Компютърна графика

Осветеност

доц. Милена Лазарова, кат. КС, ФКСУ

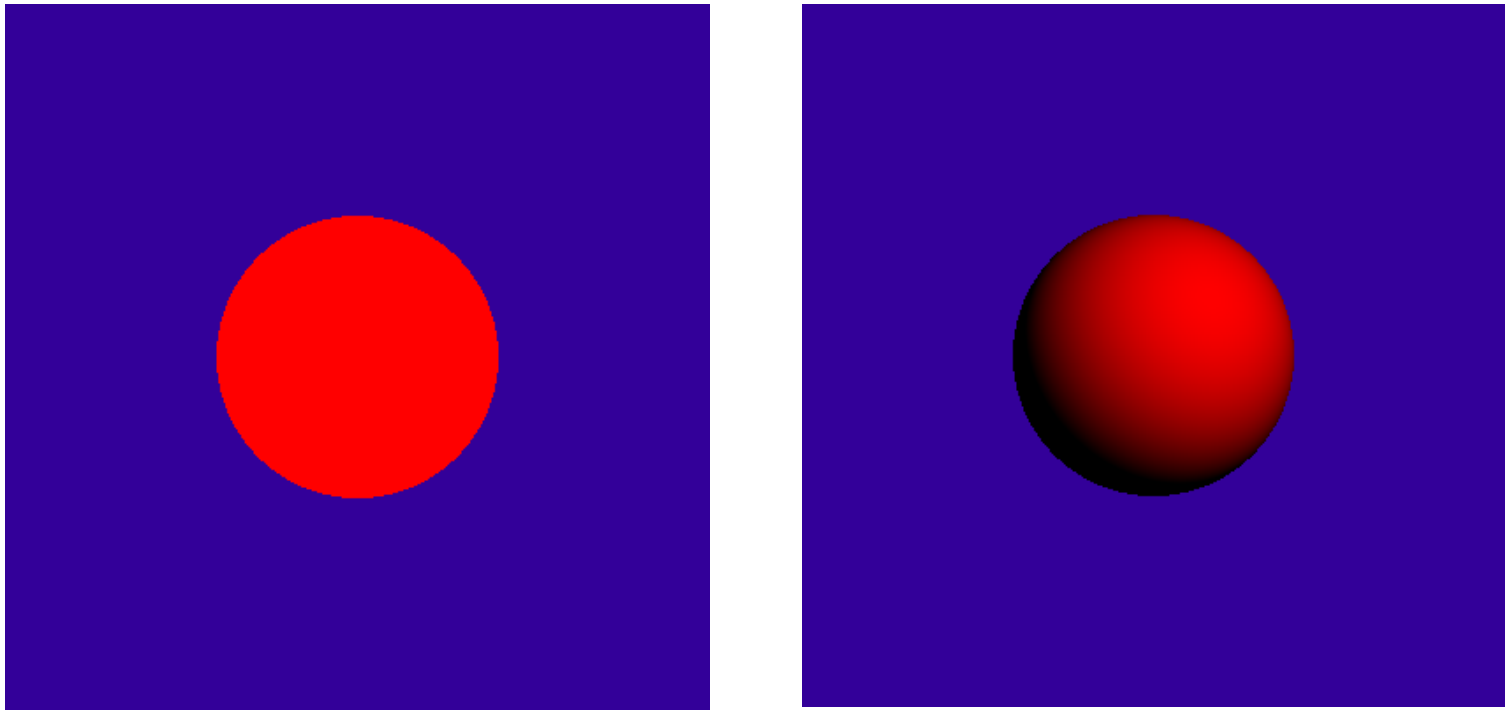
Реализъм в компютърната графика

Реалистични изображения

- Повърхностите и обектите трябва да се визуализират в “естествените” им цветове
  - цветовете, които ние бихме видели ако наистина наблюдаваме сцената
- Наблюдаваните цветове зависят от
  - геометричната форма и разположение на обектите
  - източниците на светлина в сцената
  - характеристиките на материала на повърхностите и обектите
    - взаимодействието на светлината с тези повърхности
      - отражение, абсорбиране, пропускане
- Нужен е модел на осветеност
  - ***Illumination, Reflection, Shading***

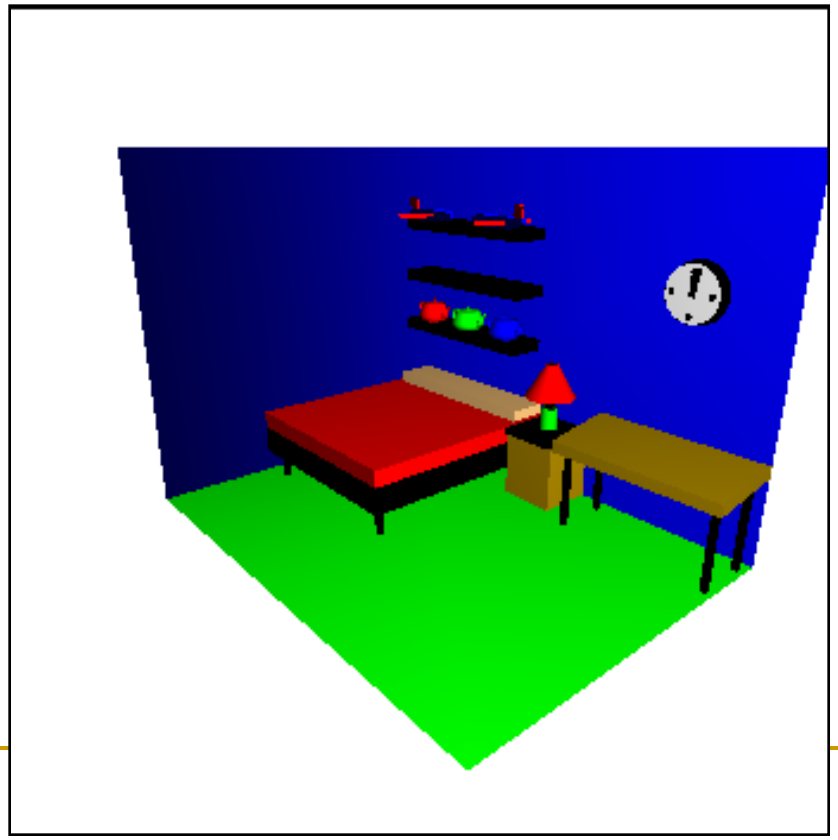
Осветеност

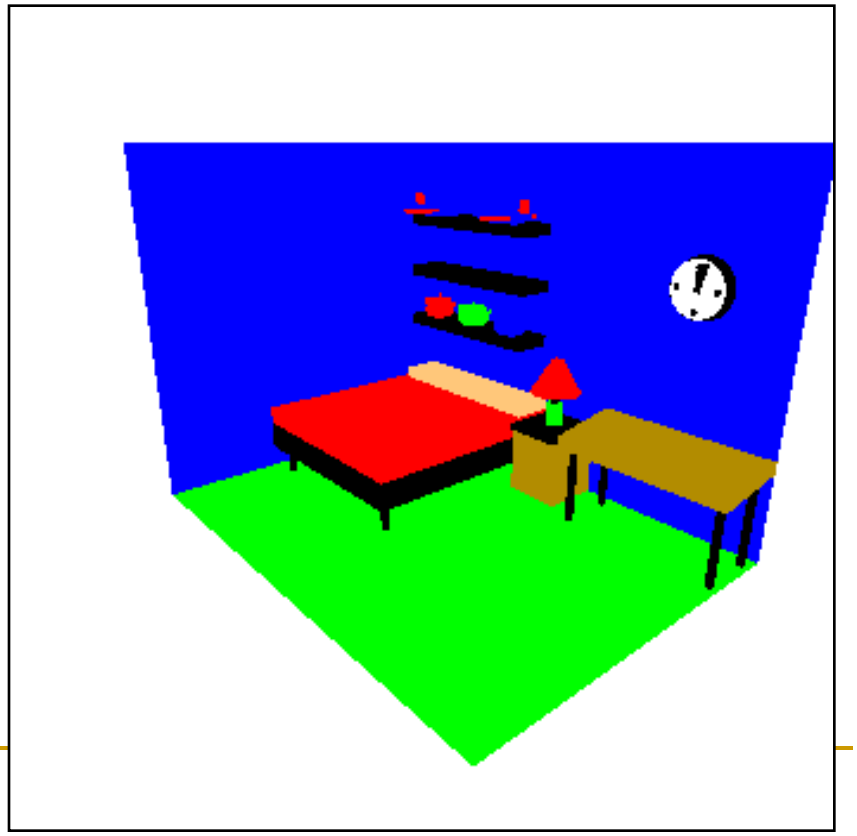
- Ако не се използва модел на осветеност обектите не изглеждат тримерни



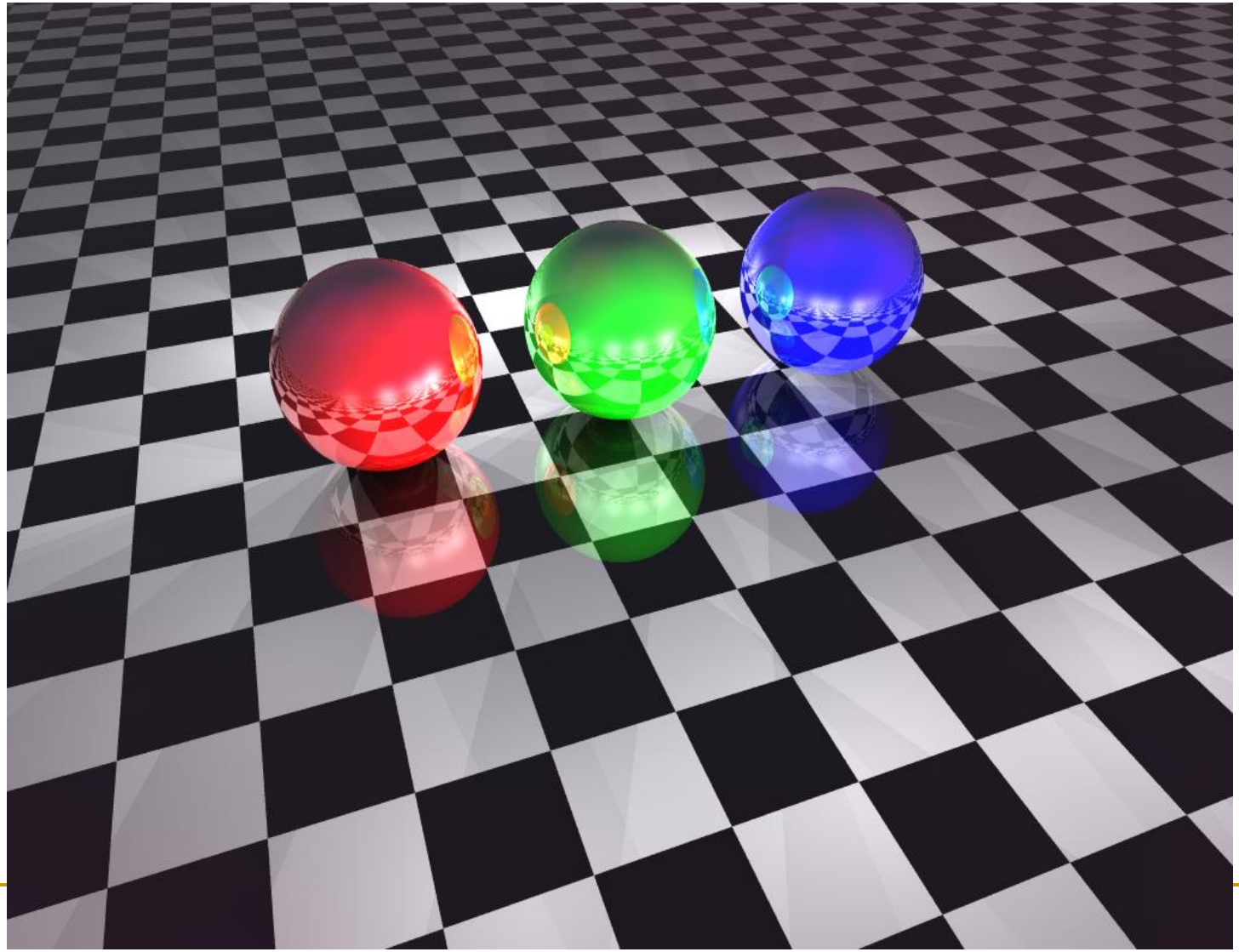
Осветеност

- Ако не се използва модел на осветеност обектите не изглеждат тримерни



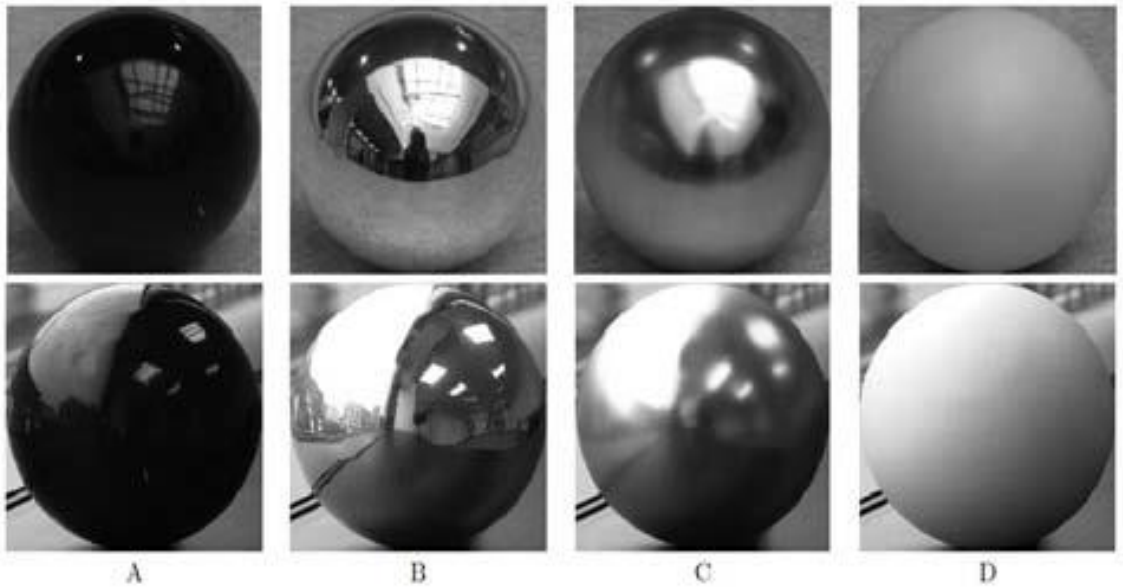


Осветеност



Осветеност

- Вариациите в наблюдаваните обекти зависят от взаимодействието на обектите със светлината



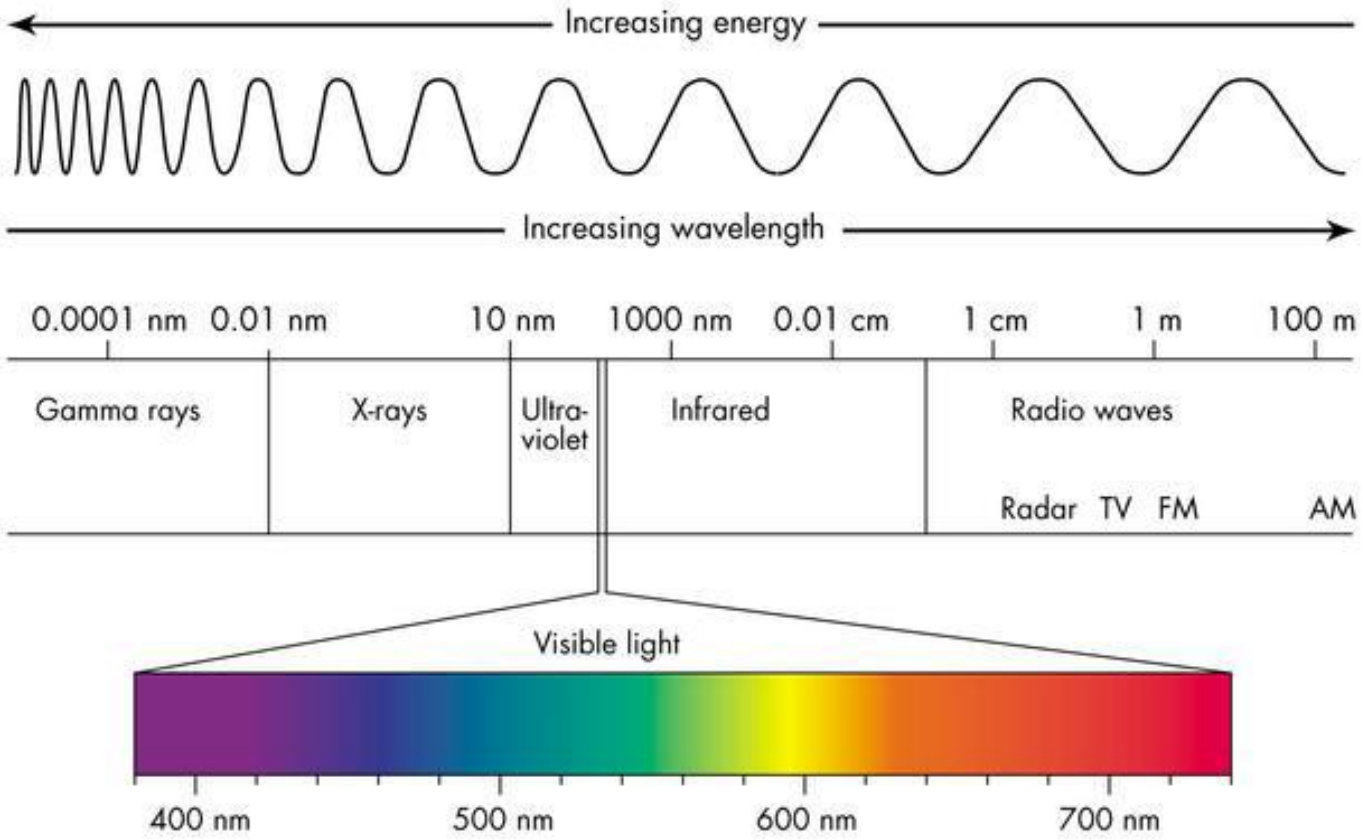
Светлина

- Светлината е електромагнитно (ЕМ) излъчване
  - електромагнитна енергия пренасяна чрез фотони – частици и вълни
    - дуалност вълна-частица в квантовата механика
  - оптиката е научна област от физиката, която се занимава с изучаване на светлината
- Вълните се дефинират чрез дължина и амплитуда на вълната
- Фотоните са частици без маса, носители на ЕМ енергия

Свойства на светлина

- скорост на светлината във вакуум
  - c = 299 792 458 m/s
- енергия на фотона
  - $E=h.f$
- константа на Планк
  - $h=6.6262*10^{-34}Js$
- дължина на вълната
  - $\lambda = c/f$
- честота на вълната
  - f

Електромагнитен спектър



- Светлина – видима част на спектъра

Осветеност

- За създаване на реалистично изглеждащи (или дори полу-реалистични) сцени трябва коректно да се моделира осветяването на сцената
  - да се генерира изображение
    - с определен геометричен модел
    - при определено осветяване
    - от определена позиция на наблюдение

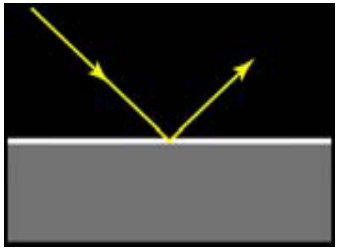
- Основен въпрос
  - как и колко светлина се отразява от обекта към наблюдателя

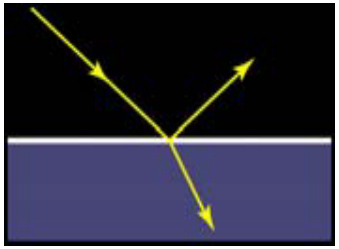
Осветеност

- Прости обекти

метал

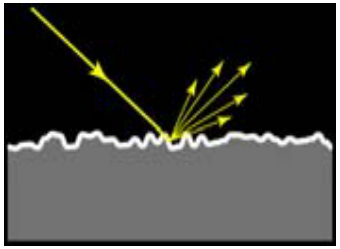
диелектрик

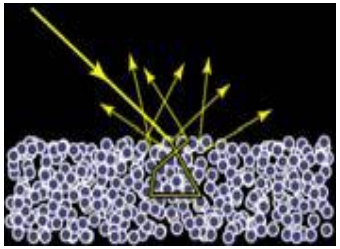




Осветеност

- Добавяне на микро-геометрия



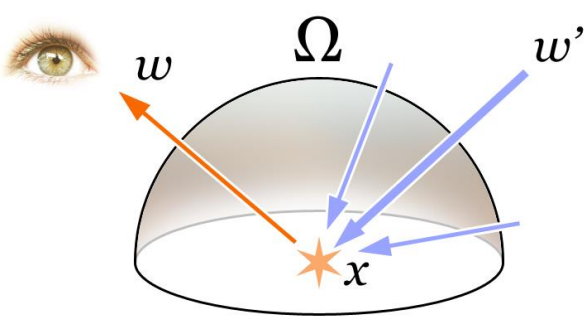


Модел на осветеност

- ***Общо уравнение за растеризиране с отчитане модел на осветеност***

  Rendering equation

  - формулирано от Jim Kajiya през 1986



  - *светлинна енергия от точка i до точка j е равна на светлината излъчена от i към j плюс интегрираното отражение S (за всички точки от всички повърхности) от точка k към i към j умножено със светлината от k към i, намалено с геометричен фактор за отслабване на интензитета*

