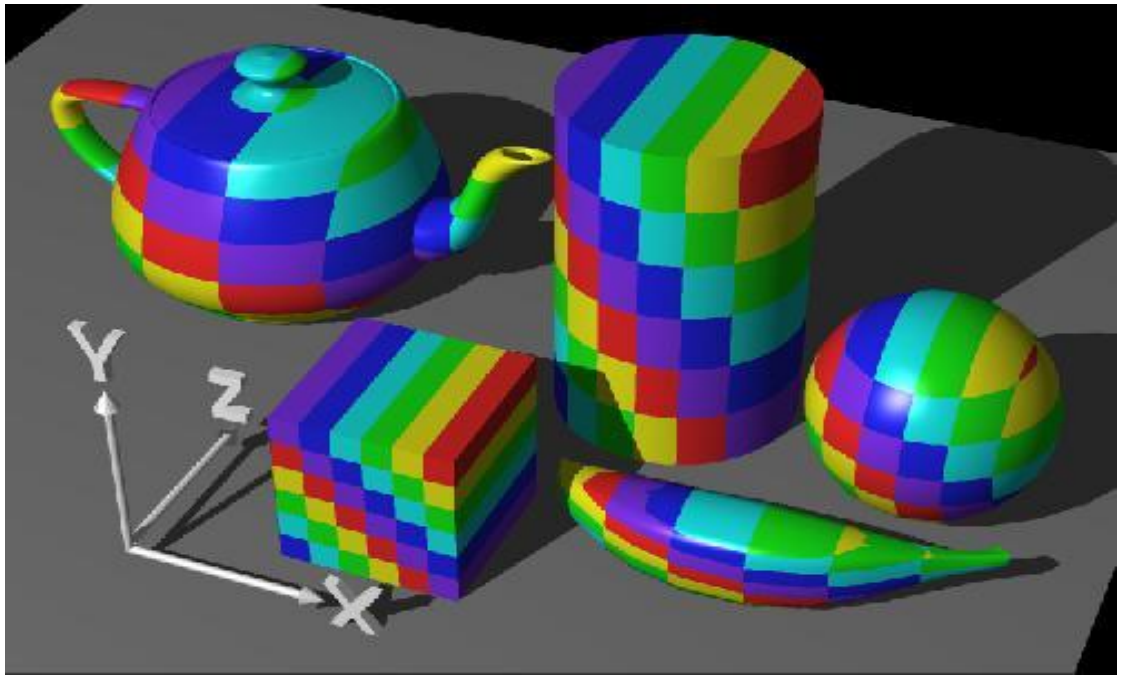
Проекция при текстуриране

- ***Планарна проекция***
  - премахва се едната от координатите на обекта
    - проекция в съответната координатна равнина



Проекция при текстуриране

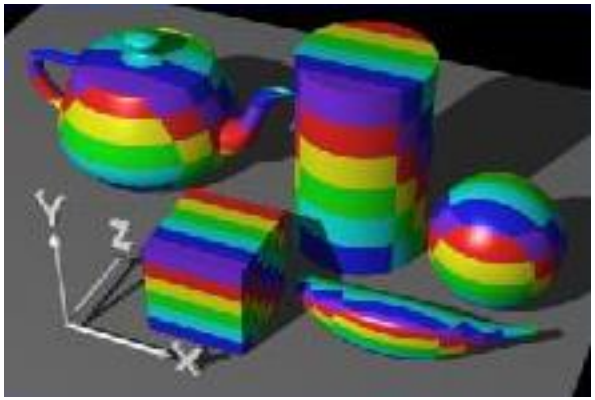
- ***Планарна проекция***
  - текстурата е константна в едно направление
    - не винаги се получава очаквания резултат

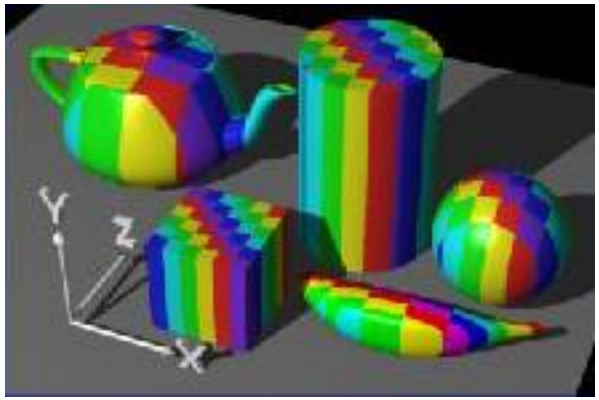


планарно проектиране на текстурата по z

Проекция при текстуриране

- ***Планарна проекция***
  - текстурата е константна в едно направление
    - не винаги се получава очаквания резултат





    - вместо по z може да се проектира по x или по y

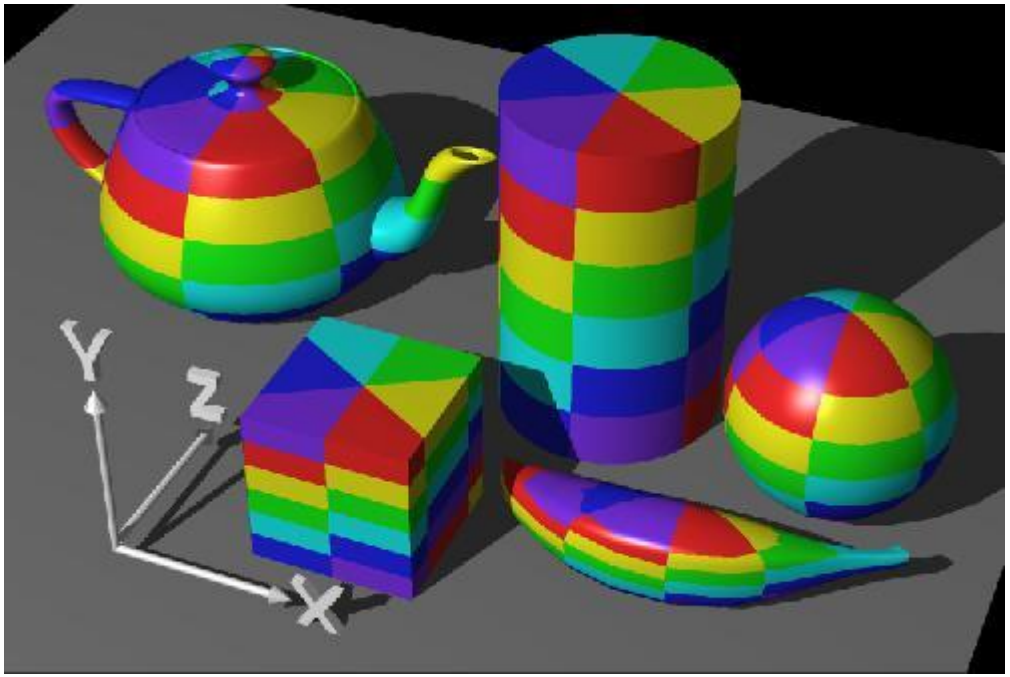
Проекция при текстуриране

- ***Цилиндрична проекция***
  - координатите (x, y, z) се преобразуват в (r, θ, h)
  - текстурата “обгръща” обекта