Модел на осветеност

- ***Rendering equation***

$$L(i \to j) = G(i \leftrightarrow j)\left[L_e(i \to j) + \int_S f(k \to i \to j) L(k \to i) dk \right]$$

  - $L(i \to j)$ – количеството светлина движеща се по лъч от точка i към точка j
  - L_e – количеството светлина излъчена от повърхността (*luminance*)
  - $f(k \to i \to j)$ – BRDF (Bidirectional Reflectance Distribution Function) на повърхността
    - описва какво количество от светлината попадаща върху повърхността в точка i от посока на точка k се отразява от повърхността в посока на точка j
  - $G(i \leftrightarrow j)$ – геометричен член, който отчита закриване, разстояние и ъгъл между повърхностите

Модел на осветеност

BRDF (Bidirectional Reflectance Distribution Function)

Слънцето е зад наблюдателя (back scattering)

Слънцето е срещу наблюдателя (forward scattering)

Модел на осветеност

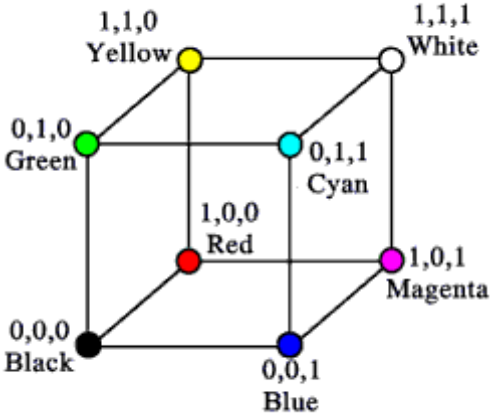
- Съществуват различно сложни и различно точни модели на осветеност
  - модели без физични основи, но резултатите изглеждат добре (“hacks”)
    - Phong, Blinn-Phong
  - сложни физично обосновани модели
    - Torrance-Sparrow, Cook-Torrance, Oren-Nayer
- Кой модел да се използва зависи много от графичното приложение
  - обикновено се прави компромис между *производителност* и *точност*

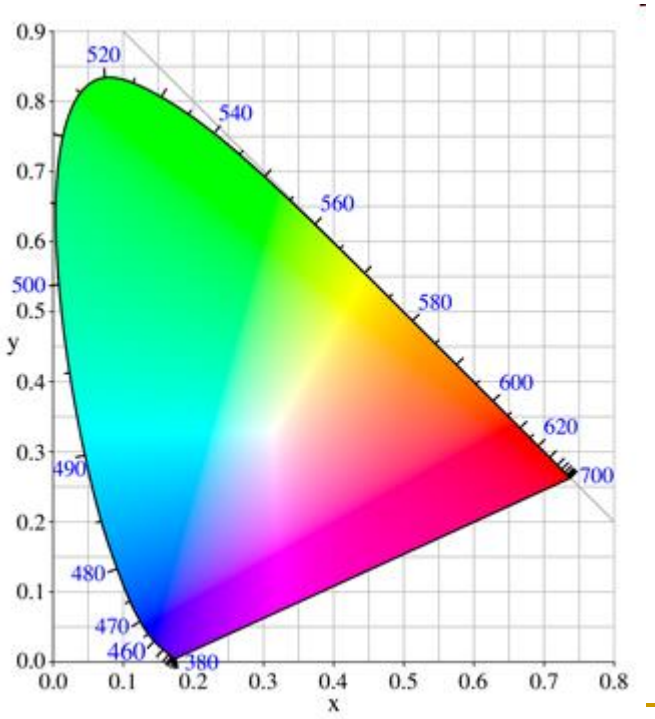
Модел на осветеност

- Доказано не съществува решение на общото рендиращо уравнение
  - въпреки, че такова се търси чрез симулации и други средства откакто светът съществува ☺
- Целта на фото-реалистичното визуализиране е да се намери апроксимация на уравнението, която да ”излъже” окото
  - за щастие човешкото зрение лесно се поддава на илюзии

Модел на осветеност

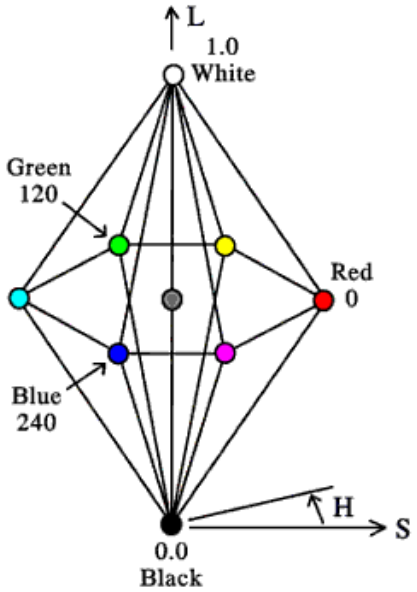
- Съществуват безкраен брой дължини на вълните, разпространяващи електромагнитна енергия
  - ограничаваме разглежданията си до три основни цвята
    - **червен**, **зелен**, **син**



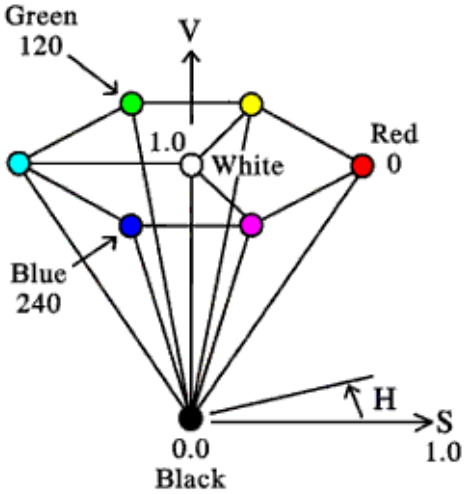


Цветово пространство CIE 1931

Цветови пространства

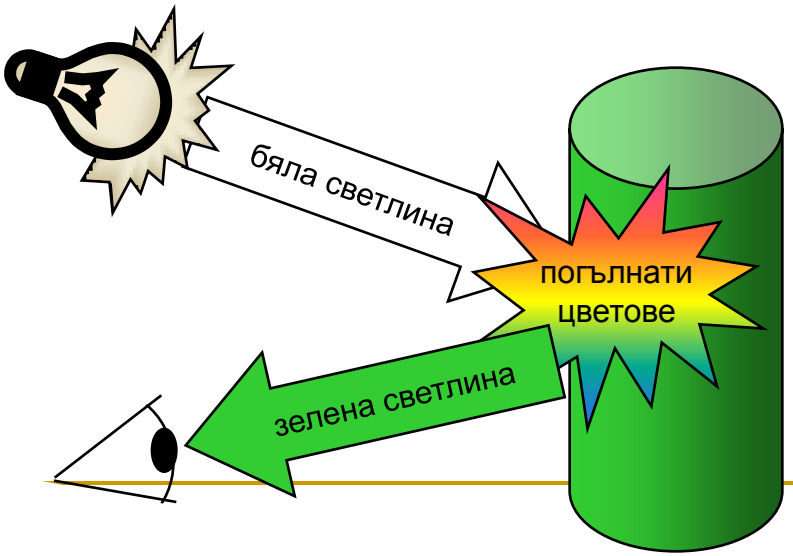


- ***Цветово пространство***
  - математически модел за представяне на цветовете чрез цветови компоненти (параметри)
    - обикновено 3 или 4 компоненти
- Основни компоненти на цветовото пространство
  - ***цветове***
    - червен-зелен-син: RGB или RGBA
      - A – прозрачност
    - циан-магента-жълт: CMY или CMYK
      - K – черен
  - ***параметри***
    - оттенък-наситеност-интензитет: HSI
    - оттенък-наситеност-стойност: HSV



Рефлективна светлина

- Цветовете, които човек възприема със зрителната си система, се определят от отразената от обектите светлината
  - светлина се излъчва към зелен обект
  - обектът поглъща енергията в повечето дължини на вълната
  - зелената светлина се отразява от обекта



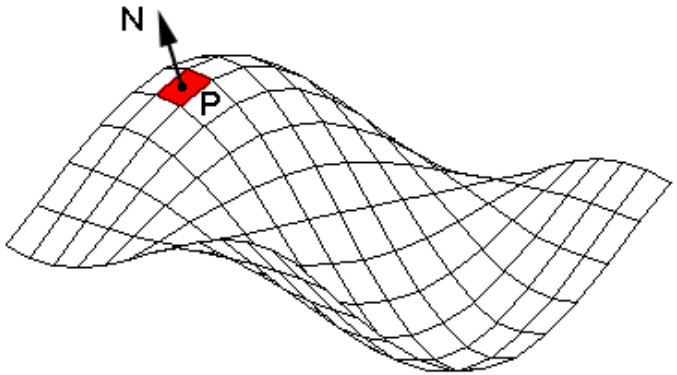
Модел на осветеност

- Околният свят е непрекъснат
  - ограничаваме се с дискретизирана версия
    - визуализират се пиксели

- Изчислява се осветеността на области от повърхности в реалния свят, които се преобразуват в данни за пиксели

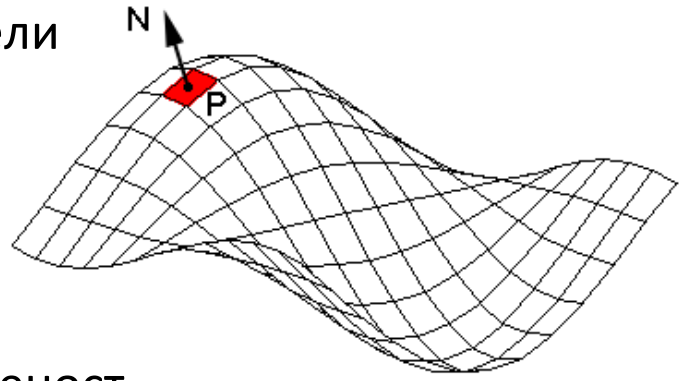
Елемент от повърхност

- ***Surface Element***
- Повечето повърхности са ***непрекъснати криви***
  - малки области от повърхностите могат да се апроксимират с части от ***тангенти равнини***
    - повърхността може да се раздели на краен брой много малки части
    - частите са също криви
    - но ако са достатъчно малки могат да са много близо до равнинни области



Елемент от повърхност

- Нормалите на повърхностите също са непрекъснато вариращи
  - всяка тангентна равнина дефинира елемент (област) от повърхността
  - всяка тангентна равнина има нормала
    - например сферата може да се раздели на триъгълници, като с всеки възел се асоциира нормала към сферата
    - малката област около възела е елемент от повърхността, който се използва за изчисляване на осветеност

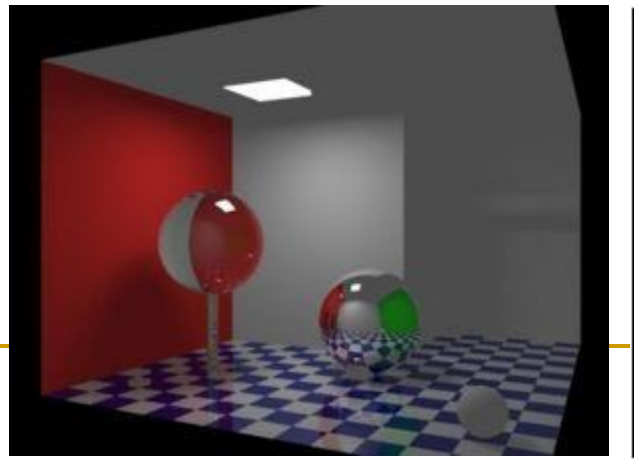


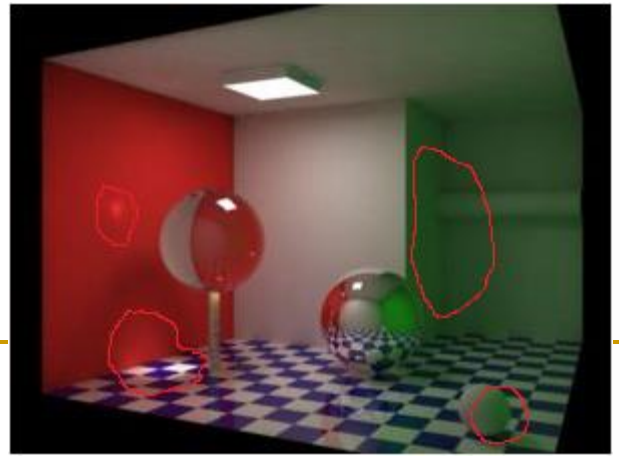
Осветеност

- ***Lighting (illumination, reflection)***
  - изчисляване на осветеност на базата на интензитет и дължина на вълната за дадена точка така както се вижда от позицията на наблюдение
    - функция на геометрията на сцената: модел, източници на светлина, позиция на наблюдение, характеристики на материала
  - *определянето на осветеността за всяка точка от сцената е изчислително сложно затова се интерполира*
- ***Shading***
  - определяне на цвета на точки чрез ***интерполиране*** между точките с известна осветеност
    - обикновено изчисленията се правят за върхове на триъгълници или полигони в мрежа
    - използва се за графични приложения в реално време (напр. игри)

Модел на осветеност

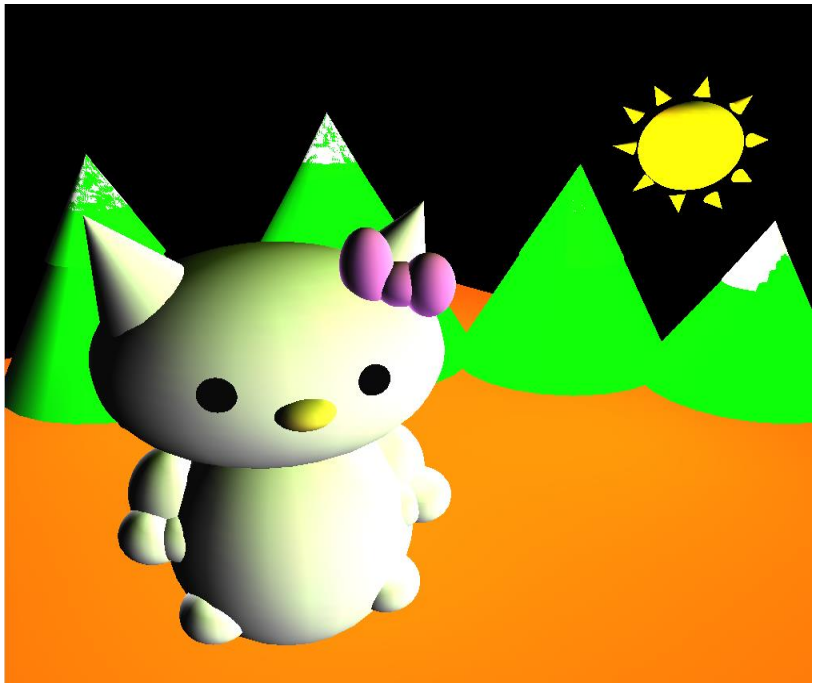
- ***Осветяване на област от повърхност***
  - определя се резултат за модела на осветеност в тази област така както се вижда от позицията на наблюдение
- ***Глобални модели***
  - в общото уравнение за растеризиране се отчита глобална информация
    - най-реалистичните модели на осветеност използват глобални данни
- ***Локални модели***
  - също водят до правдоподобни резултати, но на много по-ниска цена





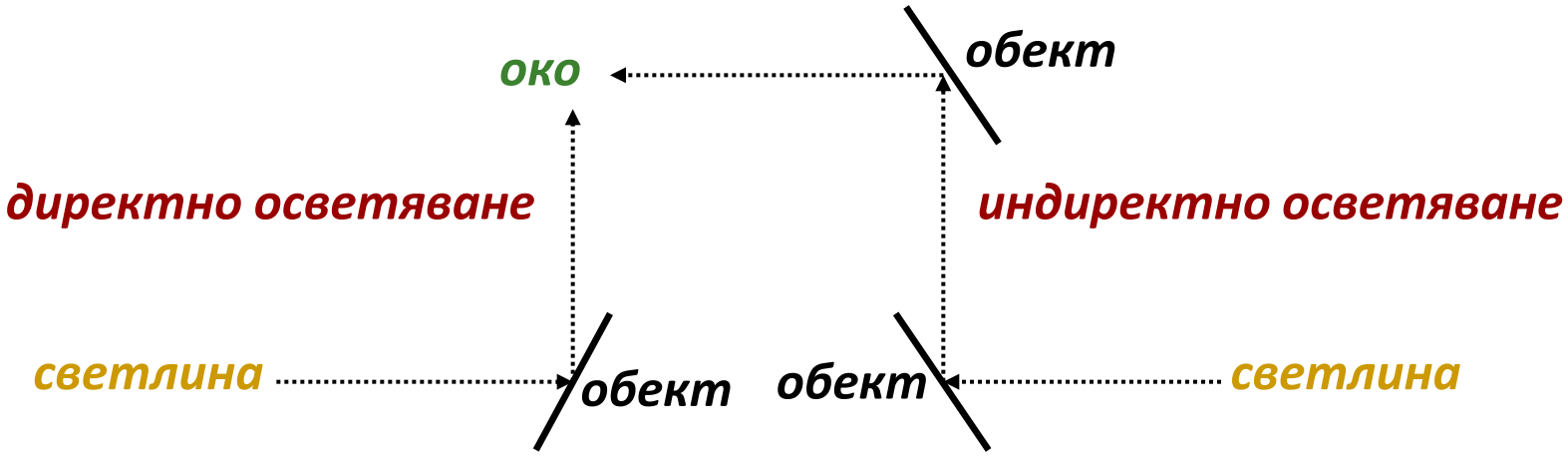
Модел на осветеност

- ***Локален модел на осветеност***
  - използва само директна информация за осветяването
  - представлява апроксимация на глобалната осветеност
  - обикновено в модела се използва член за ***обкръжаващата светлина*** за да се зададе минимално осветяване на обектите



Модел на осветеност

- ***Глобална осветеност***
  - симулира влиянието на другите обекти и елементи в сцената върху светлината достигаща до елемент от дадена повърхност



Модел на осветеност

- ***Светлини и сенки***
  - по-голяма част от светлината стигаща до обект идва директно от източници излъчващи светлина
    - ***директно осветяване (direct illumination)***
  - понякога светлината от светлинен източник се блокира от други обекти
    - обектът попада в “сянка” от този източник на светлина
- ***Отразено осветяване между обектите***
  - светлината от източниците се отразява от обектите в сцената преди да достигне до повърхността на даден обект
    - ***индиректно осветяване (indirect illumination)***

“Luxo Jr.”, Pixar

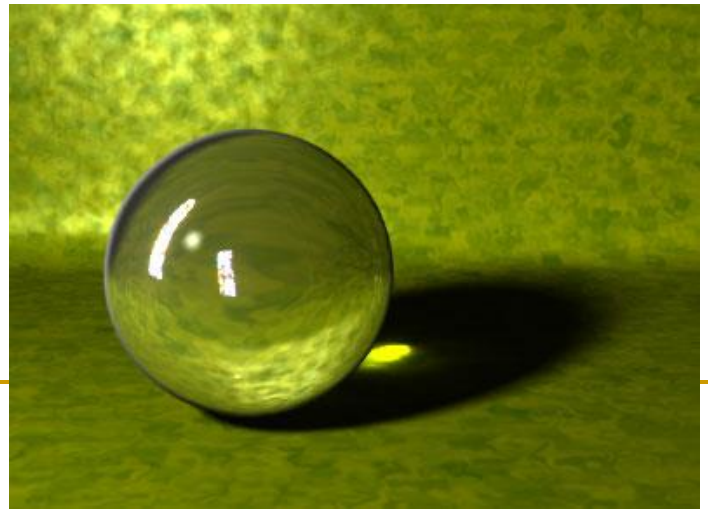
Модел на осветеност

- ***Локални модели***
  - светлина само от директни източници
    - ***предимства***
      - сцената може бързо да бъде визуализирана
    - ***недостатъци***
      - загуба на реализъм – липсват интересни светлинни ефекти поради игнориране на влиянието на другите обекти в осветяването на дадена повърхност

- ***Глобални модели***
  - обхващат цялата информация за осветеността
    - ***предимства***
      - сенки, отразено осветяване между обектите, пречупване, полупрозрачни обекти, ефекти от въздух, вода, мъгла
    - ***недостатъци***
      - бавно визуализиране

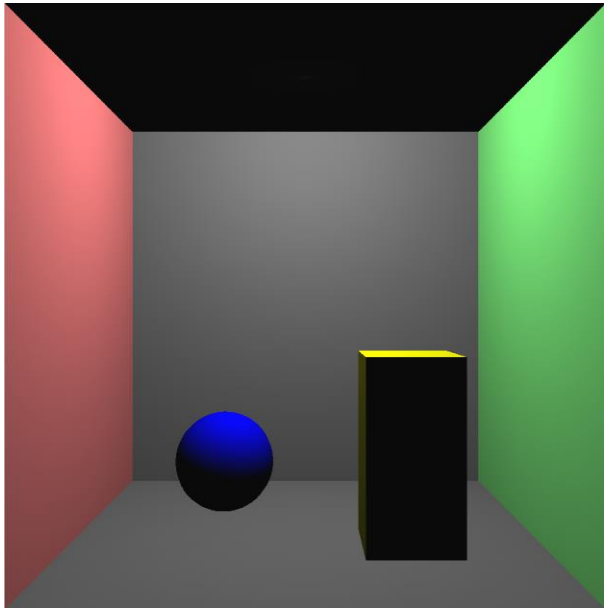
Изчисляване на модел на осветеност

- ***Рендериране на полигони***
  - ***Shading***
  - изчисляване на осветеността в няколко точки и интерполиране във всички останали пиксели за получаване на крайното изображение
- ***Симулация на движението на светлината***
  - ***Lighting***
  - изчисляване на осветеността в достатъчно точки за получаване на крайното изображение без да се интерполира



Изчисляване на модел на осветеност

- ***Рендериране на полигони***
  - използва се в приложения изискващи изчисления в реално време (игри)
  - изпълнява се от GPU
  - по-лошо качество
- ***Симулация на движението на светлината***
  - използва се за висококачествено фото реалистично визуализиране
  - някои имплементации работят в реално време на GPU, но по-сложни модели на осветеност изискват изпълнение от CPU
  - визуализация с най-високо качество може да отнеме дни



Модел на осветеност

- Част от светлината достигнала до обекта се отразява и се връща обратно към наблюдателя
  - отражението е абсорбиране и бързо излъчване на фотони
  - посоката на отразяване зависи от материала
  - посоката на отразената светлина зависи от посоката на падащата светлина

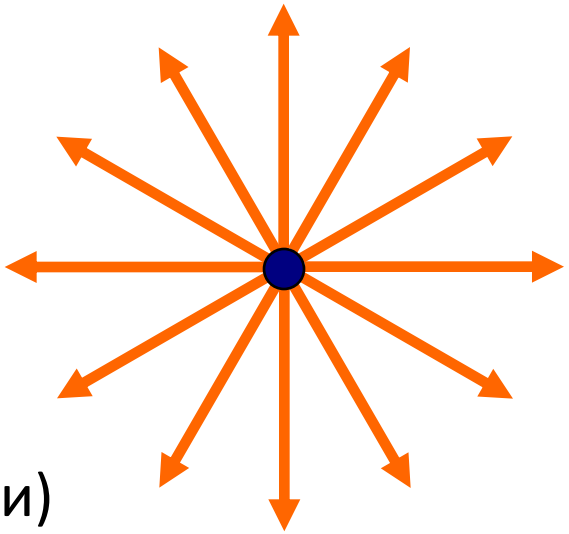
- ***BRDF*** (Bidirectional Reflectance Distribution Function)
  - функция за измерване на двупосочно разпространение на отражението
    - отразява зависимостта между входни стойности за
      - ***интензитет***, ***дължина на вълната*** и ***ъгъл на падане*** на светлината върху обекта и
      - ***ъгъл на вектор на наблюдение***, ***интензитет*** и ***дължина на вълната*** на отразената светлина

Източници на светлина

- Два основни вида
  - ***Обкръжаващи светлинни източници*** (***ambient***)
    - ***ненасочени***, дифузни, фонова светлина
      - приемат се за константа в сцената
      - грубо апроксимират многократно отразената светлина
      - “глобално” отразяване
  - ***Точкови светлинни източници*** (***spotlight***)
    - апроксимират се чрез серия от точкови източници
    - ***насочени***

Точков източник на светлина

- ***Point light source***
  - най-простият модел
    на източник на светлина
- Задават се
  - ***позиция на източника***
  - ***цвят на светлината*** (RGB стойности)



- Светлината се разпространява във всички посоки
- Полезен модел при малки източници на светлина

Точков източник на светлина

- ***Намаляване на радиалния интензитет***

  Radial Intensity Attenuation

  - с движението на светлината от източника интензитета й намалява

- На разстояние d_l от източника на светлина интензитета отслабва с коефициент $1/d_l^2$
  - не се получават добри резултати с този коефициент
  - използва се друг подход за намаляване на интензитета на светлината

Точков източник на светлина

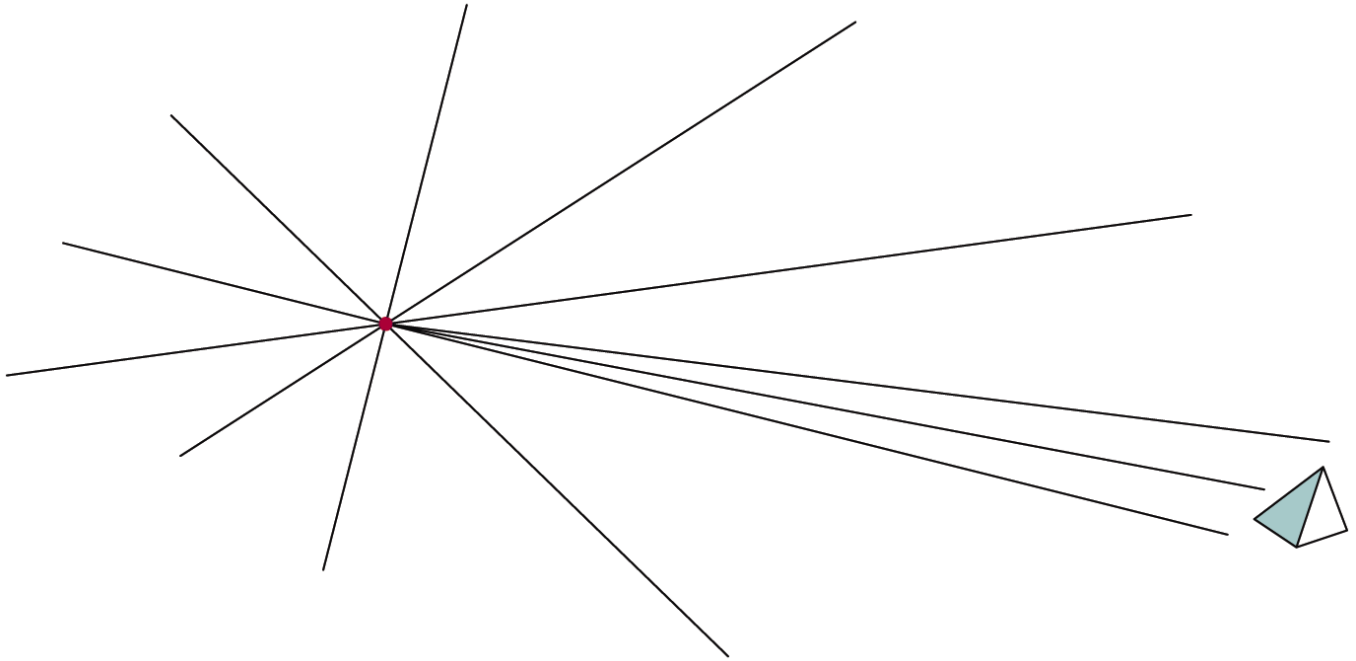
- ***Намаляване на радиалния интензитет***
  - използва се инверсна квадратична функция по отношение на интензитета

$$f_{radatten}(d_l) = \frac{1}{a_0 + a_1 d_l + a_2 d_l^{\,2}}$$

където с вариране на коефициентите a_0, a_1 и a_2 се променя резултата за постигане на желан ефект

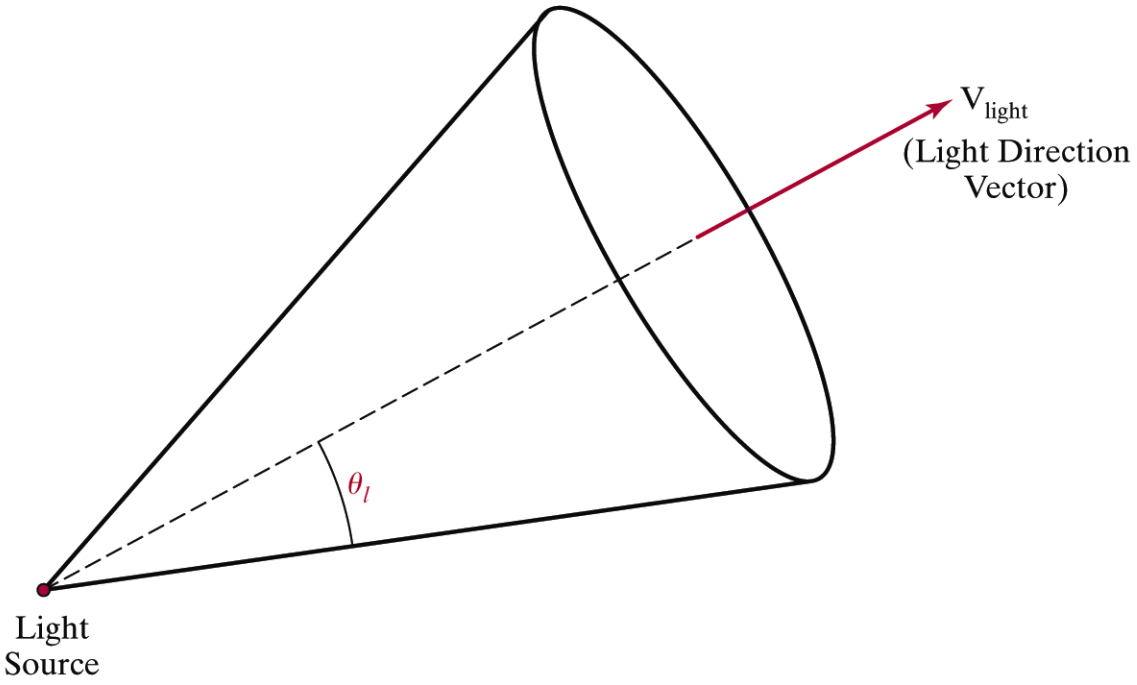
Безкрайно отдалечен източник

- Отдалечен източник на светлина може да бъде моделиран като точков източник
  - например Слънцето
- В такъв случай би имал много малък ефект ако се използва намаляване на радиалния интензитет



Насочен източник на светлина

- ***Spotlight***
- За да се преобразува точковия източник в насочен се добавя ***вектор на посока*** и ***ъглово ограничение*** θ_l

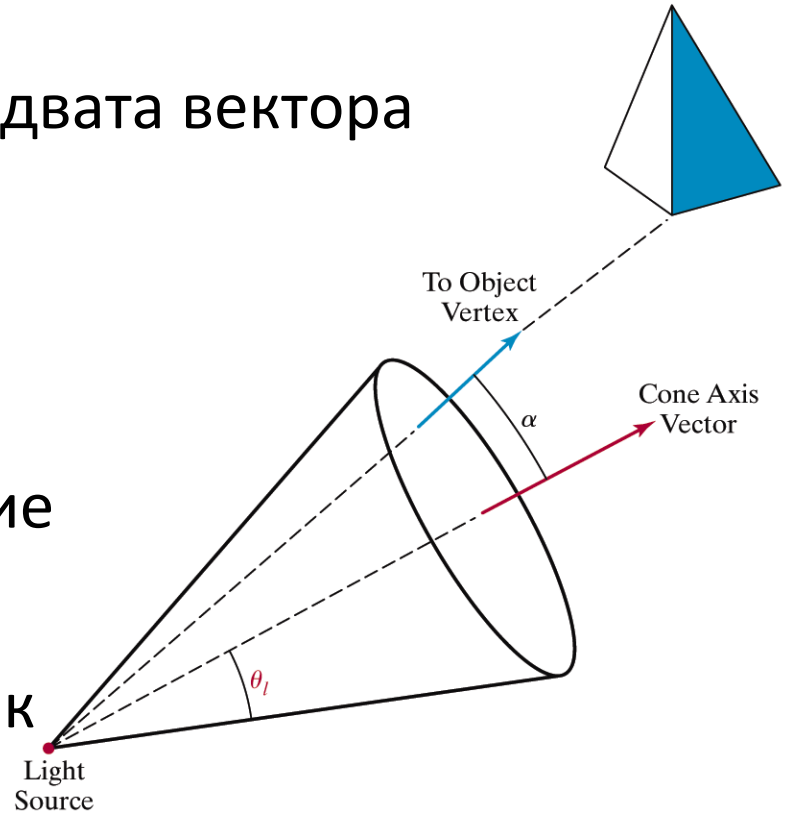


Насочен източник на светлина

- V_{light} – единичен вектор в посоката на светлината
- V_{obj} – единичен вектор в посока от източника на светлина към обекта
- Скаларното произведение на двата вектора определя ъгъла между тях

$$V_{obj} \cdot V_{light} = \cos \alpha$$

- Ако α е в ъгловото ограничение на светлината, то обектът е осветен от насочения източник



Насочен източник на светлина

- ***Намаляване на ъгловия интензитет***

  Angular Intensity Attenuation

- Освен намаляването на интензитета с отдалечаване от източника, интензитетът намалява и ъглово
  - използва се функция в зависимост от ъгъла

$$f_{angatten}(\phi) = \cos^{a_l} \phi \qquad 0^\circ \leq \phi \leq \theta$$

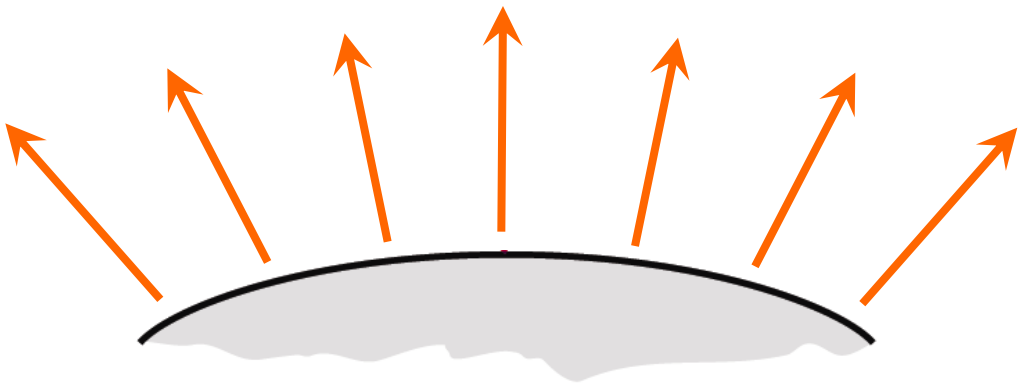
където за експонентата на отслабването a_l се задава положителна стойност и ъгъл ϕ се измерва от абцисата на конуса

Ефекти при осветяване

- Светлината отразена от повърхност зависи от материала на повърхността
  - ***лъскавите материали отразяват*** част от падащата светлина
    - ***reflection***
  - ***матовите материали абсорбират*** част от падащата светлина
    - ***absorption***
  - ***прозрачните повърхности пропускат*** част от падащата светлина
    - ***transmission***
      - част от светлината преминава през материала

Дифузно отразяване

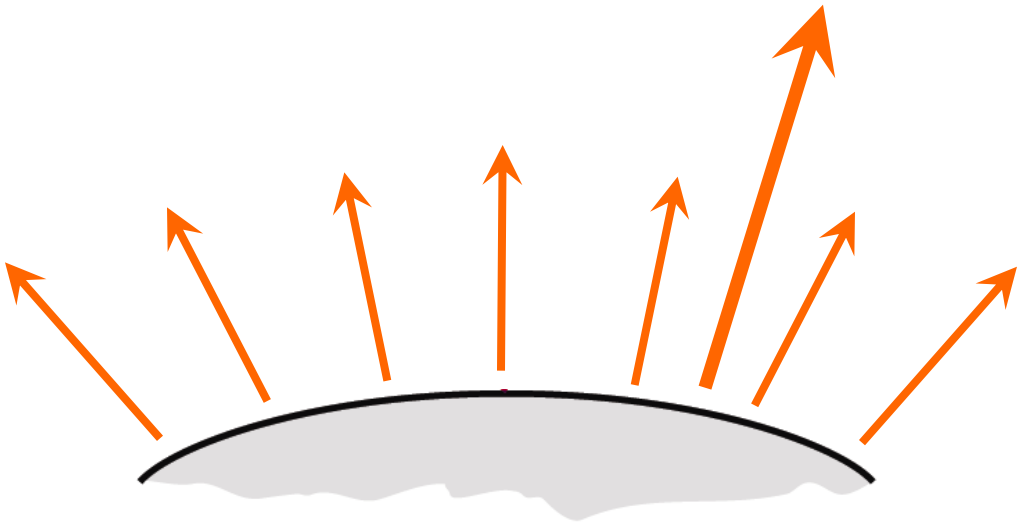
- Повърхностите, които са неравни и грапави отразяват светлината във всички посоки
  - ***дифузно отразяване*** **(*diffuse reflection*)**
    - например хартия, дърво



- Тъй като светлината се отразява еднакво във всички посоки, то повърхността изглежда еднакво от всички позиции на наблюдение
  - “view independent”

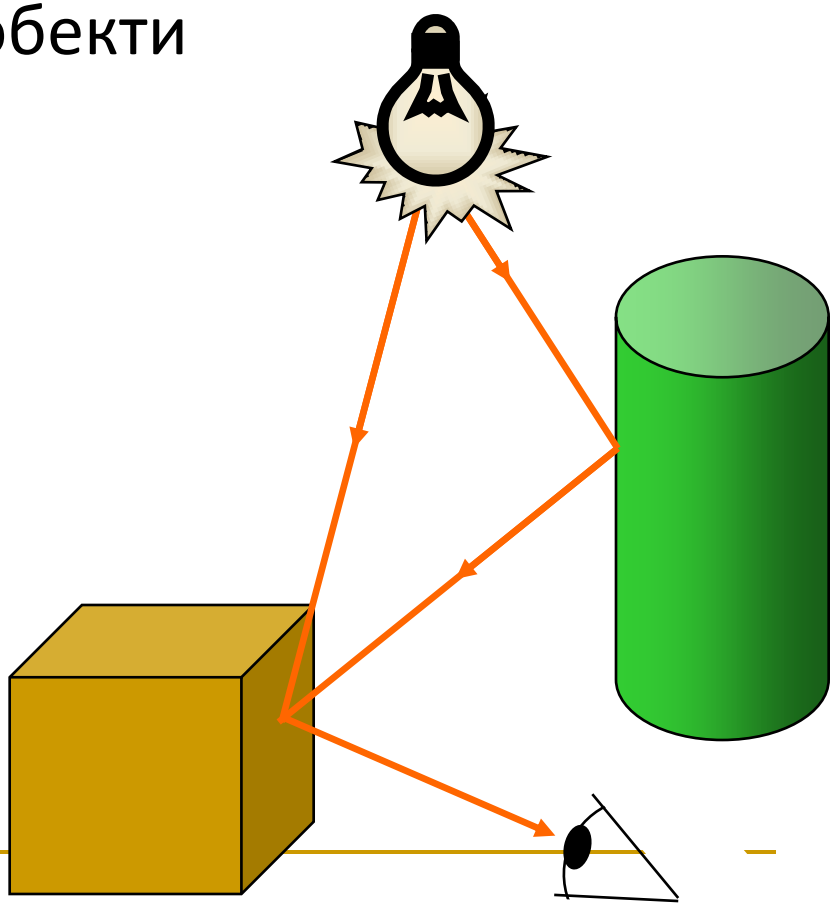
Огледално отразяване

- За някои повърхности освен дифузното отразяване е характерно концентриране на отразената светлина в определена осветена точка или светло петно
  - ***огледално отразяване*** **(*specular reflection*)**



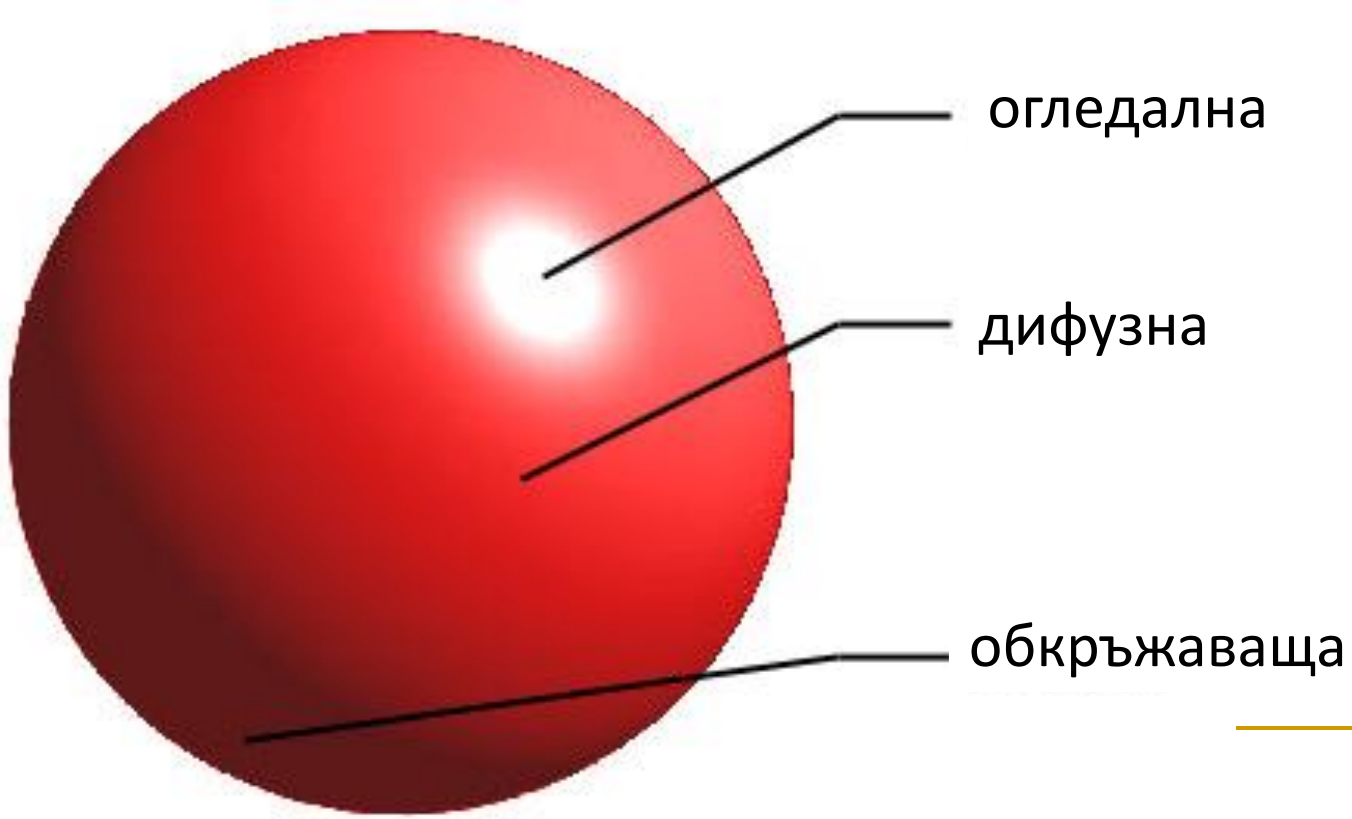
Обкръжаваща светлина

- Повърхност, която не е изложена на директна светлина може да бъде осветена от отразена светлина от други близки обекти
  - ***обкръжаваща светлина*** (***ambient light***)