- стойността на θ се преобразува в координатата u
- стойността на h се преобразува в координатата v

Проекция при текстуриране

- ***Цилиндрична проекция***
  - за горната и долната страна на цилиндъра текстурата се “свива” (pinch)

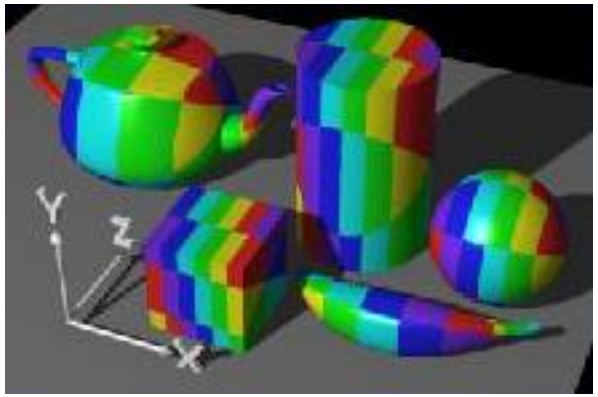


цилиндрична проекция

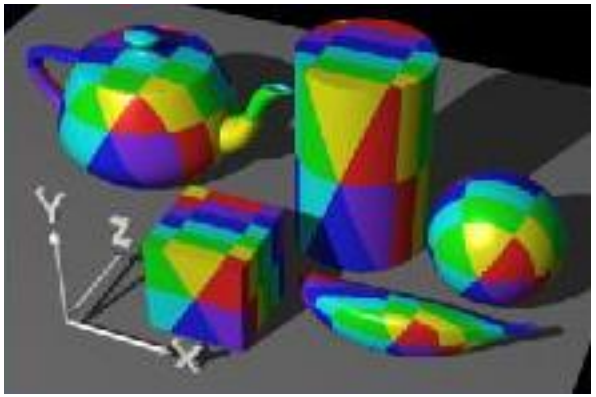
текстурата е паралелна на оста z

Проекция при текстуриране

- ***Цилиндрична проекция***
  - за горната и долната страна на цилиндъра текстурата се “свива” (pinch)



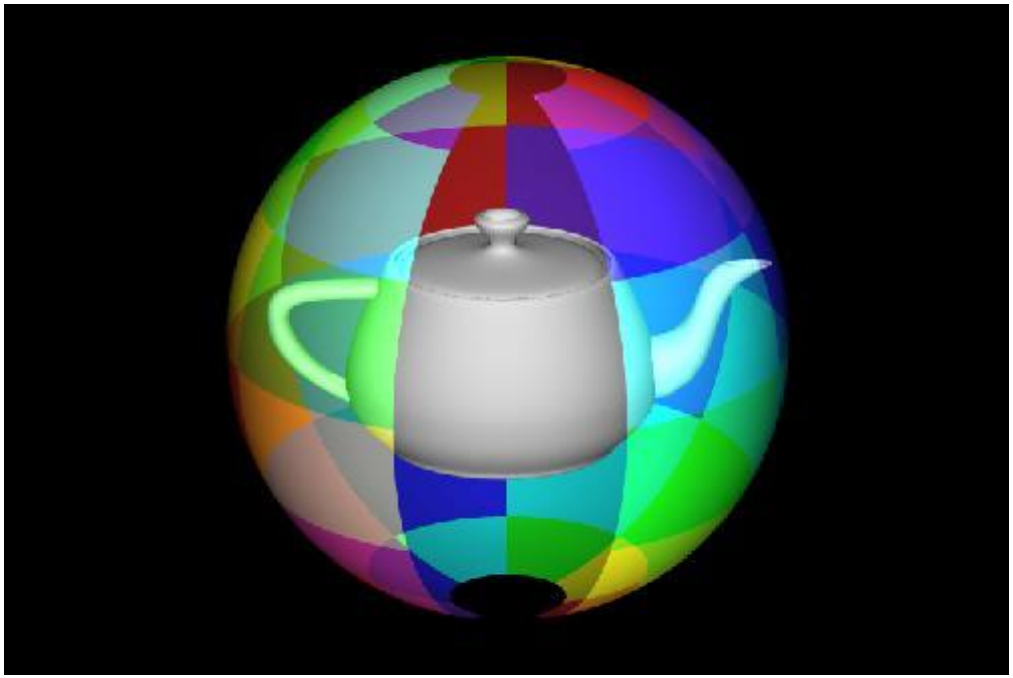
цилиндрична проекция по x



цилиндрична проекция по y

Проекция при текстуриране

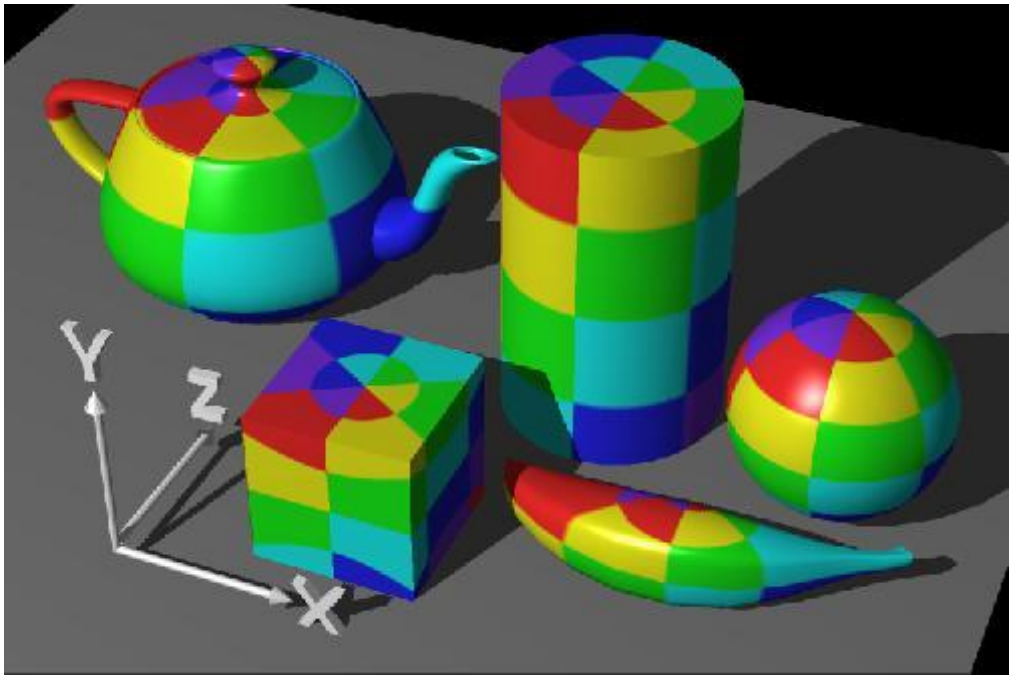
- ***Сферична проекция***
  - координатите (x, y, z) се преобразуват в сферични координати