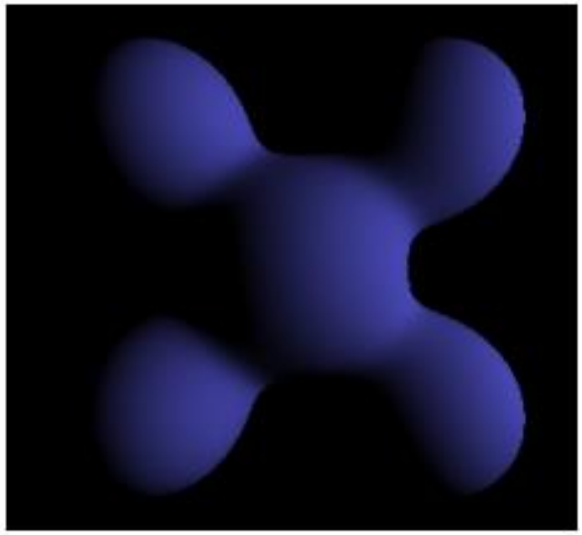
Пример

- Общата отразена светлина от дадена повърхност е сумата на всички директни източници на светлина и обкръжаващата светлина



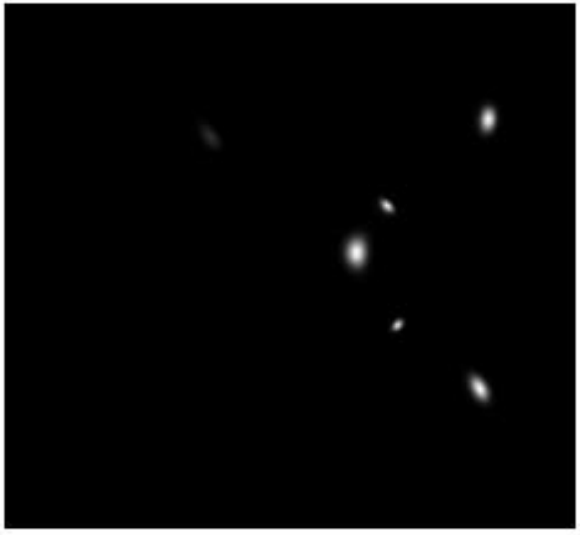
Пример

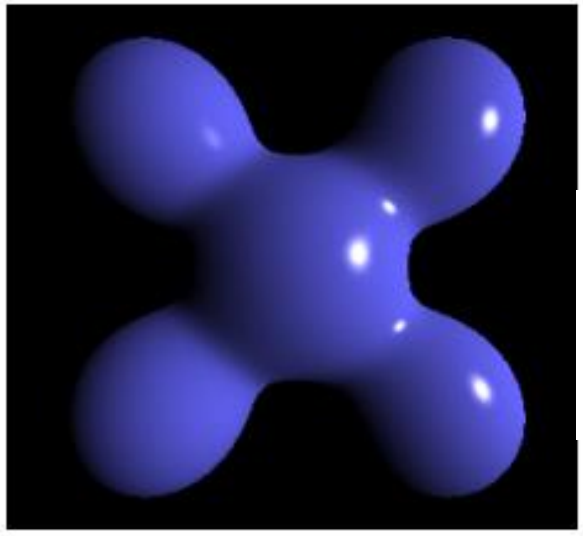
обкръжаваща



дифузна

огледална





крайно изображение

Основен модел на осветеност

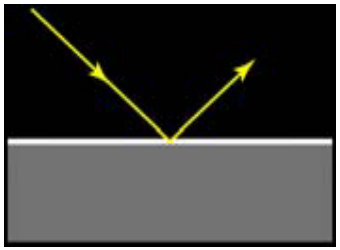
- Базов модел на осветеност
  - дава сравнително добри резултати
  - използва се в повечето графични системи

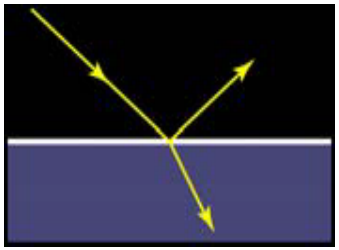
- Компоненти на модела
  - ***Обкръжаваща светлина*** (***Ambient light***)
  - ***Дифузно отразяване*** (***Diffuse reflection***)
  - ***Огледално отразяване*** (***Specular reflection***)

- Приема се, че светлината е монохромна

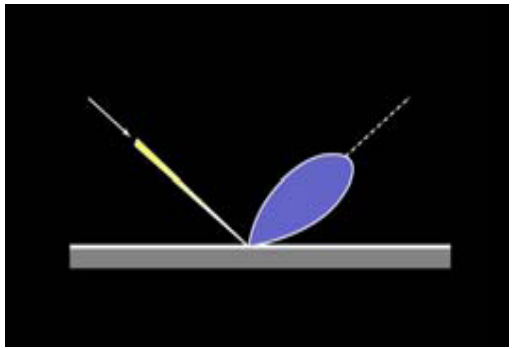
Модел на отразяване

идеално огледално отразяване (модел на Fresnel)

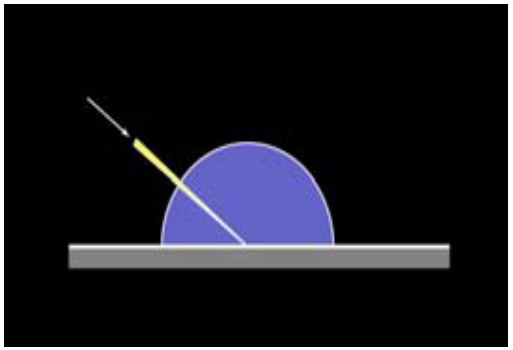




дифузно огледално отразяване



Ламбертово дифузно отразяване



Обкръжаваща светлина

- За да се включи в модела фоновата обкръжаваща светлина се задава ***общо ниво на осветеност*** за сцената
  - апроксимира се ***общото дифузно отразяване*** от различни повърхности в сцената
  - означава се с I_a

Дифузно отразяване

- Допуска се, че повърхностите отразяват падащата светлина с еднакъв интензитет във всички посоки
  - ***идеални дифузни отражателни повърхности*** или ***Ламбертови отражателни повърхности*** (***ideal diffuse reflectors*** или ***Lambertian reflectors***)
    - например стените около нас

Дифузно отразяване

- ***Коефициент на дифузно отразяване***

  diffuse-reflection coefficient или ***diffuse reflectivity***

  - задава за всяка повърхност
  - означава се с k_d
  - определя каква част от падащата върху повърхността светлина се разсейва като дифузно отразяване
- Стойността на k_d е между 0.0 и 1.0
  - при $k_d = 0.0$
    - матова повърхност, която поглъща почти цялата светлина
  - при $k_d = 1.0$
    - лъскава повърхност, която отразява почти цялата светлина

Дифузно отразяване

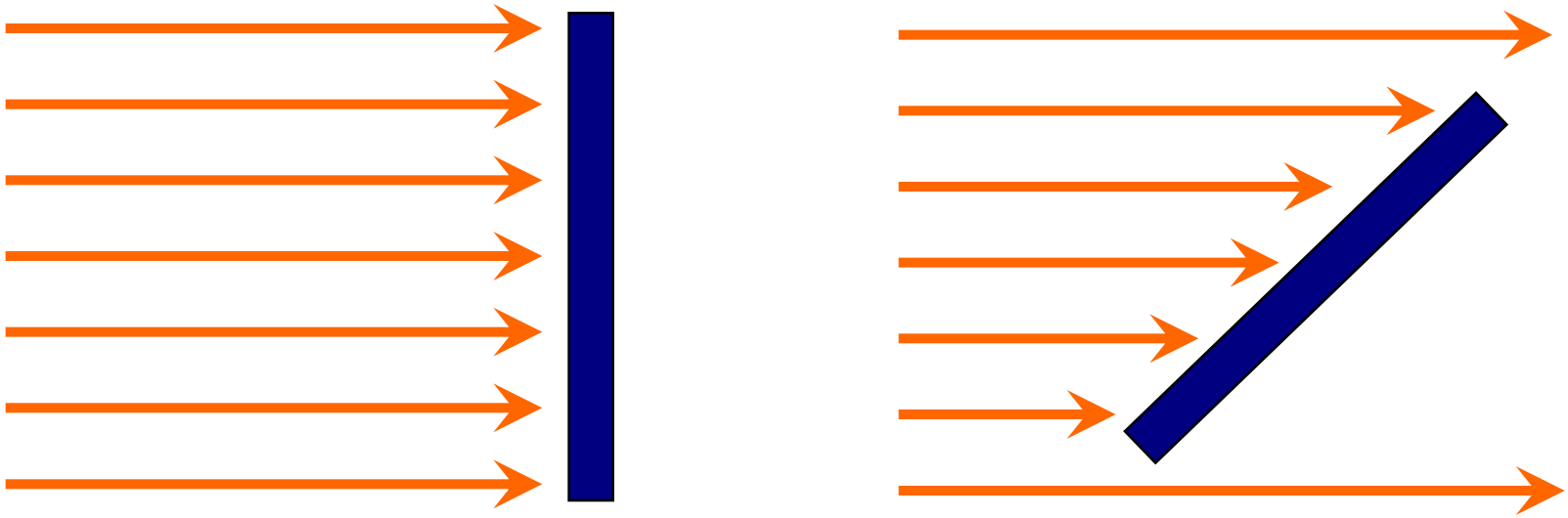
- ***Дифузно отразяване и обкръжаваща светлина***
  - за определяне на фоновото обкръжаващо осветяване може да се допусне, че всяка повърхност е напълно осветена от обкръжаващата светлина на сцената I_a
- Участие на обкръжаващата светлина в дифузното отразяване

$$I_{ambdiff} = k_d I_a$$

- Сама за себе си фоновата обкръжаваща светлина не е интересна като визуален ефект, затова се използват и други източници на светлина в сцената

Дифузно отразяване

- Когато повърхност се освети от източник на светлина, количеството попадаща светлина зависи от ***ориентацията на повърхността*** спрямо посоката на източника на светлина

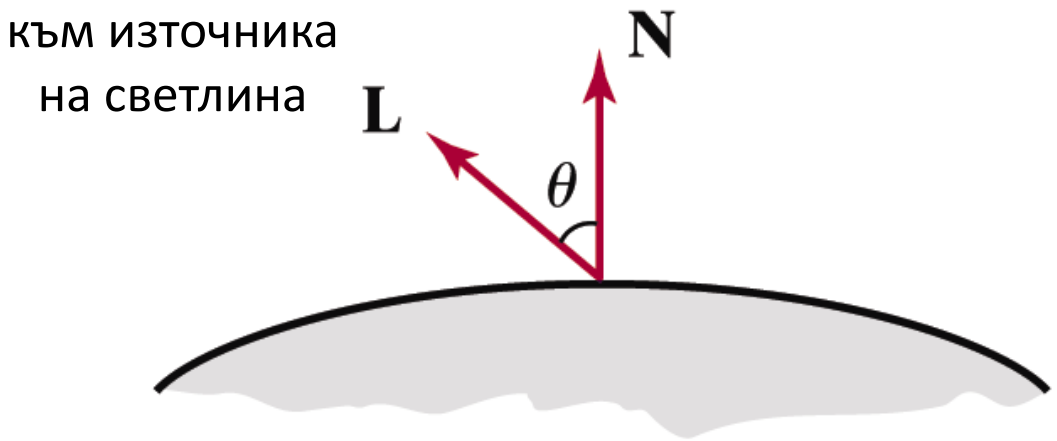


Дифузно отразяване

- ***Ъгъл на падащата светлина ϑ***

  angle of incidence

  - ъгълът между посоката на идващата светлина и нормалата на повърхността



Дифузно отразяване

- Количеството попадаща върху повърхността светлина е

$$I_{l,incident} = I_l \cos\theta$$

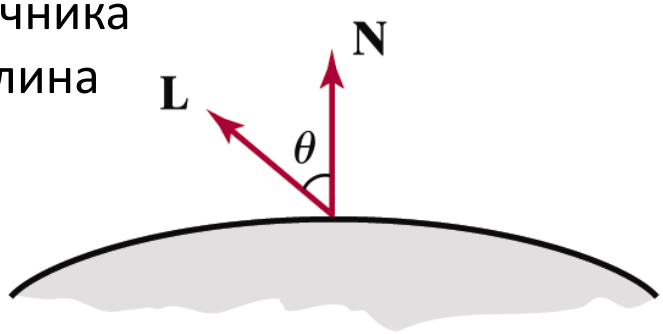
- Дифузното отразяване може да се моделира като

$$\begin{aligned} I_{l,diff} &= k_d I_{l,incident} \\ &= k_d I_l \cos\theta \end{aligned}$$

Дифузно отразяване

- **N** – нормала към повърхността
- **L** – единичен вектор по посока на източника на светлина

към източника
на светлина



$$N \cdot L = \cos\theta$$

- ***Ламбертов BRDF***

$$I_{l,diff} = \begin{cases} k_d I_l (N \cdot L) & \text{ако } N \cdot L > 0 \\ 0 & \text{ако } N \cdot L \leq 0 \end{cases}$$

Дифузно отразяване

- ***Комбиниране на дифузното отразяване, обкръжаващата и директно падащата светлина***
  - обикновено се използват два отделни коефициента за дифузно отразяване
    - k_a за обкръжаващата светлина
    - k_d за директно падащата светлина
- ***Уравнение на общото дифузно отразяване при един източник на светлина***

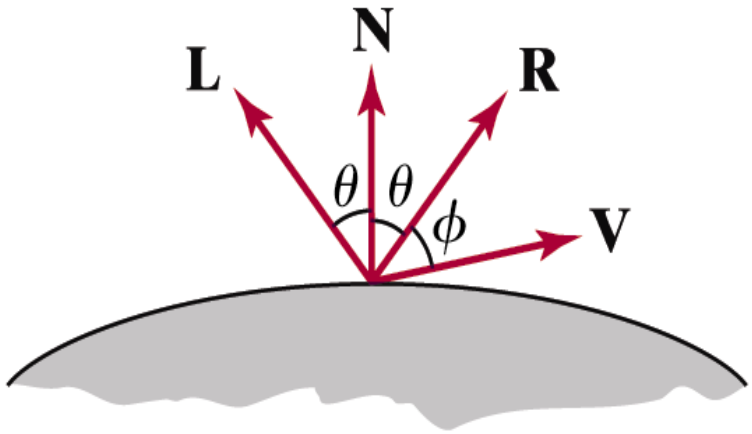
$$I_{diff} = \begin{cases} k_a I_a + k_d I_l (N \cdot L) & \text{ако } N \cdot L > 0 \\ k_a I_a & \text{ако } N \cdot L \leq 0 \end{cases}$$

Огледално отразяване

- ***Ъгъл на огледално отразяване***

  specular reflection angle

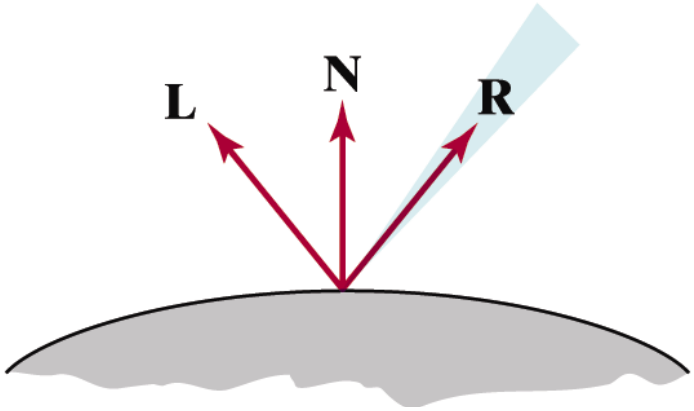
  - интензитетът на отразената светлина зависи от посоката на наблюдение
  - резултат от почти пълното отразяване на падащата светлина в концентриран регион около ъгъла на огледално отразяване



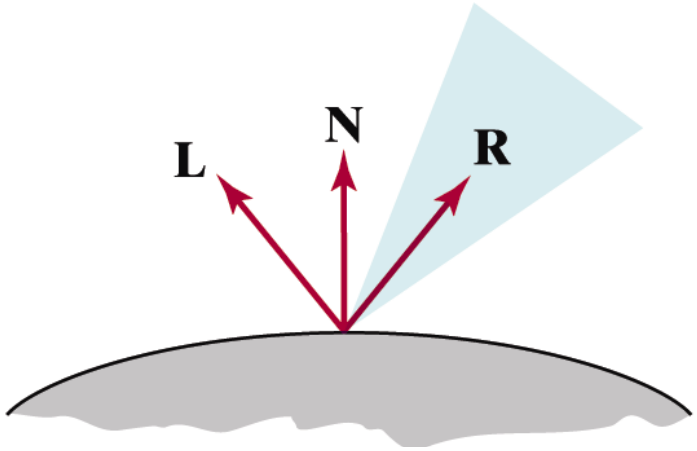
  - светло петно, което се вижда върху лъскава и гладка повърхност
  - ъгълът на огледално отразяване е равен на ъгъла на падащата светлина

Огледално отразяване

- ***Идеално огледало***
  - отразява само в посоката на огледално отразяване
    - максимална стойност на BRDF в тази посока
- Други обекти имат огледално отражение в определен краен обхват от позиции на наблюдение около вектор *R*



гладка и лъскава повърхност



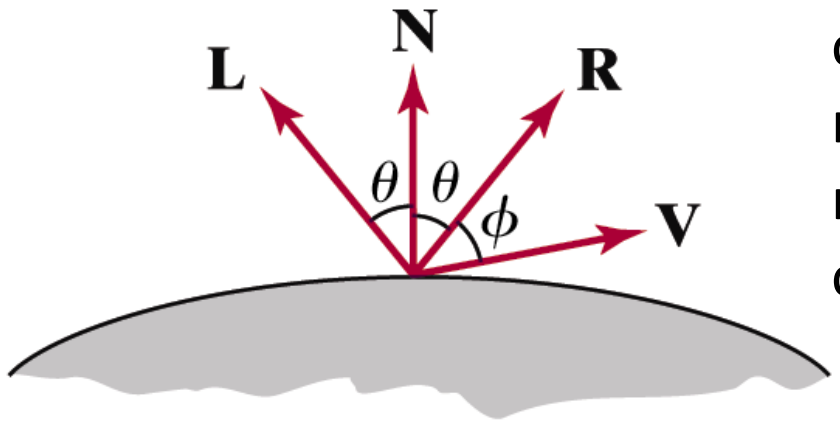
груба и грапава повърхност

Огледално отразяване

- ***Модел на Фонг за огледално отразяване***

  Phong Specular Reflection Model

  - емпиричен модел за изчисляване на обхвата на огледално отразяване
  - създаден през 1973 от Phong Bui Tuong



интензитетът на огледално отразяване е пропорционален на ъгъла между вектора на наблюдение **V** и вектора на огледално отразяване **R**

Огледално отразяване

- ***Модел на Фонг за огледално отразяване***
  - огледалното отразяване е пропорционално на

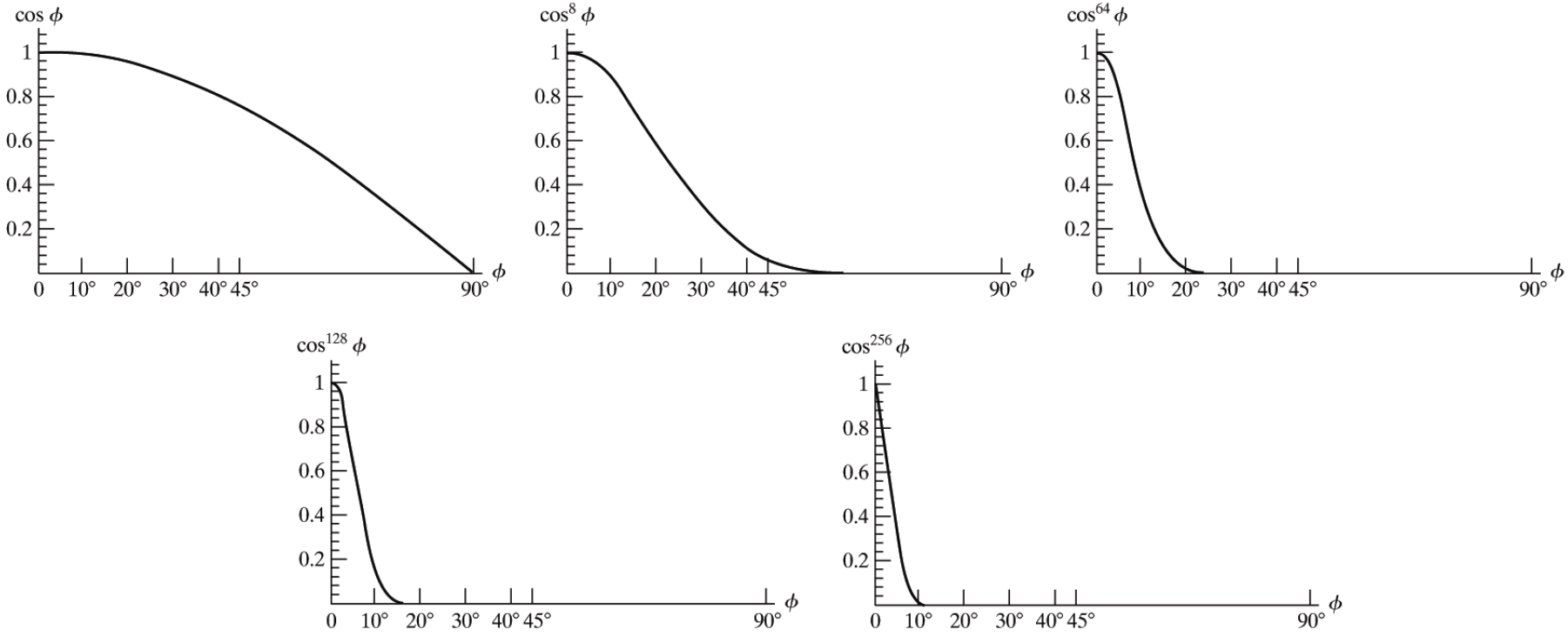
$$\cos^{n_s} \phi$$

    - ъгълът ϕ се изменя между 0° и 90°,

      при което $\cos\phi$ варира от 1.0 до 0.0
  - ***степен на огледално отразяване n_s***
    - определя се от типа на повърхността, която се визуализира
    - за гладки и лъскави повърхности стойността е голяма (>100)
    - за груби и грапави повърхности стойността е близка до 1

Огледално отразяване

- Зависимост между степента на огледално отразяване n_s и ъгловия обхват на огледално отразяване



Огледално отразяване

- За някои материали количеството огледално отразяване зависи силно от ъгъла на падащата светлина

- ***Закон на отразяването на Fresnel***
  - описва поведението на огледалното отразяване
  - http://hyperphysics.phy-astr.gsu.edu/hbase/phyopt/freseq.html

- В компютърната графика обикновено ефектът на огледално отразяване се апроксимира с константен коефициент на огледално отразяване k_s

Огледално отразяване

- ***Интензитет на огледално отразяване***

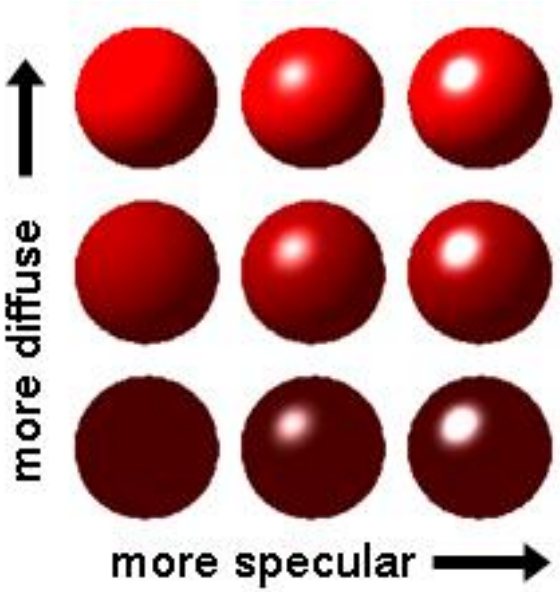
$$I_{l,spec} = k_s I_l \cos^{n_s} \phi$$

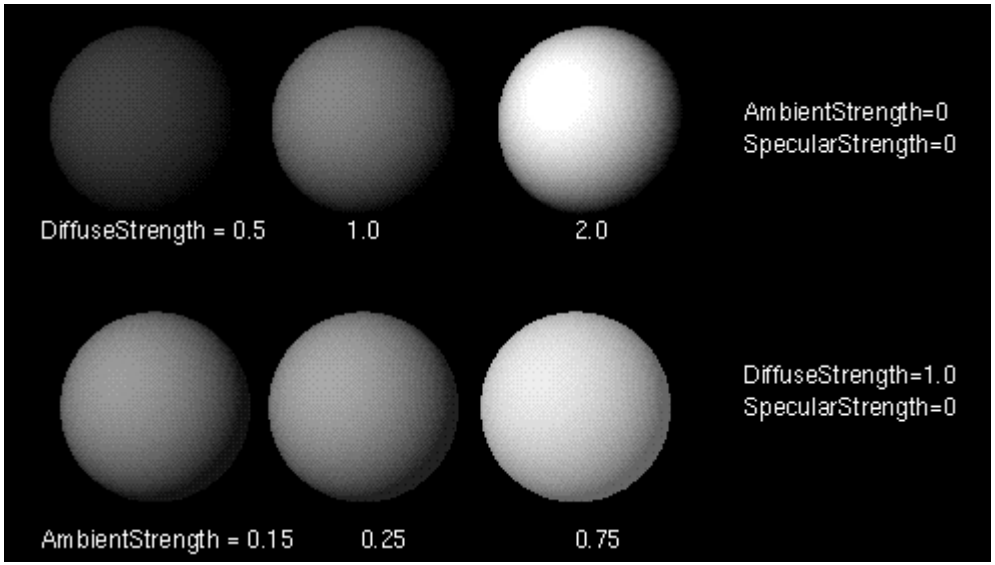
- При това $V \cdot R = \cos\phi$

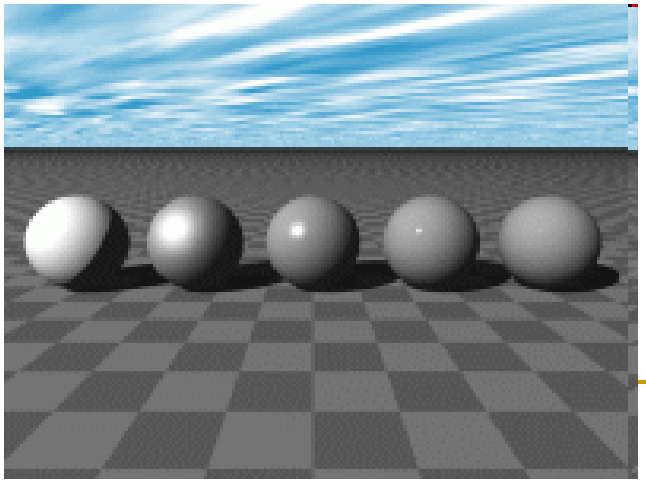
- Следователно

$$I_{l,spec} = \begin{cases} k_s I_l (V \cdot R)^{n_s} & \text{ако } V \cdot R > 0 \text{ и } N \cdot L > 0 \\ 0.0 & \text{ако } V \cdot R < 0 \text{ или } N \cdot L \leq 0 \end{cases}$$

Примери

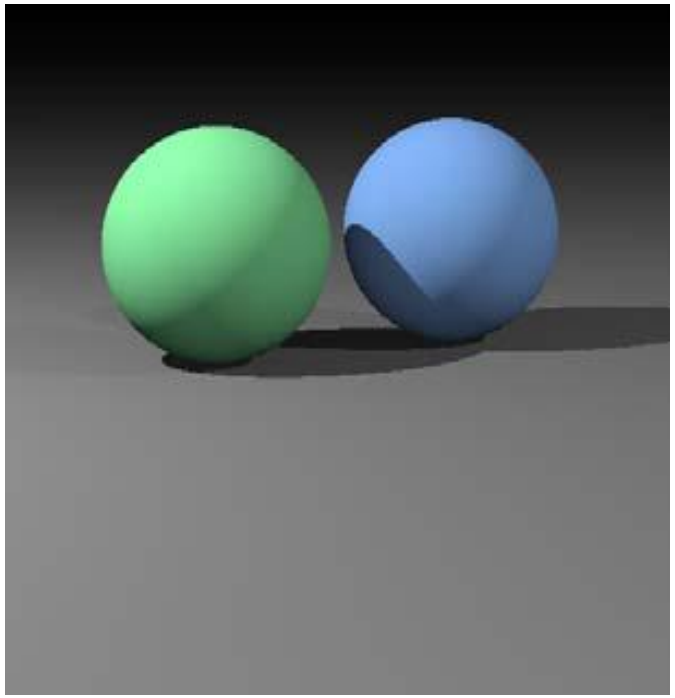

more diffuse
more specular


AmbientStrength=0
SpecularStrength=0
DiffuseStrength = 0.5
1.0
2.0
DiffuseStrength=1.0
SpecularStrength=0
AmbientStrength = 0.15
0.25
0.75

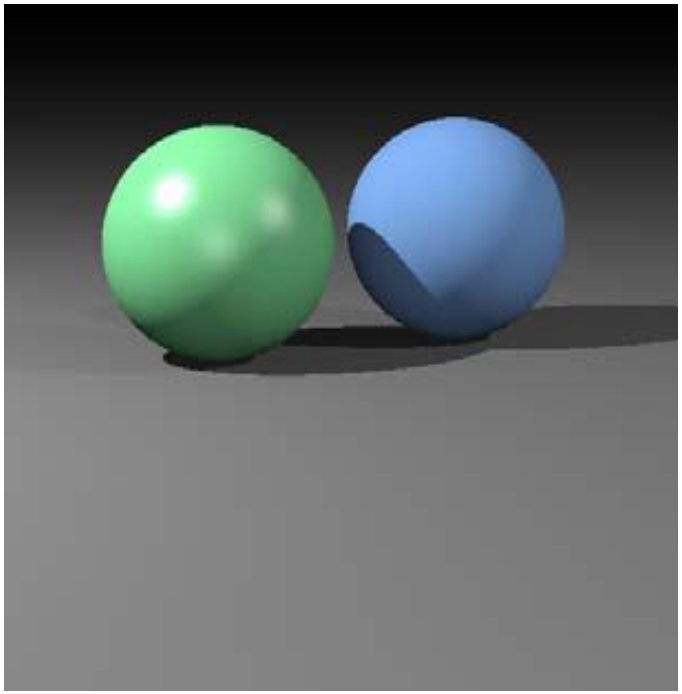


Примери

- Дифузно отразяване
  - ефект на матова повърхност



- Дифузно + огледално отразяване



Дифузно и огледално отразяване

- ***Комбиниране на дифузно и огледално отразяване при единствен източник на светлина***

$$I = I_{diff} + I_{spec}$$

$$= k_a I_a + k_d I_l (N \cdot L) + k_s I_l (V \cdot R)^{n_s}$$

Дифузно и огледално отразяване

- ***Комбиниране на дифузно и огледално отразяване при множество източници на светлина***
  - в сцената може да има произволен брой източници на светлина
  - определя се ***сумарно*** дифузно и огледално отразяване от всички източници на светлина

$$I = I_{ambdiff} + \sum_{l=1}^{n} \left[I_{l,diff} + I_{l,spec} \right]$$

$$= k_a I_a + \sum_{l=1}^{n} I_l \left[k_d \left(N \cdot L \right) + k_s \left(V \cdot R \right)^{n_s} \right]$$

Общ модел на осветеност

- Отчитане в модела на осветеност на ***радиалното и ъглово намаляване на интензитета***

$$I = I_{ambdiff} + \sum_{l=1}^{n} \left[f_{l,radatten} f_{l,angatten} \left(I_{l,diff} + I_{l,spec} \right) \right]$$

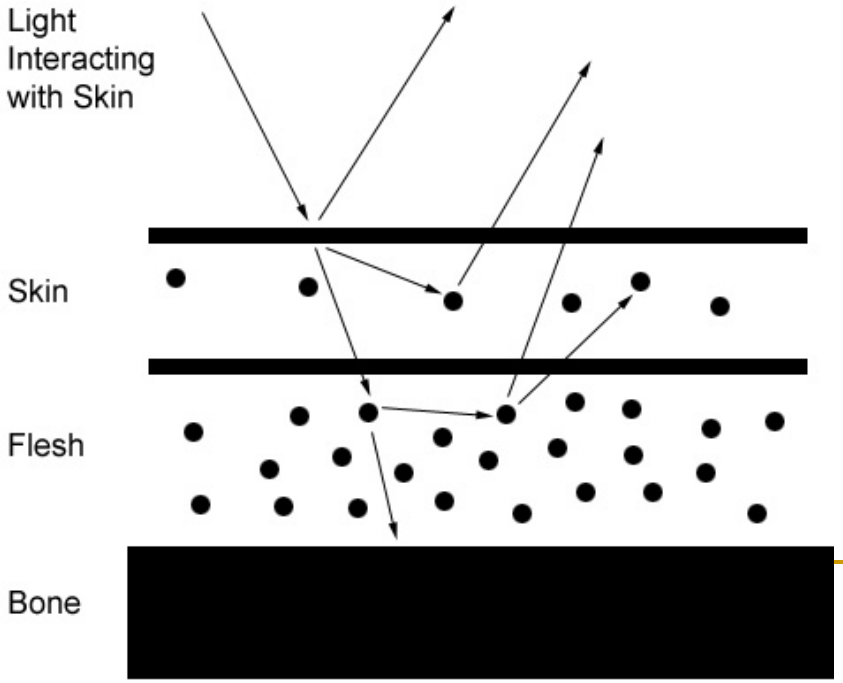
където $f_{radatten}$ и $f_{angatten}$ са функциите, отчитащи съответно радиалното и ъгловото намаляване на интензитета

BRDF

- BRDF моделира отразяването когато светлината директно стига до повърхността
- Обобщени BRDF
  - отчитат факта, че светлината може да се отразява/абсорбира при движение през някаква среда или обект (напр. мъгла, вода)
- BSSRDF (Bidirectional Scattering Surface Reflectance Distribution Function)
  - дефинират се допълнителни членове за да се отчете пречупването и излъчването от повърхностите
- За някои материали чрез специфични устройства се определят таблици с данни за BRDF

BSSRDF

- Някои повърхности могат да отразяват светлината вътрешно преди тя да се отрази обратно
  - ***subsurface scattering***
    - кожа, восък, коса, мляко и други течности
- Тези взаимодействия се отчитат с по-сложни модели



RGB цветове

- За цветно представяне в RGB цветово пространство за всеки от интензитетите се задава три елементен вектор
  - за всеки източник на светлина

$$I_l = (I_{lR}, I_{lG}, I_{lB})$$

- Всички параметри се задават по подобен начин като вектори

$$k_a = (k_{aR}, k_{aG}, k_{aB}) \qquad k_d = (k_{dR}, k_{dG}, k_{dB})$$

$$k_s = (k_{sR}, k_{sG}, k_{sB})$$

RGB цветове

- Всяка компонента на цвета на повърхността се изчислява с отделен израз
  - например

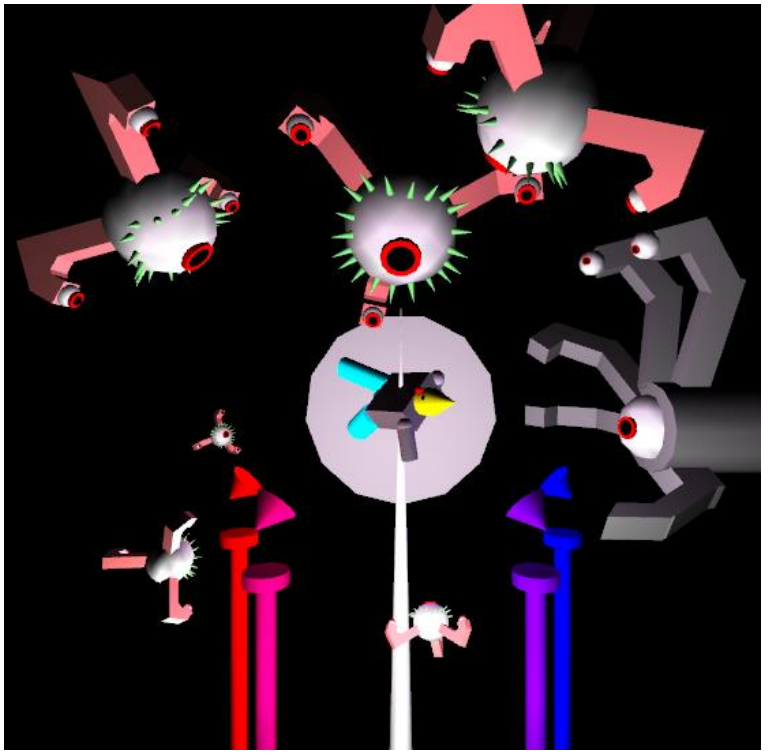
$$I_{lR,diff} = k_{dR} I_{lR} (N \cdot L)$$

$$I_{lG,diff} = k_{dG} I_{lG} (N \cdot L)$$

$$I_{lB,diff} = k_{dB} I_{lB} (N \cdot L)$$

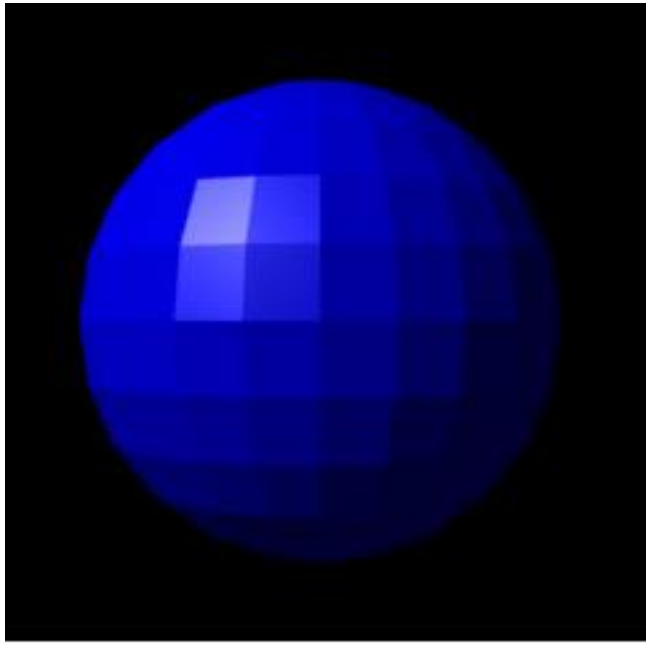
Модел за shading

- ***Shading***
  - определяне на цвета на точки чрез интерполиране между точките с известна осветеност
- ***Модели за shading***
  - плосък модел (flat)
  - модел на Гуро (Gouraud)
  - модел на Фонг (Phong)



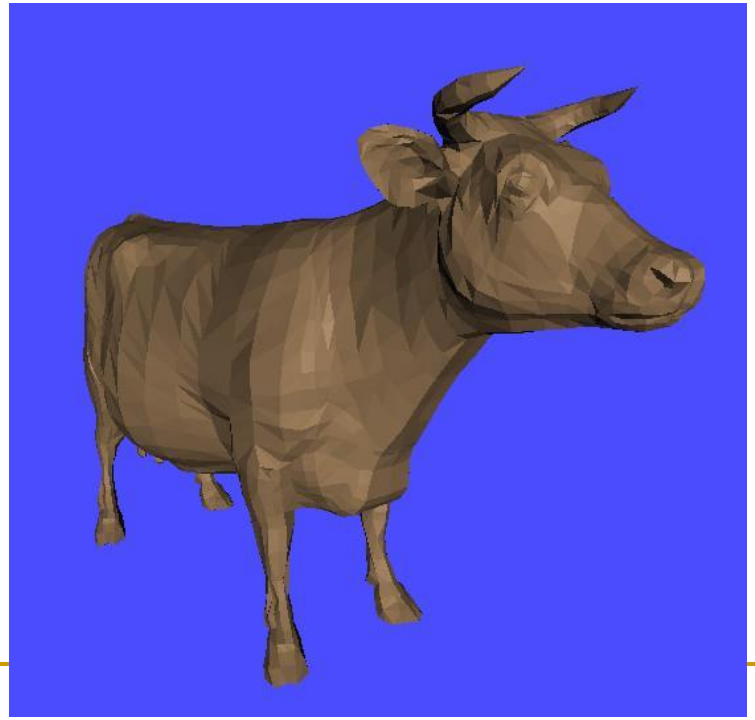
Модел за shading

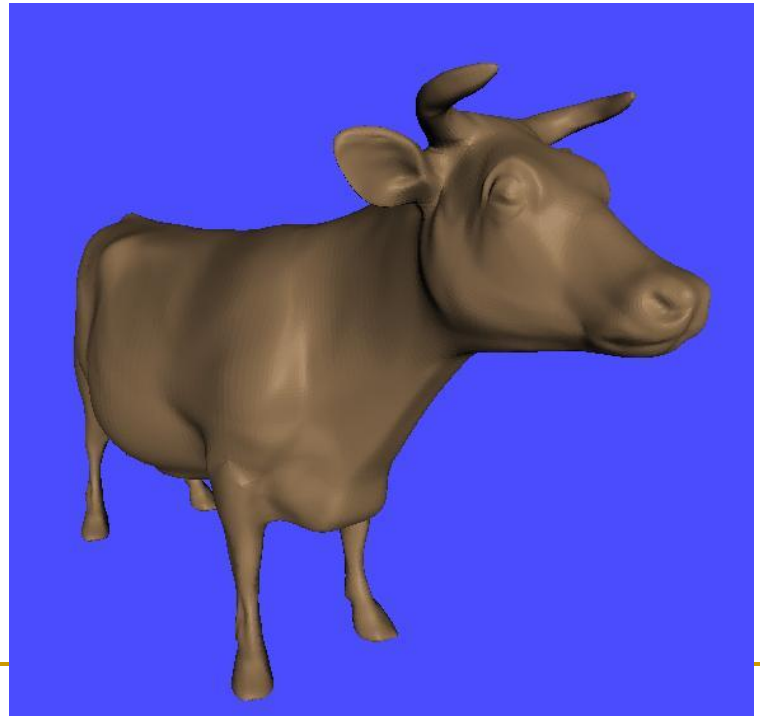
- ***Плосък (flat)***
  - най-простият метод за рендериране на полигони
  - дефинира се нормала за всеки полигон (не във възлите)
  - ***Lighting***
    - изчислява се BRDF в центъра на всеки полигон с използване на нормалата
  - ***Shading***
    - използва се изчислената стойност за всяка друга точка от този полигон
  - ***Предимства***
    - много бързо
  - ***Недостатъци***
    - може да бъде нереалистично