- стойността на ширината (latitude) се преобразува в координатата u
- стойността на дължината (longitude) се преобразува в координатата v

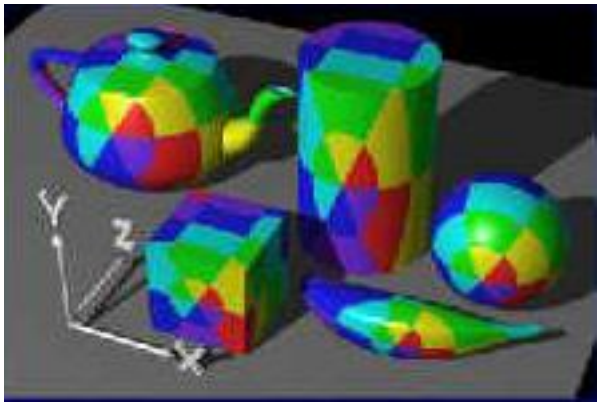
Проекция при текстуриране

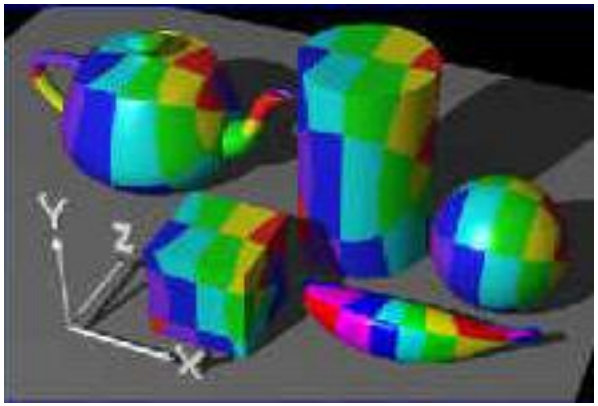
- ***Сферична проекция***
  - текстурата се “свива” в полюсите, но по различен начин в сравнение с цилиндричното проектиране



Проекция при текстуриране

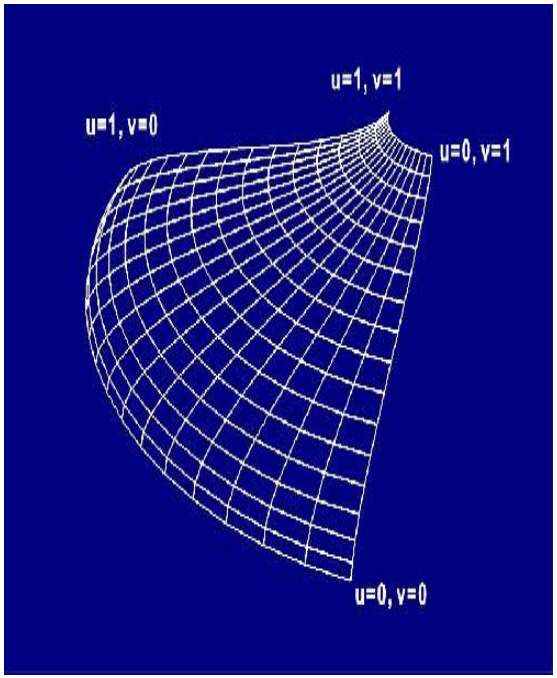
- ***Сферична проекция***
  - текстурата се “свива” в полюсите, но по различен начин в сравнение с цилиндричното проектиране
    - полюси в посока x и y

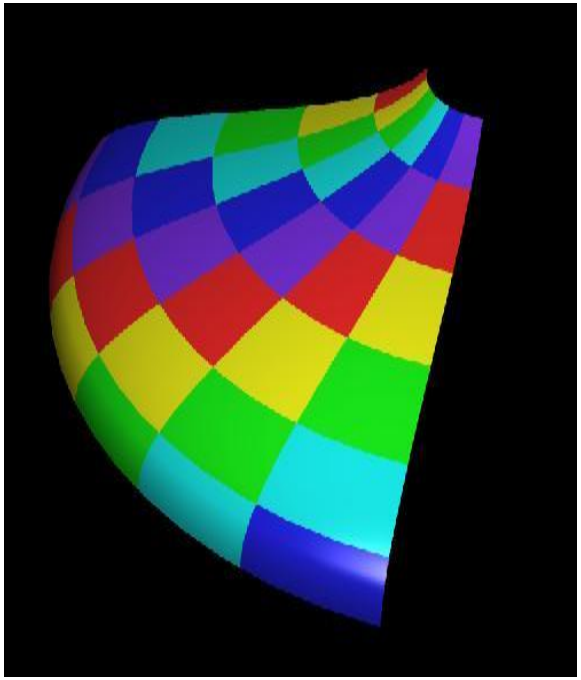




Текстуриране на параметрични повърхности

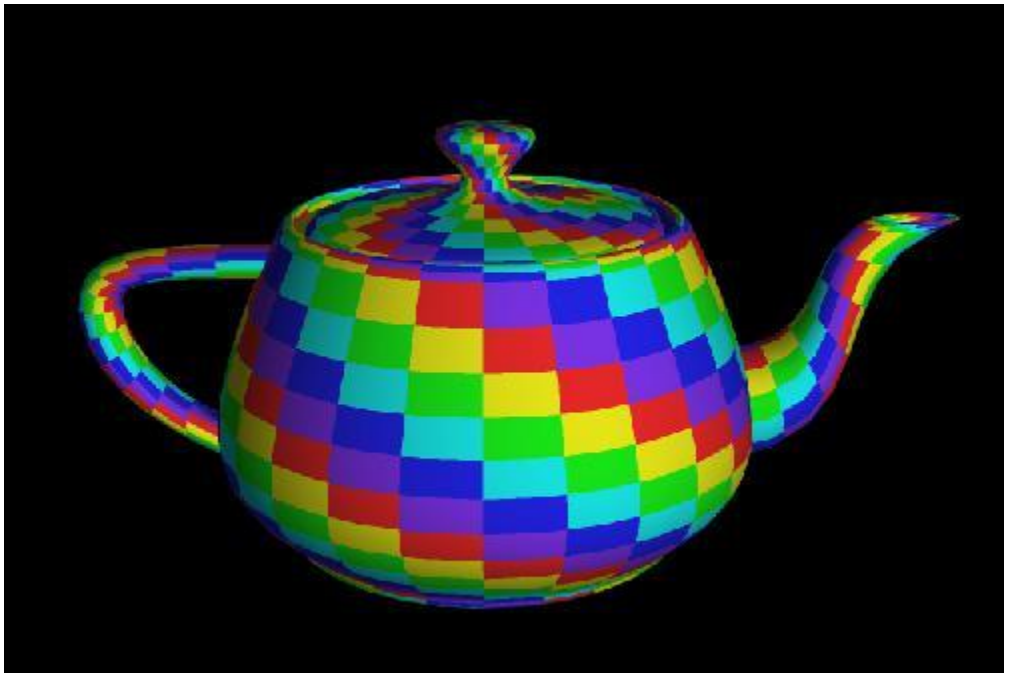
- Параметричните повърхности са параметризирани с (s, t)
  - параметрите (s,t) се използват като текстурни параметри (u,v)





Текстуриране на параметрични повърхности

32 параметрични области



Филтриране след текстуриране

- ***Тексел (Texel)***
  - рядко съответстват на пикселите
  - текстурите обикновено са правоъгълници,

    а полигоните при визуализиране обикновено не са
  - след прилагане на трансформации и текстурно изобразяване екранните пиксели могат да съответстват на
    - малък регион от текстурата (увеличение)
    - голям брой тексели (намаление)

увеличаване ***намаляване***

Филтриране след текстуриране

- ***Пост-филтрация***
  - няколко стойности от тектурата се филтрират преди прилагането им към фрагмент от полигон на обекта

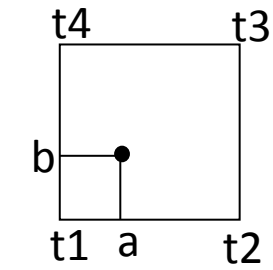
- ***Параметри на текстурното изобразяване***
  - Кои стойности на тексели да се използват за визуализиране при увеличение/намаление?
  - Каква интерполация (осредняване) да се направи за изчислените стойности?

Филтриране след текстуриране

- ***Интерполация по най-близък съсед***
  - използва се стойността на тексел, който е най-близо разположен до центъра на пиксела
  - изчислително просто
  - може да доведе до *aliasing*

- ***Билинейна интерполация***
  - претеглено осредняване на 4 тексела, които са най-близо разположени до центъра на пиксела

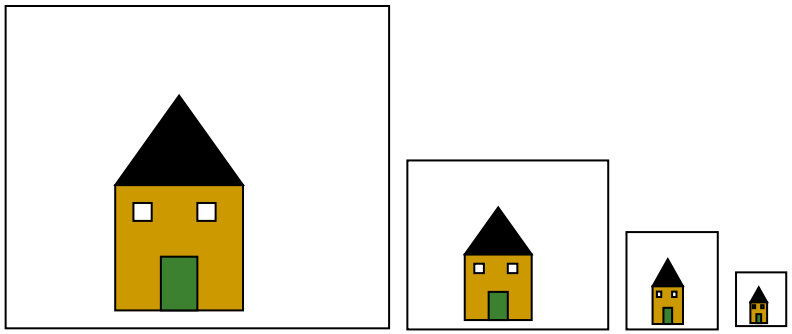
$$textureValue = (1-a)(1-b)t_1 + a(1-b)t_2 + abt_3 + (1-a)bt_4$$



  - по-голяма изчислителна сложност
    - хардуерна имплементация на 2D текстурно изобразяване предполага билинейна интерполация
  - по-плавни преходи и по-гладки резултати

Филтриране преди текстуриране

- ***MipMaping***
  - текстурираните обекти трябва да намаляват при отдалечаване от позицията на наблюдение
    - получават се сериозни дефекти дори и с използване на филтриране
      - колкото по-малко са пикселите от полигона, толкова по-малко са и стойностите от текстурата
  - решение: да се намали размера на текстурата преди изобразяването (намалява се разделителната способност)
    - по-малка по размер текстура се изобразява върху по-малък полигон

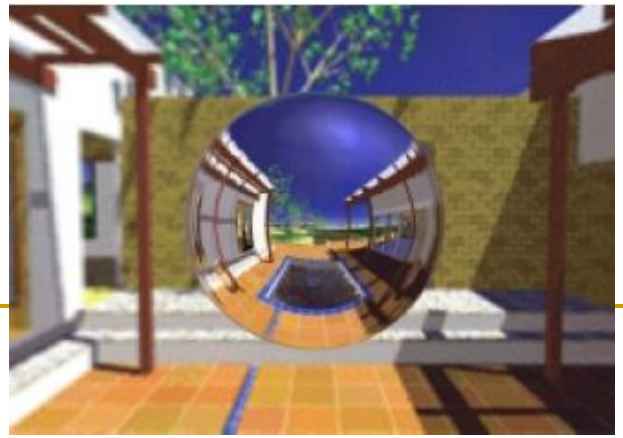


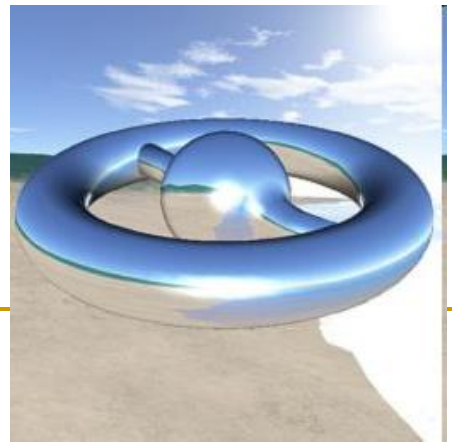
Mipmaping – от лат. дума mip (*multim im parvo*), което означава “много неща в малко място”

Mipmap текстури (всяка с ¼ размера на предишната

Environment Mapping

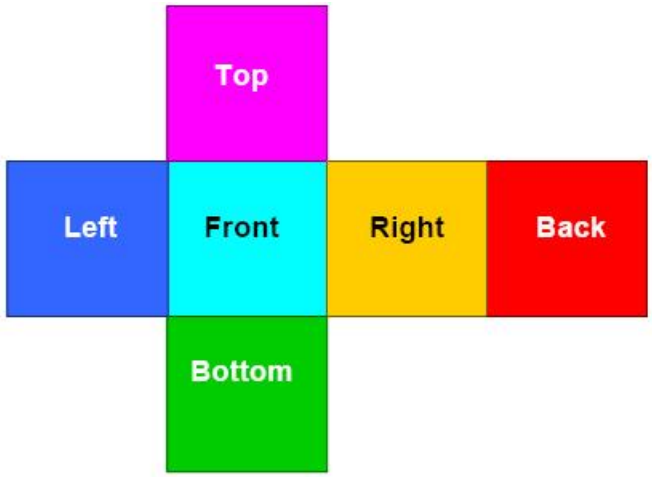
- ***Имитиране на отражение (Faking Reflections)***
  - симулира отразяване с текстуриране на ограждаща сфера или куб, които се разглеждат като груба апроксимация на околността
    - текстурата може да бъде фото-изображение или рендерирано изображение на сцената от позицията и перспективата на конкретен обект
  - прилага се при отразяващи повърхности
    - отражението е апроксимирано
    - по-неточно, но по-ефективно отколкото трасиране на лъчи
    - с рекурсия може да се добави само-отразяване (self-reflection)