Модел за shading

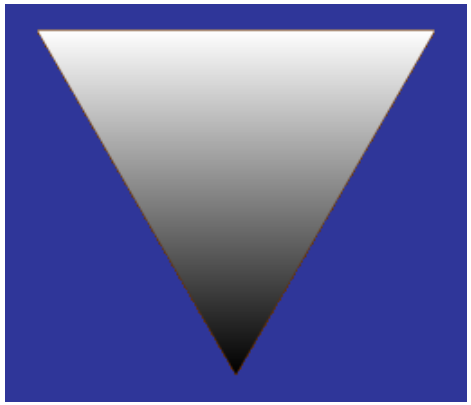
- ***Плосък (flat)***
  - за преодоляване на недостатъка на плоския модел се добавят нови полигон
  - но така изчисленията се увеличават и рендерирането става бавно





Модел за shading

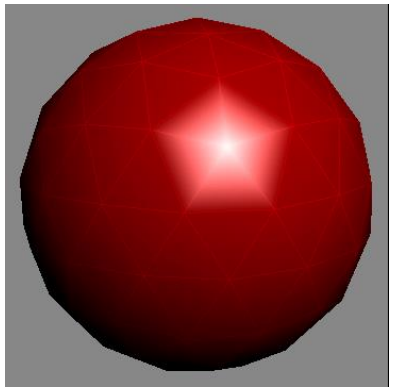
- ***Модел на Гуро***
  - предложен от Henri Gouraud през 1970
    - работи в Университета на Юта с Ivan Sutherland и David Evans
  - нарича се още ***intensity-interpolation surface rendering***
  - дефинира се нормала във всеки възел

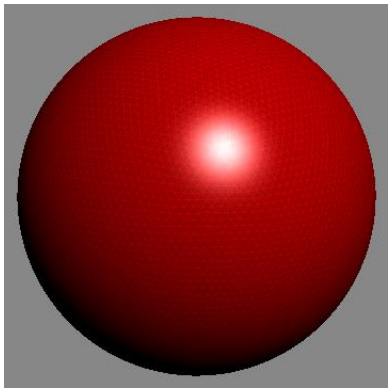


  - ***Lighting***
    - изчислява се BRDF във всеки възел
      с използване на съответната нормала
  - ***Shading***
    - цвета на всяка друга sampling точка в полигона се интерполира линейно чрез възлите с изчислена осветеност

Модел за shading

- ***Модел на Гуро***





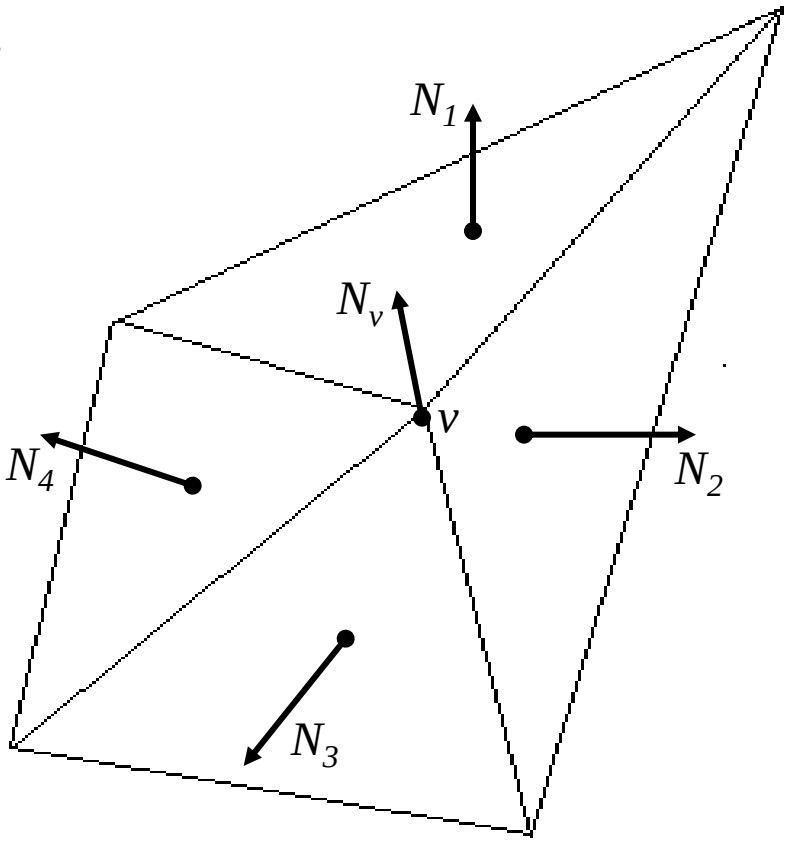
Модел за shading

- ***Модел на Гуро***
  - осреднен единичен нормален вектор във възел *v*

$$N_v = \frac{N_1 + N_2 + N_3 + N_4}{\left| N_1 + N_2 + N_3 + N_4 \right|}$$

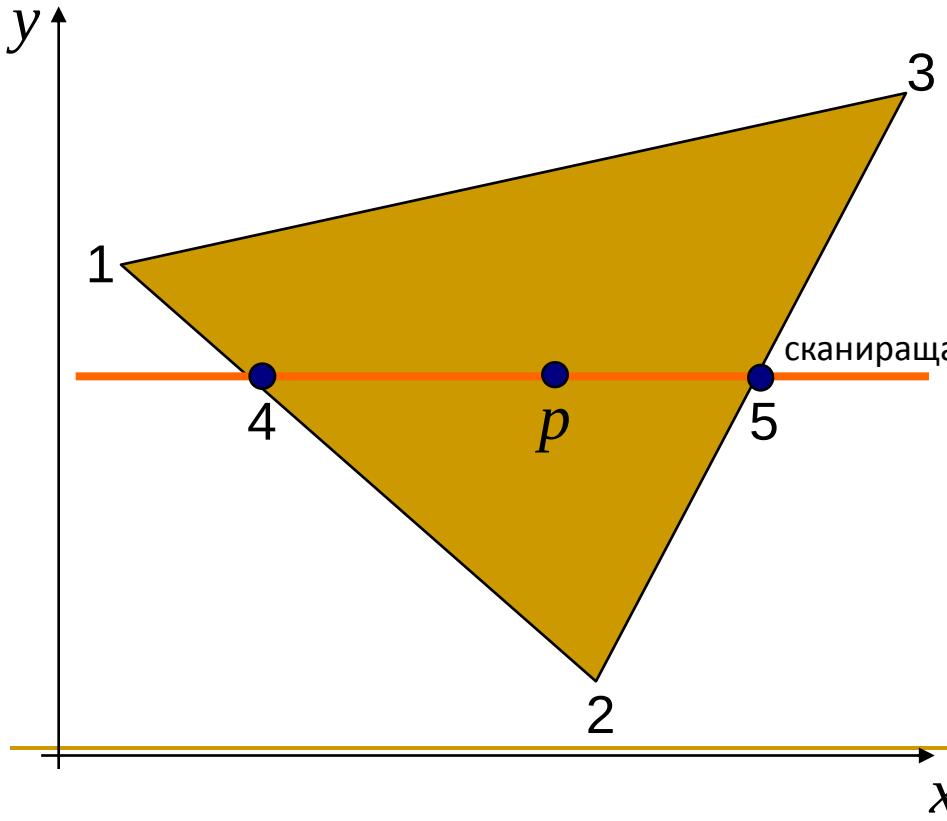
  - в общия случай

$$N_v = \frac{\sum_{i=1}^{n} N_i}{\left| \sum_{i=1}^{n} N_i \right|}$$



Модел за shading

- ***Модел на Гуро***
  - осветеността се интерполира линейно по всяка сканираща линия



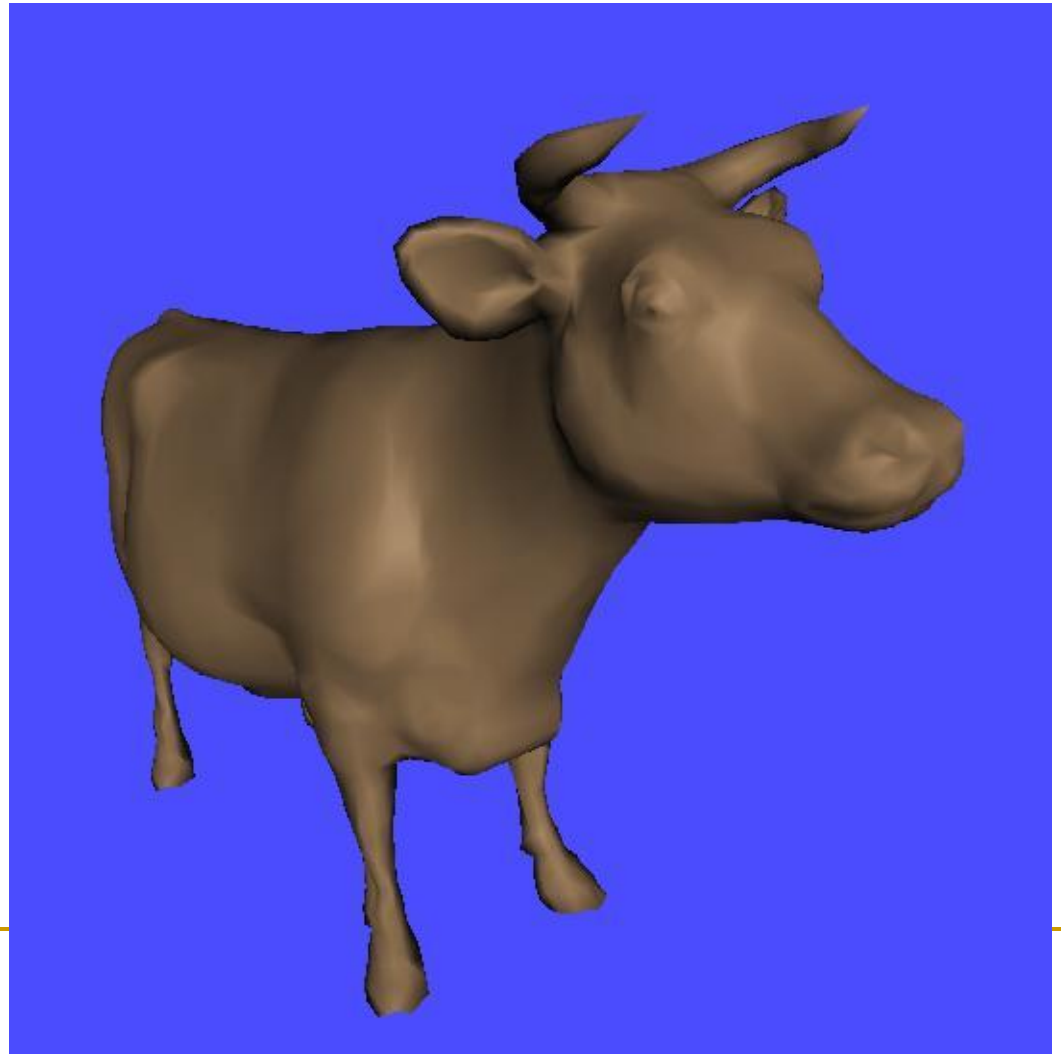
$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

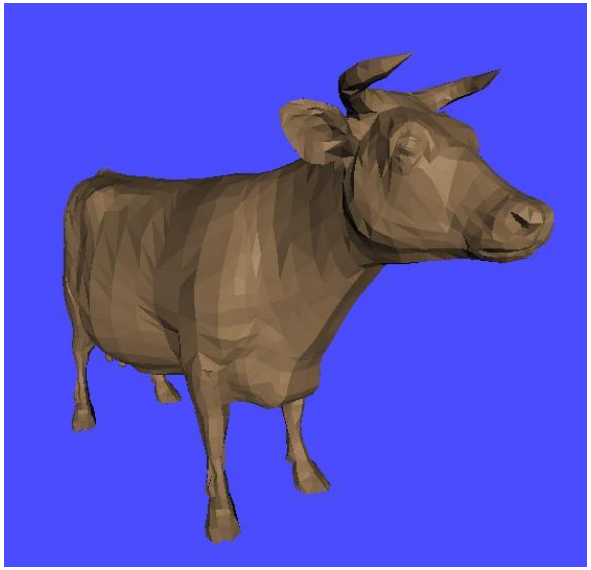
$$I_5 = \frac{y_5 - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_5}{y_3 - y_2} I_2$$

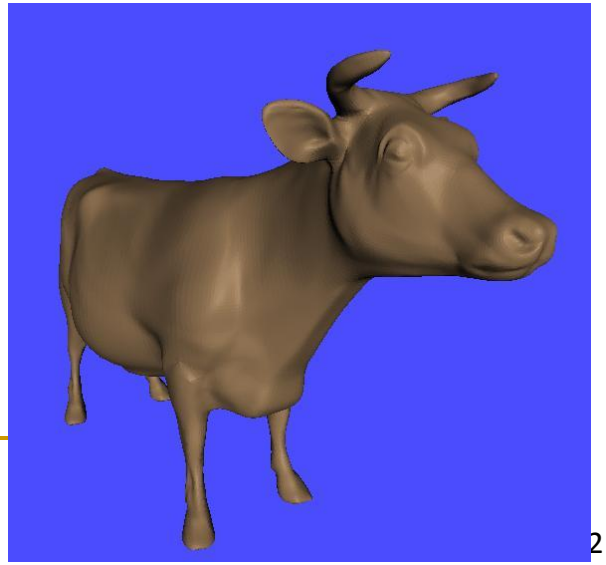
$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

Модел за shading

- ***Модел на Гуро***

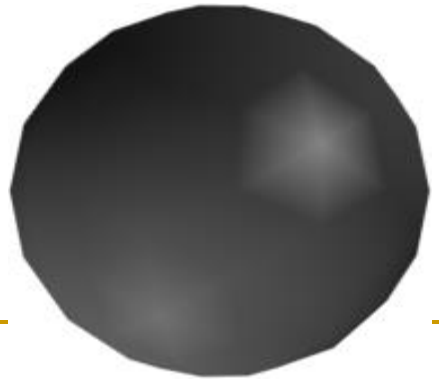




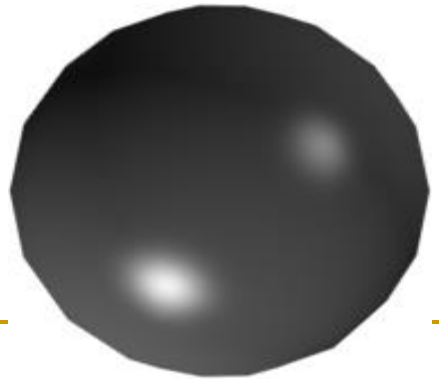


Модел за shading

- ***Модел на Гуро***
  - може да се използва ефективна итеративна имплементация
  - обикновено се прилага като част от етапа на определяне на видими повърхнини
  - недостатъци
    - възможно е получаване на аномалии от вида Mach bands
    - не моделира удачно огледално отразяване



модел на Гуро



модел на Фонг

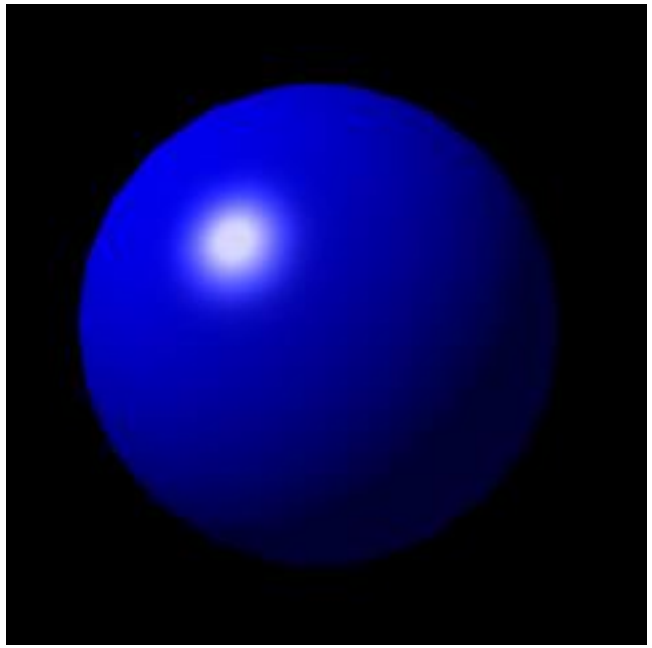
Mach bands

Модел за shading

- ***Модел на Фонг***
  - по-точен модел за shading чрез интерполиране от модела на Гуро
  - предложен от Phong Bui Tuong
  - нарича се още ***normal-vector interpolation rendering***
  - използва интерполиране *не* на стойностите на осветеността, а на векторите на нормалите

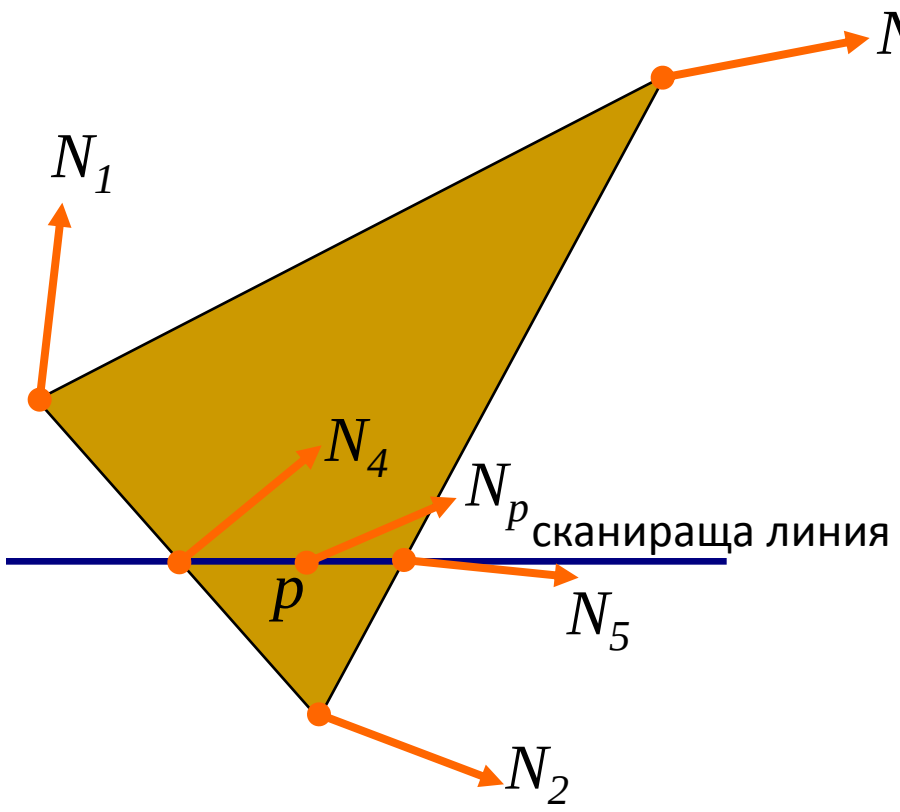
Модел за shading

- ***Модел на Фонг***
  - дефинира се нормала във всеки възел
  - ***Lighting***
    - изчислява се BRDF за всеки възел с използване на нормалата му
  - ***Shading***
    - за всяка друга sampling точка от този полигон се интерполират нормалите във всички възли на полигона и се използва BRDF за определяне на цвета на точката