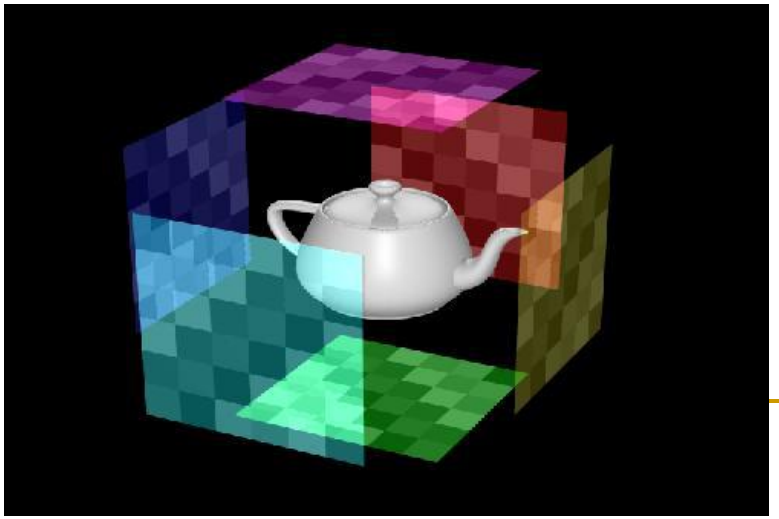




Environment Mapping

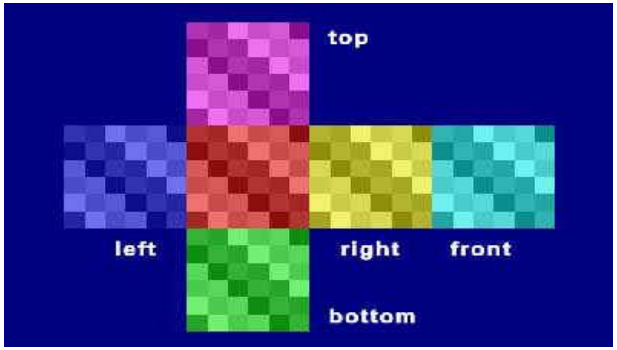
- ***Имитиране на отражение (Faking Reflections)***
  - ***Box map***
    - текстурата се създава чрез долепване на 6 проектирани текстури
    - сцената се рендерира 6 пъти като стените на куба са проекционни равнини
      - получените изображение са текстури на околния свят
    - нормалата на повърхността определя коя текстура да се използва
      - текстурните координати се определят от пресичането на лъч от позицията на наблюдение със стените на куба

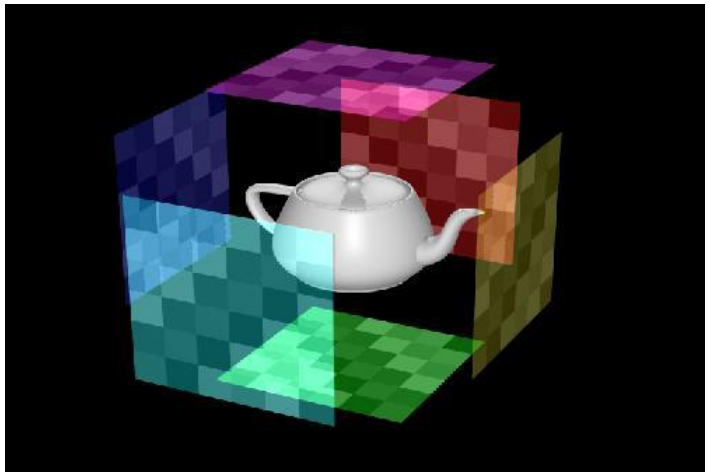


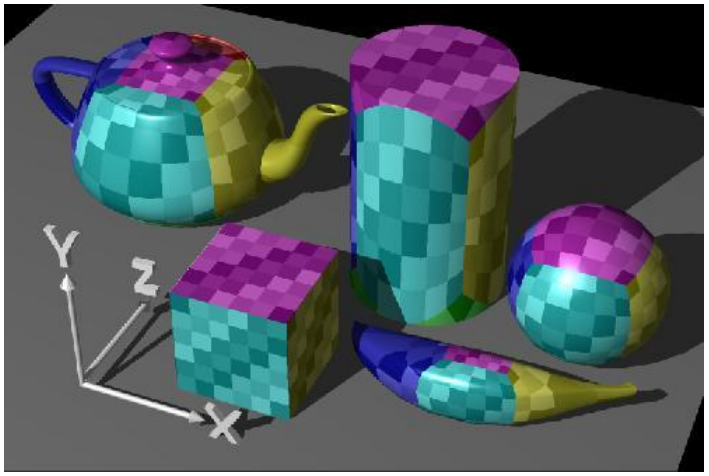


Environment Mapping

- ***Имитиране на отражение (Faking Reflections)***
  - ***Box map***

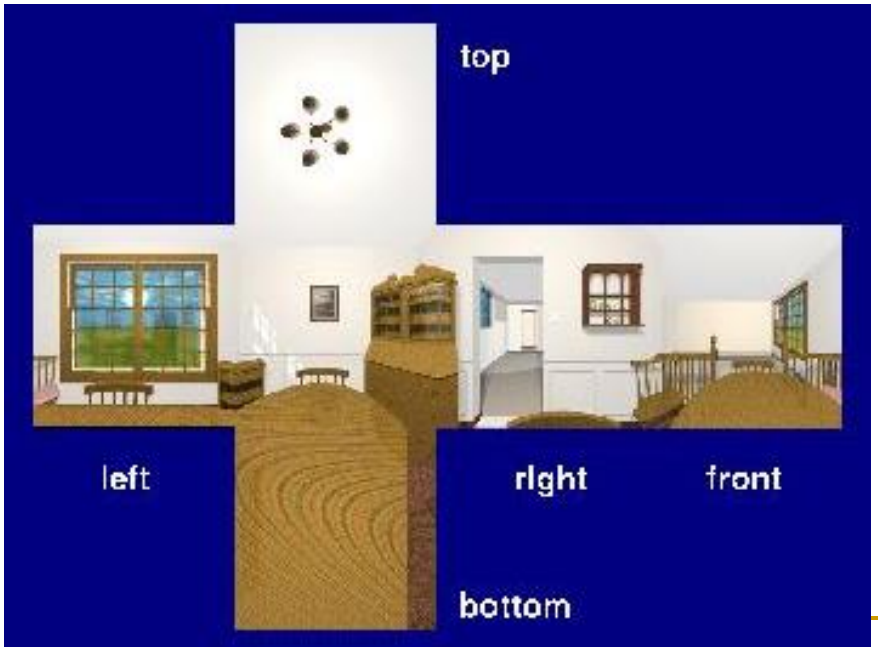






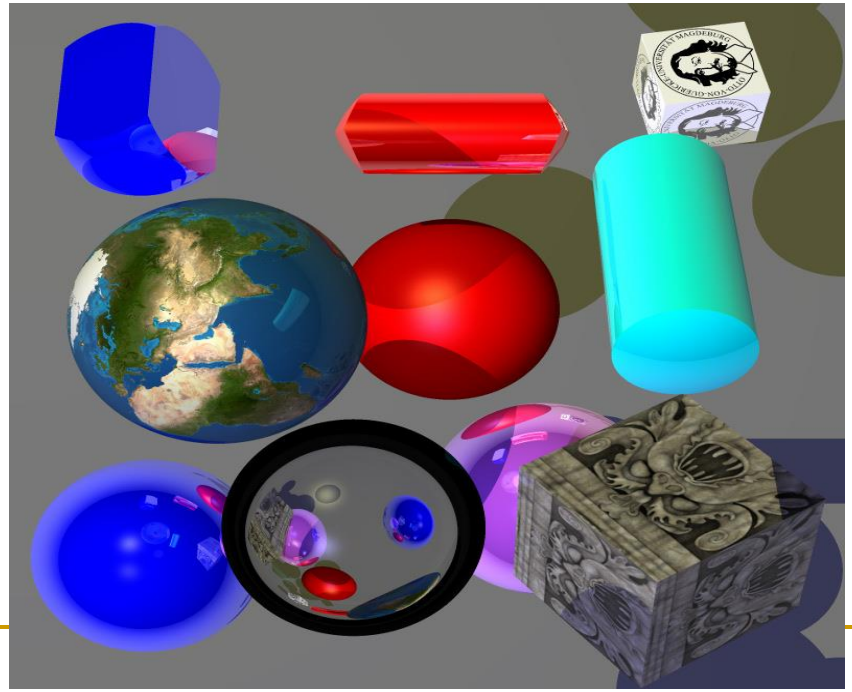
Environment Mapping

- ***Имитиране на отражение (Faking Reflections)***
  - ***Box map***
    - текстурите за отделните стени на куба могат да са различни



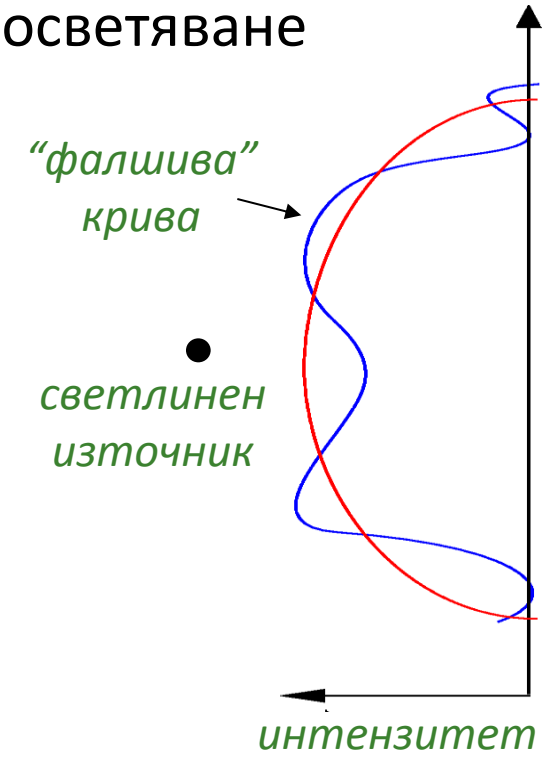
Environment Mapping

- ***Имитиране на отражение (Faking Reflections)***
  - За сфера текстурирането на околността е аналогично
    - проектира се текстурата на околността върху сфера
    - определя се пресечна точка със сферата за да се изчисли отражението



Bump Mapping

- Текстурирането на груби и грапави повърхности води до некоректно визуализиране на осветяване
- Метод на Blinn
  - променят се нормалите на повърхността (perturbation)
    - изчислява се градиента и се добавя към нормалата
  - получава се желания ефект във вътрешността на обектите, но не и по контурите

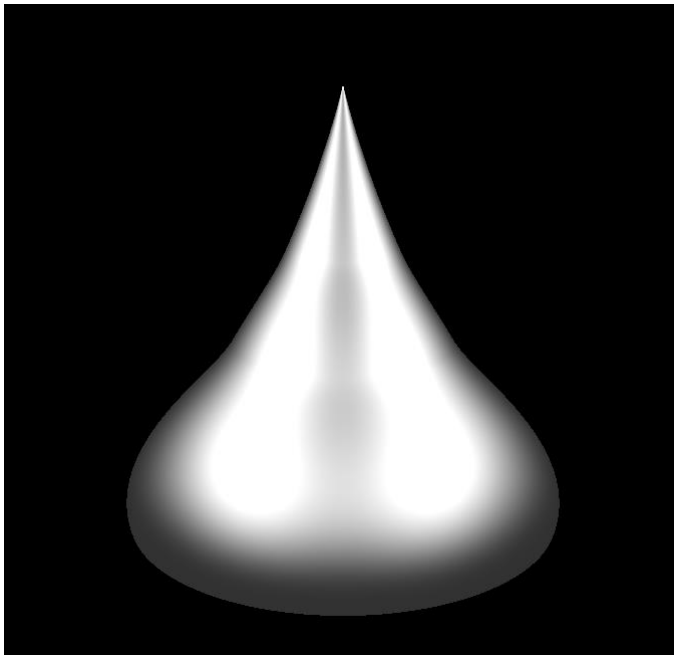


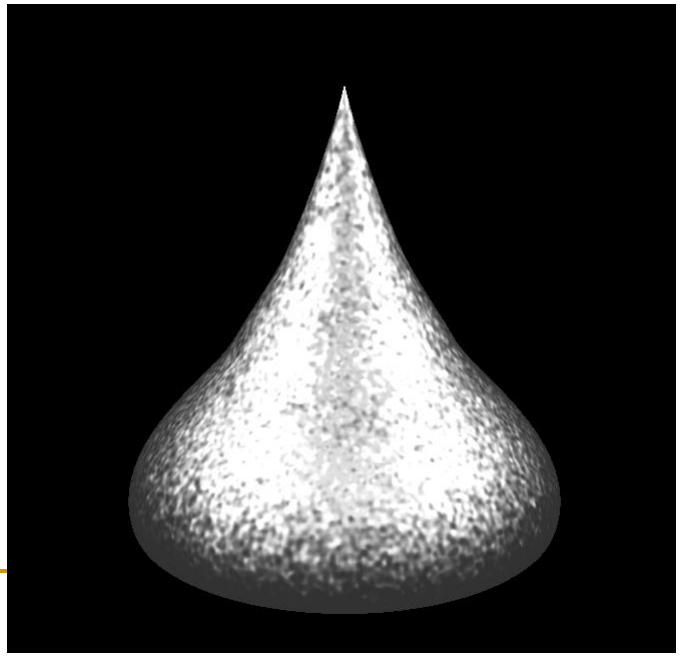
червена линия – стандартна дифузна полусфера