Модел за shading

- ***Модел на Фонг***



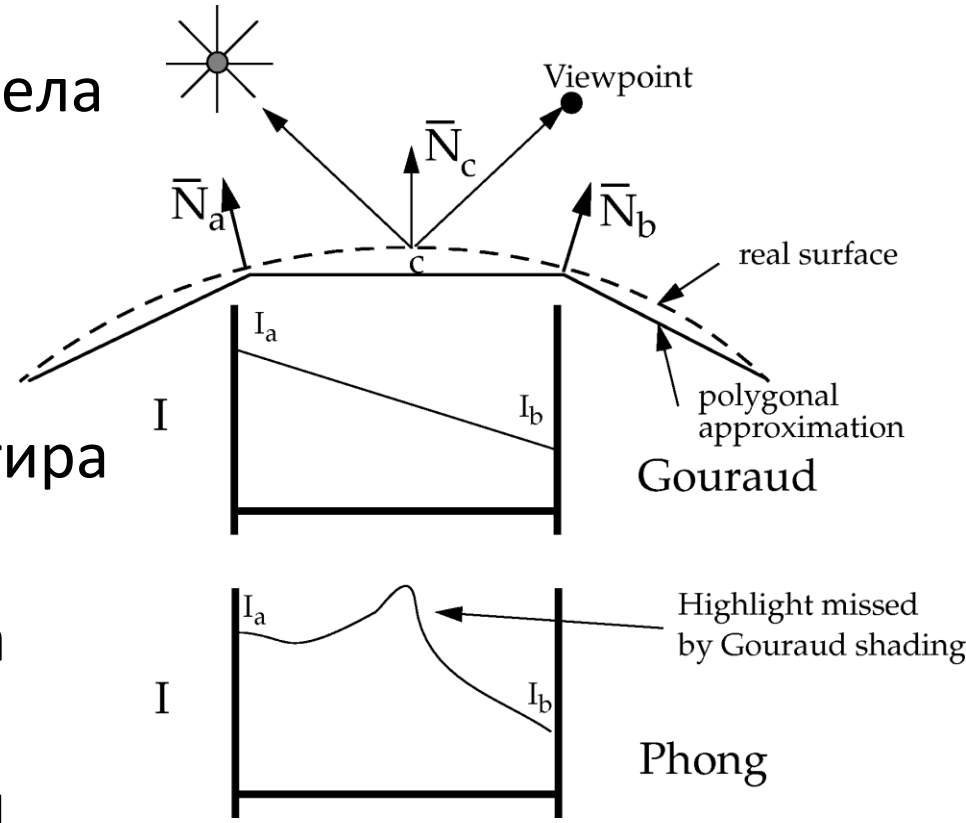
$$N_4 = \frac{y_4 - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y_4}{y_1 - y_2} N_2$$

$$N_5 = \frac{y_5 - y_2}{y_3 - y_2} N_3 + \frac{y_3 - y_5}{y_3 - y_2} N_2$$

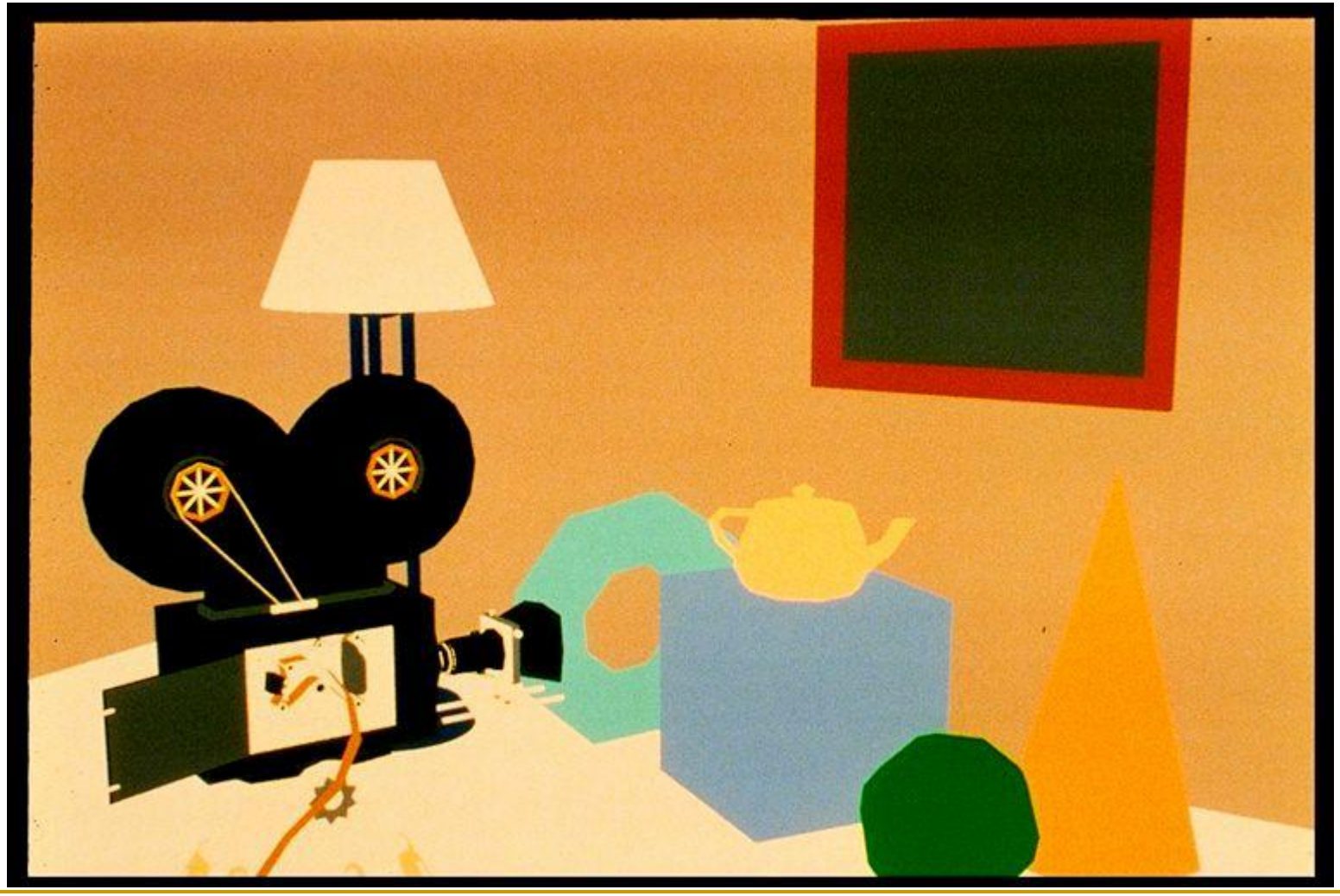
$$N_p = \frac{y_p - y_5}{y_4 - y_5} N_4 + \frac{y_4 - y_p}{y_4 - y_5} N_5$$

Модел за shading

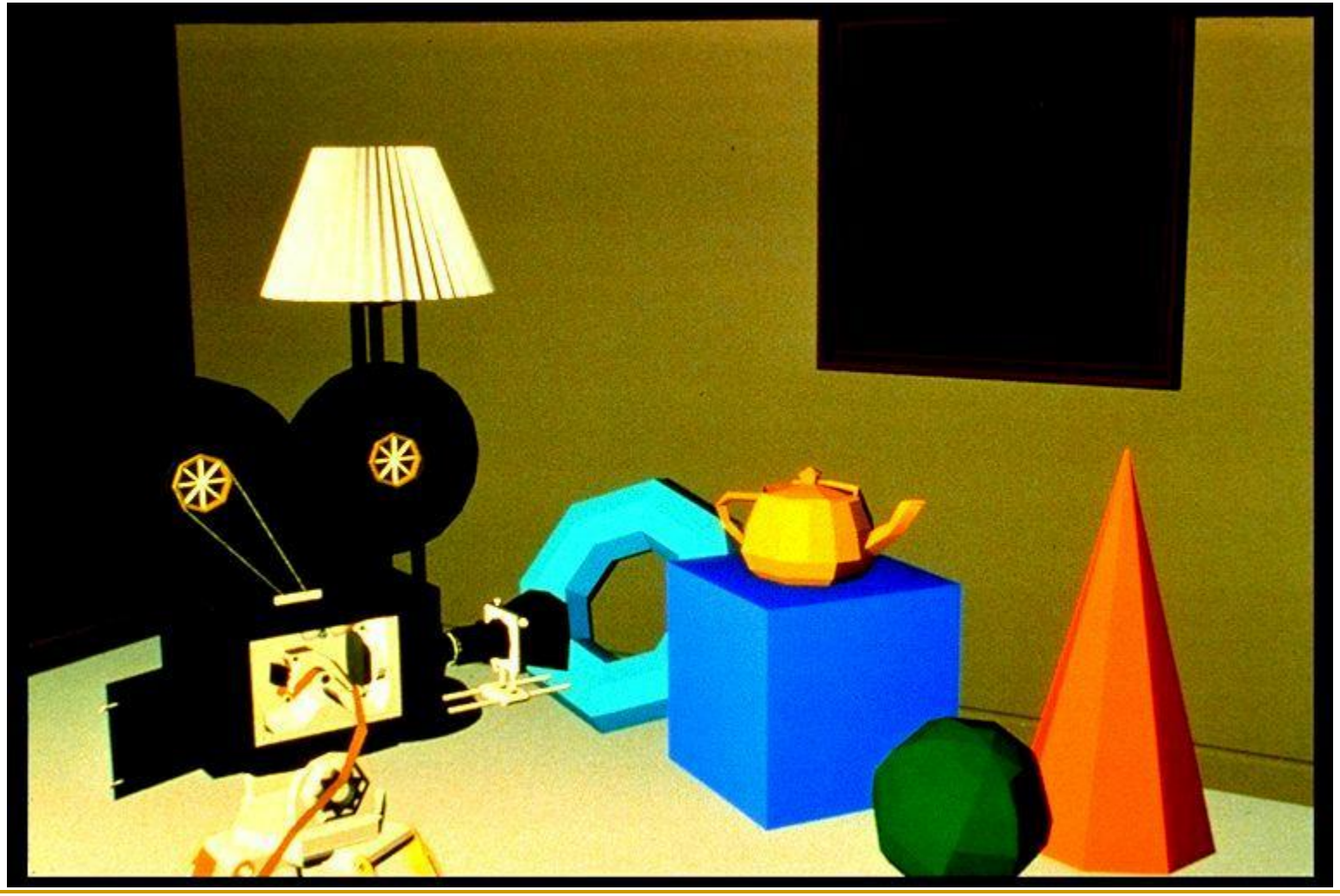
- ***Модел на Фонг***
  - много по-бавен от модела на Гуро
  - няма проблем с огледално отражение
  - може да се имплементира итеративно
  - обикновено се прилага като част от етапа на определяне на видими повърхнини