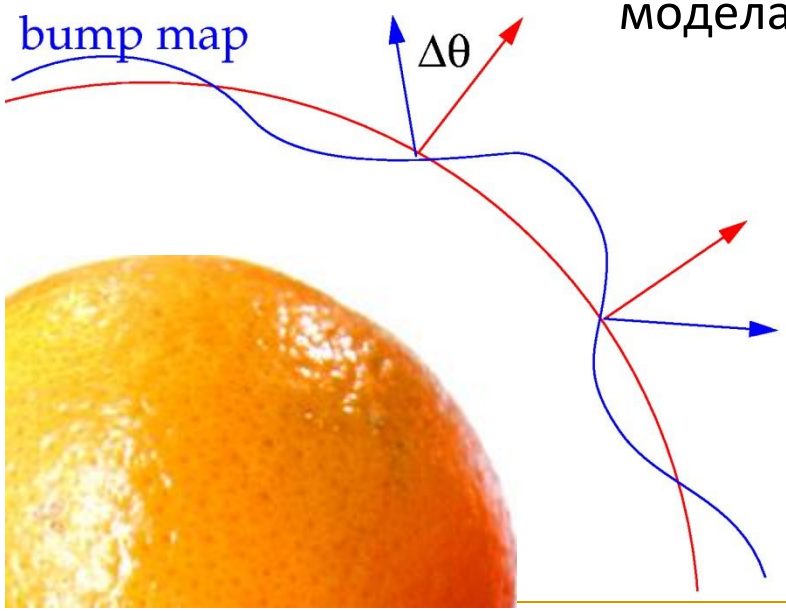
синя линия – теоретичен ефект на bump map

Bump Mapping

- Повърхност от станиол
  - твърде гладка
- Bump mapping
  - повърхност от станиол без да се усложнява модела







Абстракция на bump map на портокала

сферична повърхност

Normal Mapping

- ***Bump mapping***
  - използва единствен масив от стойности (grayscale map)
  - grayscale map се разглежда като теренна карта (height field)
  - нормалите ***се променят*** според градиента на височинната карта
  - осветеността се определя според нормалите
- ***Normal mapping***
  - подобна идея като тази на bump mapping, но с по-добри резултати
  - използва няколко стойности (multichannel или RGB map)
  - напълно ***се заменят*** съществуващите нормали
  - RGB стойностите на всеки пиксел съответстват на x,y,z компонентите на вектора на нормалата

Normal Mapping

- ***Предимства***
  - допустими са по-големи вариации в нормалите
  - нивото на детайли е ограничено от разделителната способност на картата на нормалите
    - а не от броя полигони в мрежата от модела на обекта
  - нормалите се определят тривиално
    - чрез таблица на съответствия (look-up table)
    - тази техника е приложима е в реално време
- ***Недостатъци***
  - добавените детайли не променят контурите
  - създаването на карта на нормалите не е лесна задача

Normal Mapping

- ***Приложение***
  - ***добавяне на детайли към прости геометрични мрежи***
    - модел с прости полигони и ниска резолюция се визуализира с по-висока разделителна способност

Геометричен модел на обекта (прости полигони, ниска резолюция)

Повече детайли от наличните в модела с използване на normal map

Normal Mapping

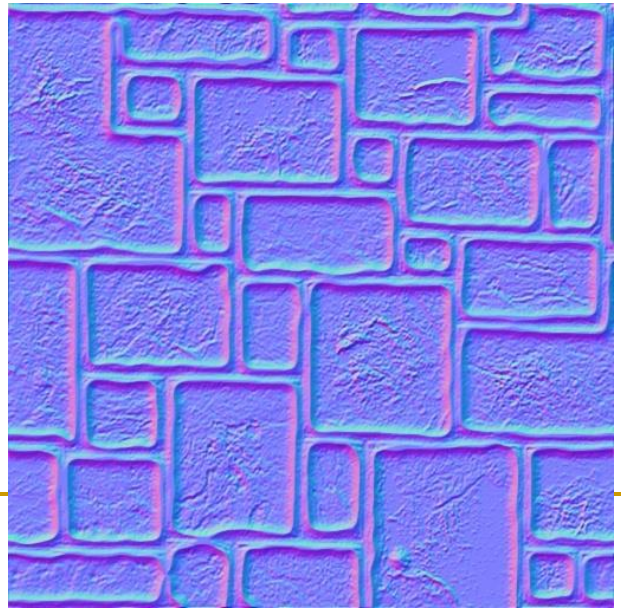
- ***Приложение***
  - ***почти пълна промяна на визуализацията на геометричен модел***

- *лява половина*
  - анизотропно отражение

- *дясна половина*
  - normal map
    - допълнителна “фалшива” геометрия
  - environment map
    - “фалшиво” отражение

Normal Mapping

- ***Създаване на normal map***
  - не може директно да се нарисува
    - цветовете са плавно променящи се
    - RGB стойностите са пространствено зависими
      - показват посоката на нормалата във всеки пиксел



Normal Mapping

- ***Създаване на normal map***
  - създава се модел с голяма разделителна способност
  - използват се нормалите на този модел за да се генерира карта на нормалите за модел с ниска разделителна способност
  - генерира се със специализиран софтуер
    - например
      - NVIDIA Plug-in for Adobe Photoshop (developer.nvidia.com/nvidia-texture-tools-adobe-photoshop)
      - Zbrush (www.pixologic.com)
      - GIMP (code.google.com/p/gimp-normalmap)
      - 3Ds Max, Maya, Blender, Lightwave, Cinema 4D, NormalMapper, xNormal
      - CrazyBump, Filter Forge, MindTex, ShaderMap, SSBump Generator

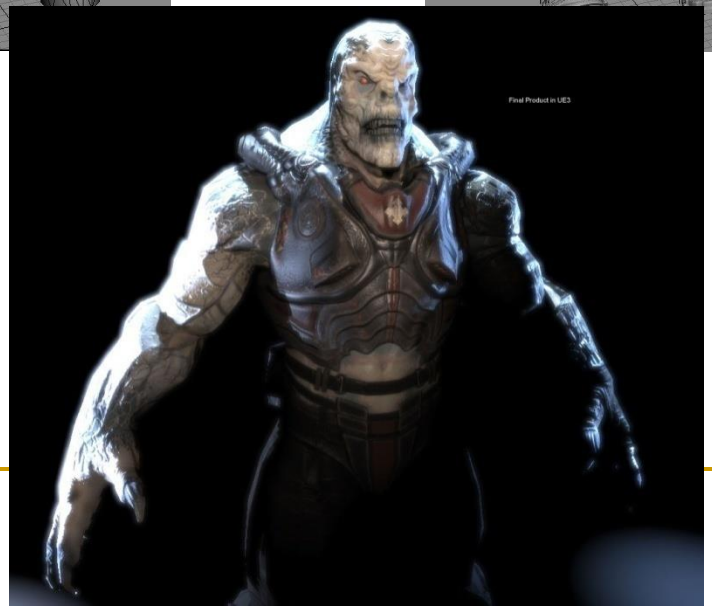
http://www.youtube.com/watch?v=RSmjxcAhkfE

Normal Mapping

Мрежа с малка резолюция (~5 000 полигона)

Модел с висока резолюция (2 млн. полигона)

3D геометричен модел



normal map

Текстури в OpenGL

- OpenGL предоставя функции за текстуриране

1. Дефинира се текстура

- създава се текстурата и се задава id
- специфицира се формат на данните и се задава текстурата

2. Специфицира се как ще се приложи текстурата

- задават се параметри на текстурното изобразяване
  - warping, филтриране
- задават се параметри за средата на текстуриране
  - комбиниране на стойностите в текстурата с текущия цвят на обекта

3. Разрешава се текстурното изобразяване и се съхранява текстурата в текстурна памет (на GPU)

4. Визуализират се обектите с използване на геометрични и текстурни координати

Текстури в OpenGL

- ***Дефиниране на текстури***
  - задаване на идентификатор за текстура
    - използването на текстурата за визуализиране е чрез идентификаторът

```
glBindTexture(GL_TEXTURE_2D, 3);
```

- ***Формат на текстурата***
  - формат за прочитане на входните данни

```
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
```

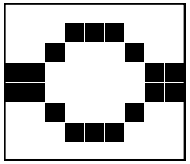
    - специфицират се данни с един байт за червена, един байт за зелена и един байт за синя компонента на цвета

Текстури в OpenGL

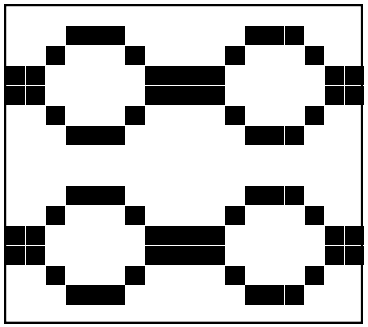
- ***Определяне на текстурни параметри***
- OpenGL има функция за задаване на различни параметри

```
glTexParameteri(target, pname, param);
```

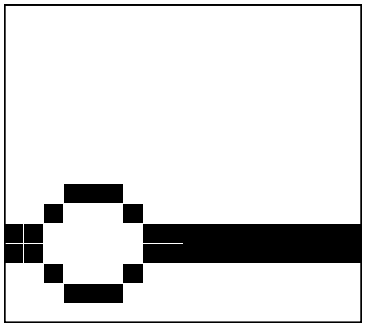
  - `target` е `GL_TEXTURE_2D`
  - `pname` е име на параметъра
    - `GL_TEXTURE_WRAP_T`
    - `GL_TEXTURE_WRAP_S`
    - `GL_TEXTURE_MIN_FILTER`
    - `GL_TEXTURE_MAX_FILTER`
  - `param` е стойност на параметъра
    - `GL_REPEAT`
    - `GL_LINEAR`



оригинална текстура



повторение и в двете направление (s и t)



изрязване и в двете направление (s и t)

Текстури в OpenGL

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

glTexParameteri (GL_TEXTURE_2D,
                 GL_TEXTURE_WRAP_T,
                 GL_REPEAT);

glTexParameteri (GL_TEXTURE_2D,
                 GL_TEXTURE_MAX_FILTER,
                 GL_LINEAR);

glTexParameteri (GL_TEXTURE_2D,
                 GL_TEXTURE_MIN_FILTER,
                 GL_LINEAR);
```

Текстури в OpenGL

- ***Определяне на текстурна среда***
  - задава се как ще се използва текстурата

```
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, param);
```

  - `param` е някоя от следните стойности
    - `GL_MODULATE` – използва осветяване
    - `GL_DECAL` – не използва осветяване
    - `GL_BLEND` – смесва цвета на обекта с цвета на фона
    - `GL_REPLACE` – използва само цвета на обекта

Текстури в OpenGL

- ***Подготовка на текстура***
- След като са зададени всички параметри текстурата може да се използва

```
glTexImage2D(target, level, internalformat, width,
                    height, border, format, type, pixels)
```

  - `target` е `GL_TEXTURE_2D`
  - `level` е степен на детайлност, подразбираща се стойност 0
  - `internalformat` е броя цветови компоненти в текстурата (`GL_RGB`)
  - `width` е ширината на изображението, трябва да бъде 2n + 2b, където е n е произволна стойност, b е размер на границата
  - `height` е височината на изображението, трябва да бъде 2m + 2b, където m е произволна стойност, b е ширината на границата
  - `border` е ширината на границата (0 или 1)
  - `format` е формата на данните (`GL_RGB`)
  - `type` е типа на данните (`GL_UNSIGNED_BYTE`, `GL_FLOAT`)
  - `pixels` е указател към изображението

Текстури в OpenGL

- ***Използване на текстурата***
- Текстурата може да се използва след обръщение към функцията `glTexImage2D`
- Разрешаване на текстурирането

```
glEnable(GL_TEXTURE_2D)
```

- Задават се текстурни координати за възлите

```
glTexCoord2f(u, v);
glVertex3f(x, y, z);
```

Текстури в OpenGL

- Пример

```
glBindTexture (GL_TEXTURE_2D, 13);
glBegin (GL_QUADS);
    glTexCoord2f (0.0, 0.0);
    glVertex3f (0.0, 0.0, 0.0);
    glTexCoord2f (1.0, 0.0);
    glVertex3f (10.0, 0.0, 0.0);
    glTexCoord2f (1.0, 1.0);
    glVertex3f (10.0, 10.0, 0.0);
    glTexCoord2f (0.0, 1.0);
    glVertex3f (0.0, 10.0, 0.0);
glEnd ();
```

Текстури в OpenGL

- Пример

```
void setupOpenGL (void) {
   glEnable (GL_TEXTURE_2D);
}
void loadAllTextures (void) {
   glBindTexture (..., 1);
   glPixelStorei (...);
   glTexParameteri (...);
   glTexEnvf (...);
   glTexImage2D (GL_TEXTURE_2D, 0, GL_RGB, imageWidth,
        imageHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, imageData);
   glBindTexture (..., 2);
   glPixelStorei (...);
   glTexParameteri (...);
   glTexEnvf (...);
   glTexImage2D (GL_TEXTURE_2D, 0, GL_RGB, imageWidth2,
        imageHeight2, 0, GL_RGB, GL_UNSIGNED_BYTE, imageData2);
   ...
}
```

Текстури в OpenGL

```
void drawTextureObjects (void) {
   glBindTexture (..., 1);
   glBegin (...);
       glTexCoord (...);
       glVertex (...);
   glEnd (...);

   glBindTexture (..., 2);
   glBegin (...);
       glTexCoord (...);
       glVertex (...);
   glEnd (...);
   ...
}
```

КРАЙ

Следваща тема:

Анимация в компютърната графика