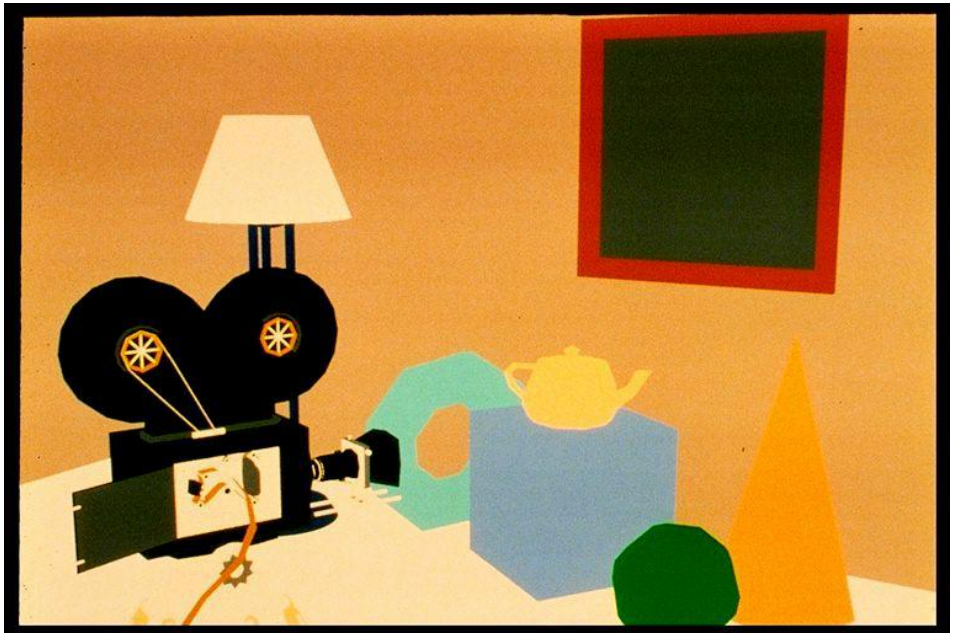
Модел за shading – без shading



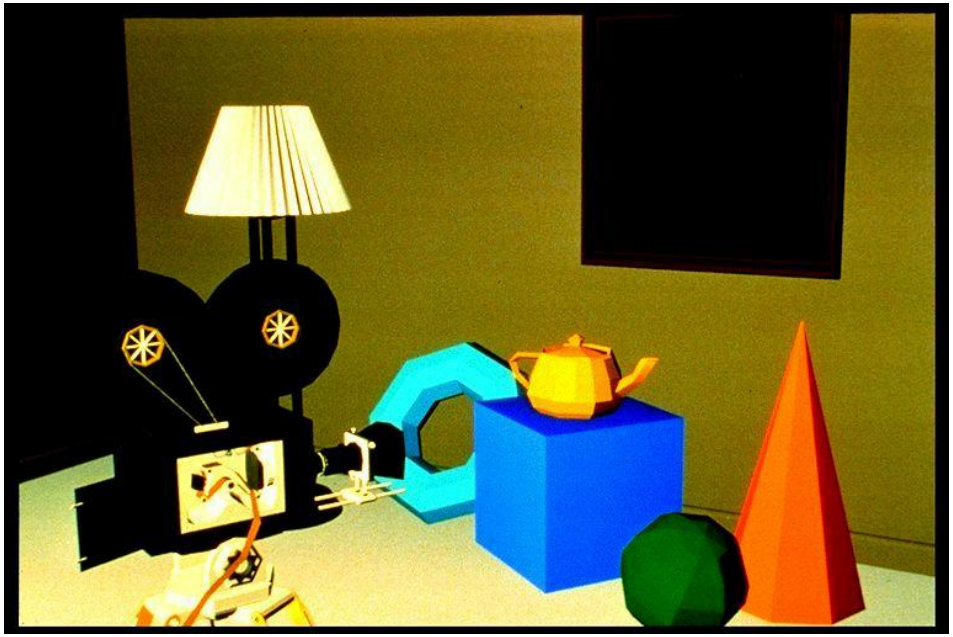
Модел за shading – плосък модел



Модел за shading

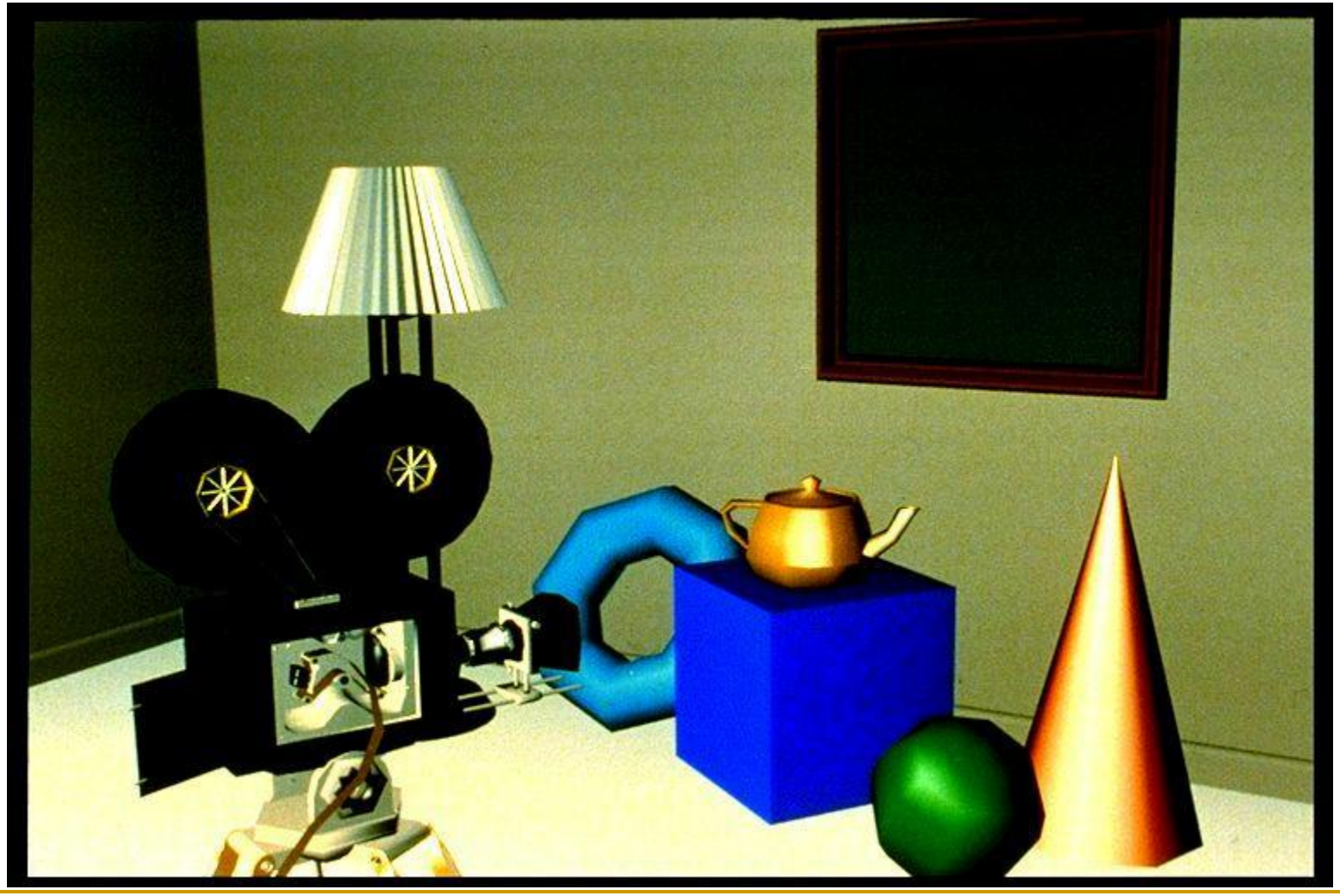


без shading

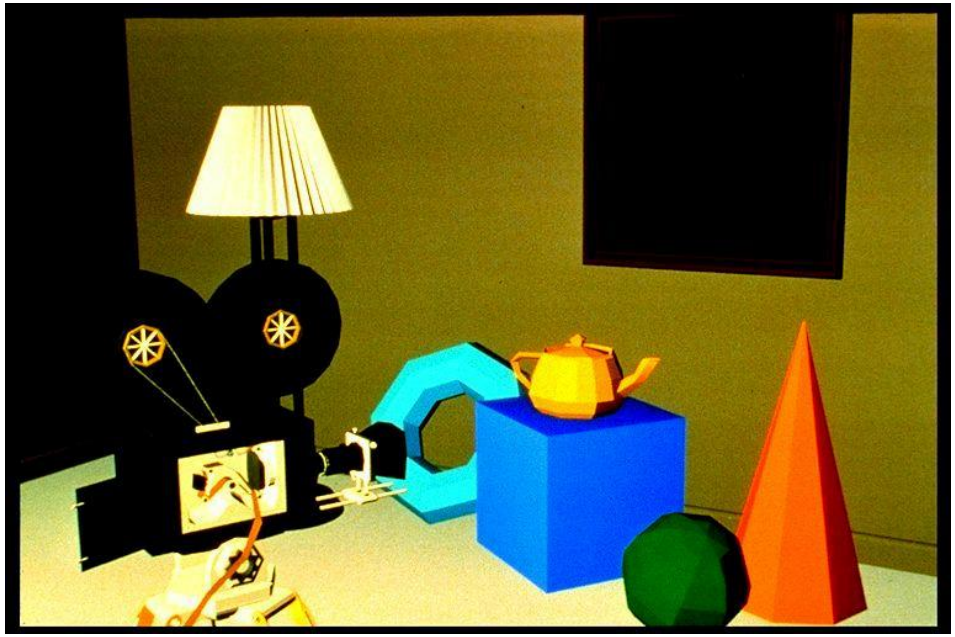


плосък модел

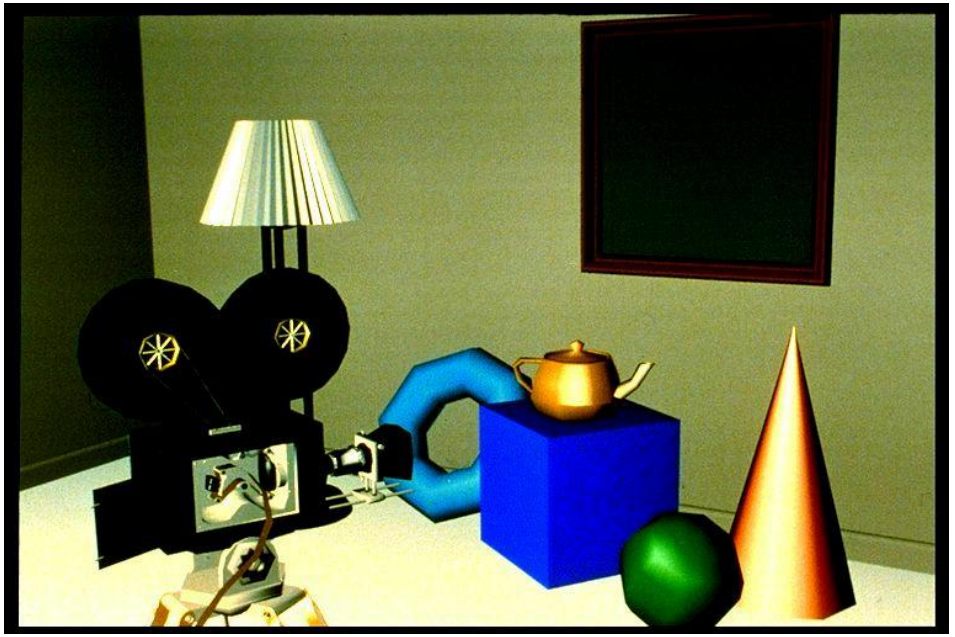
Модел за shading – модел на Гуро



Модел за shading

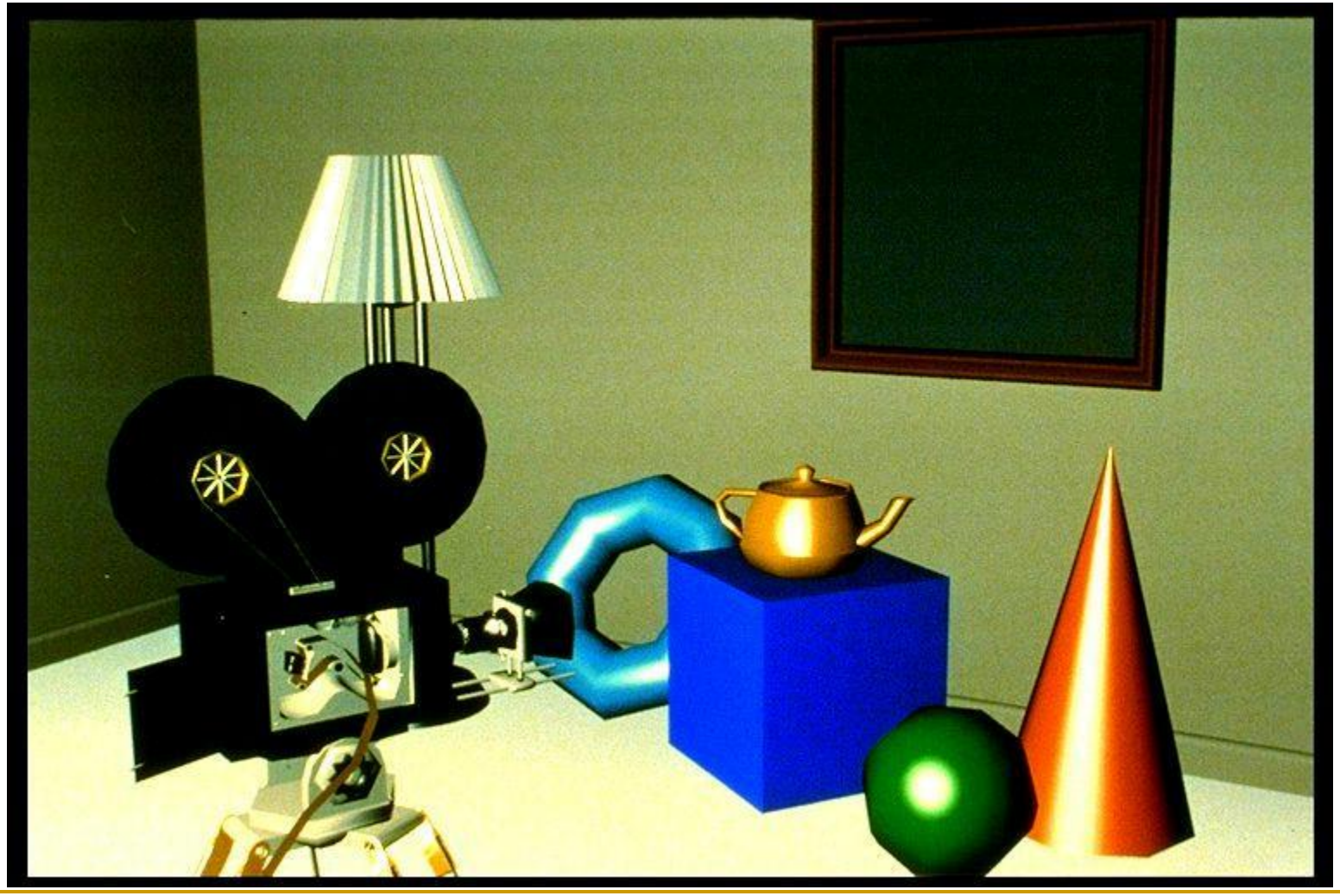


плосък модел

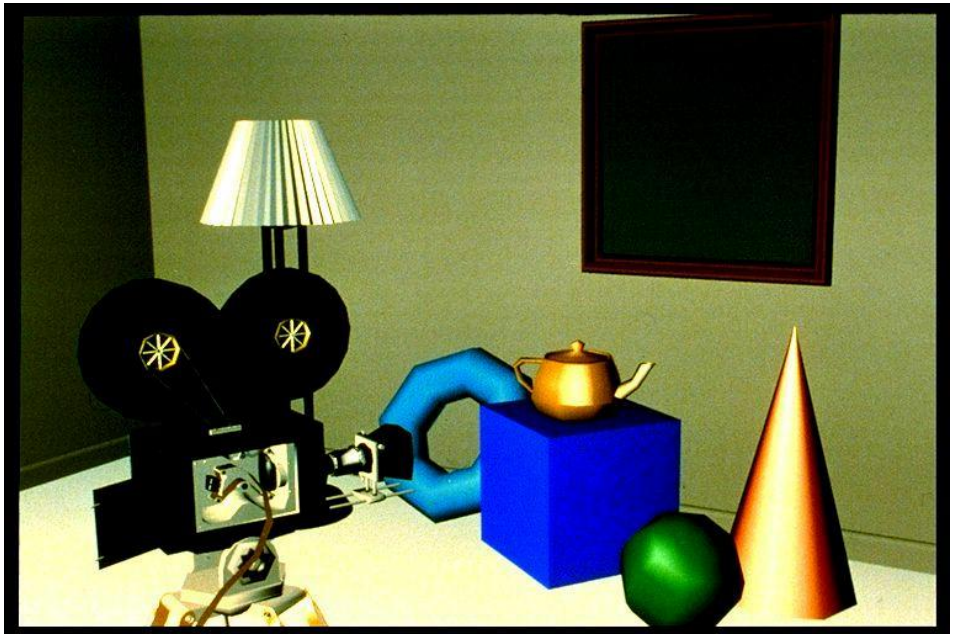


модел на Гуро

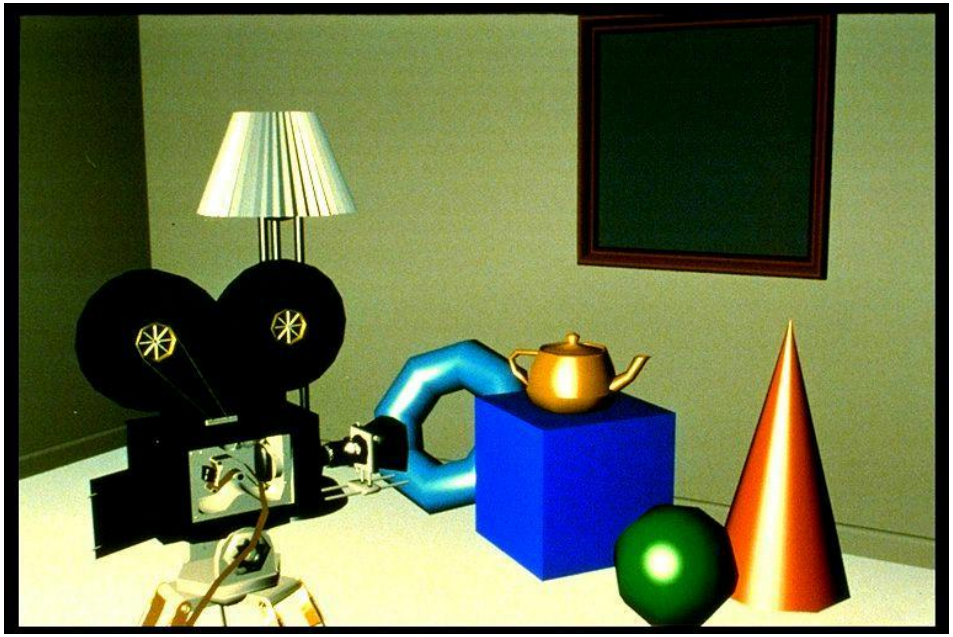
Модел за shading – модел на Фонг



Модел за shading



модел на Гуро

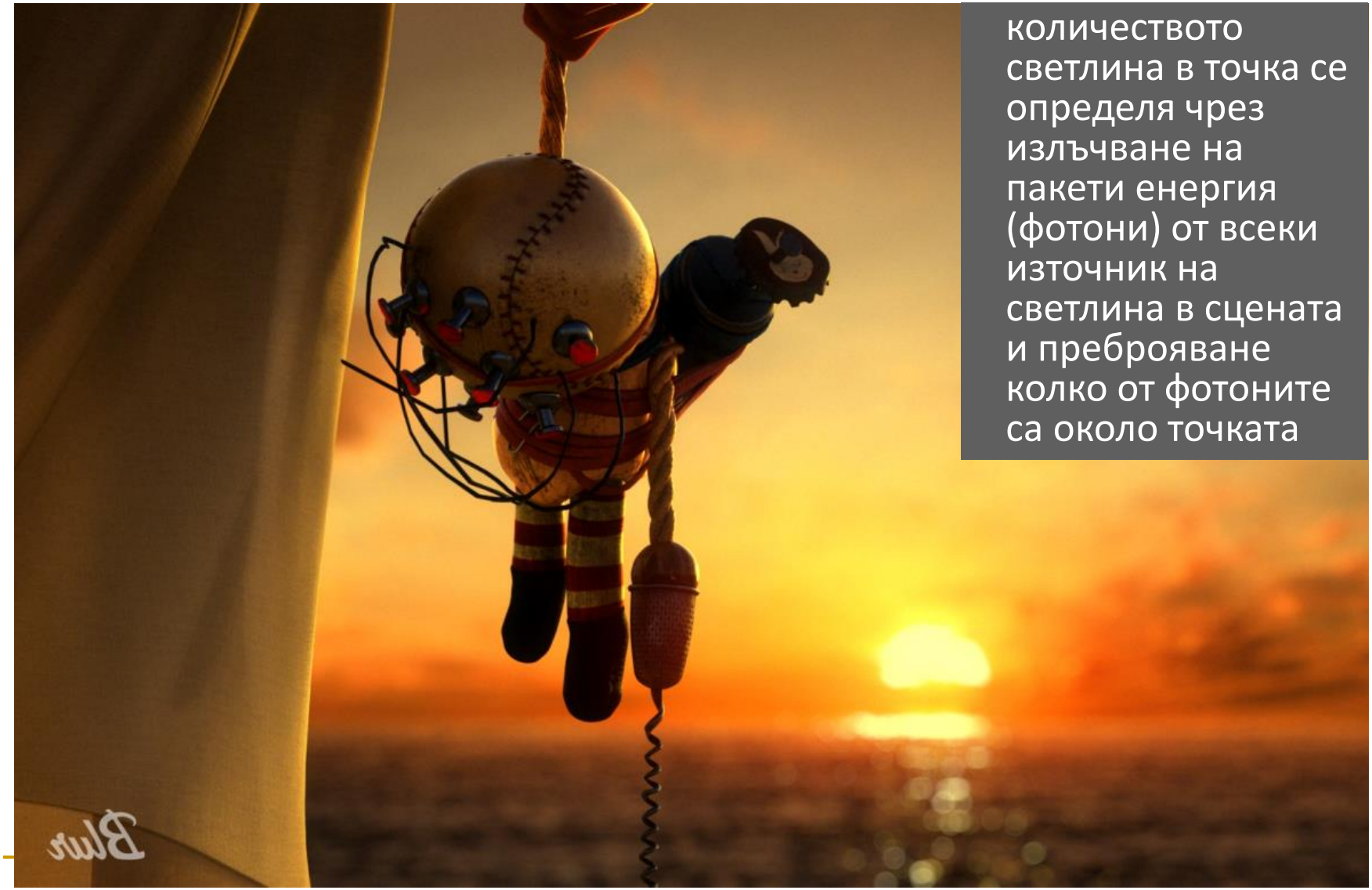


модел на Фонг

Специфични техники

- Изчисляването на глобалната осветеност на дадена сцена на практика е много сложно и изчислително интензивно
- Съществуват много алгоритми за изчисляване или апроксимиране на глобалната осветеност
  - Алгоритми, работещи в реално време
    - обкръжаващо закриване
    - осветяване базирано на изображения
  - Алгоритми, не работещи в реално време
    - Metropolis light transport, photon mapping, radiosity, point based color bleeding

Photon Mapping



количеството светлина в точка се определя чрез излъчване на пакети енергия (фотони) от всеки източник на светлина в сцената и преброяване колко от фотоните са около точката

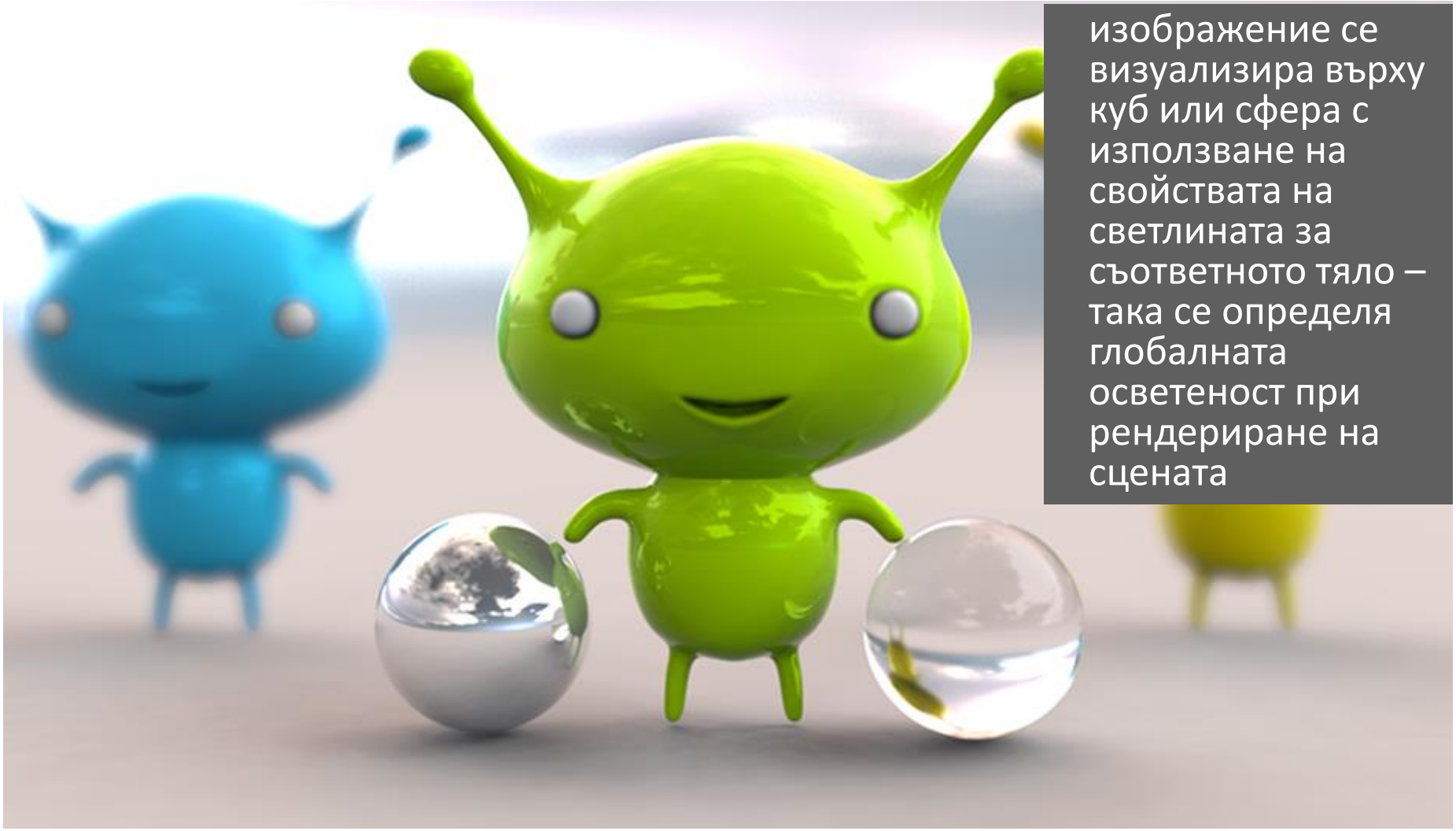
Bioshock 2, Mental Ray Renderer

Point-Based Color Bleeding

най-напред се генерира “облак” за директно осветяване на точката, облакът се използва за апроксимиране на дифузната светлина отразена от други обекти с отчитане на отражение от другите точки в облака

Up!, Renderman Renderer

Image Based Lighting



изображение се визуализира върху куб или сфера с използване на свойствата на светлината за съответното тяло – така се определя глобалната осветеност при рендериране на сцената

Shiny Aliens, Sunflow Renderer

Осветеност в OpenGL

- Осветеността в OpenGL се базира на модел на осветеност на Фонг
  - за всеки ***възел*** в модела на обекта се определя цвета на базата на материала на повърхността на обекта и източниците на светлина

- ***Фактори за определяне на осветеност***
  - ***Характеристики на източниците на светлина***
  - ***Характеристики на материали на повърхността***
  - ***Характеристики на модела на осветеност***

99

Осветеност в OpenGL

- За определяне на цвета на възел се използват ***четири*** изчислени цветови компоненти
  - ***Ambient***: осветеност на обекта от всички индиректни източници на светлина в сцената
  - ***Diffuse***: основният цвят на обекта при наличното осветяване
    - трябва да има източник на светлина насовен към обекта за да има дифузна компонента
  - ***Specular***: компонента за огледално осветяване на обекта
  - ***Emission***: компонента, която се добавя ако обектът излъчва светлина
    - например блестене (glow)

Материали в OpenGL

- Характеристиките на материала описват цвета и повърхностните свойства на материала на обекта
  - матов, лъскав, и т.н.

- Дефиниране на свойства за повърхността на примитив

```
glMaterialfv(face, property, value);
```

GL_DIFFUSE	основен цвят
GL_SPECULAR	цвят на огледално отражение
GL_AMBIENT	цвят при индиректно осветяване
GL_EMISSION	цвят на излъчване (блясък)
GL_SHININESS	гладкост на повърхността

Материали в OpenGL

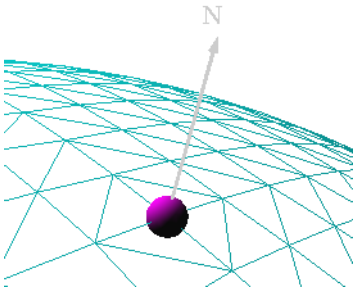
- Характеристики на материала в OpenGL
  - `GL_DIFFUSE`
    - основен цвят на обекта
  - `GL_SPECULAR`
    - цвят на огледално отразяване на обекта
  - `GL_AMBIENT`
    - цвят на обекта при индиректно осветяване
  - `GL_EMISSION`
    - цвят на излъчване на обекта (“светулка”)
  - `GL_SHININESS`
    - концентриране на огледално отражение за обекта
    - стойността варира
      - от 0 (много груба повърхност – няма огледално отражение)
      - до 128 (много лъскава повърхност)

Материали в OpenGL

- Характеристики на материала в OpenGL се задават както за ***предната***, така и за ***задната*** стена на обект според възлите на модела на обекта

- Характеристиките на материала се задават за предна и задна стена поотделно
  - `GL_FRONT`
    - за предна стена
  - `GL_BACK`
    - за задна стена
  - `GL_FRONT_AND_BACK`
    - за предна и задна стена едновременно

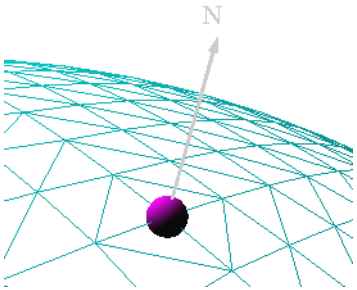
Нормали в OpenGL



- Нормала за осветеност в OpenGL определя как обект отразява светлина в близост до възел
  - `glNormal*()`
  - `glNormal3f(x, y, z)`
    - задава текуща нормала, която се използва за изчисляване на осветеност за всички възли докато не се зададе нова нормала

- Нормалите за осветеност трябва да са нормализирани до ***единична*** дължина за коректно определяне на осветеността
  - `glScale*()`
    - засяга нормали и възли, с което може да се промени дължината на нормала, така че след мащабирането вече да не е нормализирана

Нормали в OpenGL



- Автоматично нормализиране на нормалите в OpenGL
  - `glEnable(GL_NORMALIZE)`
  - `glEnable(GL_RESCALE_NORMAL)`
    - специален режим за развномерно мащабиране на нормалите
  - `GL_NORMALIZE`
    - нормализират се всички нормали, но се изисква изчисляване на квадратен корен, което намалява производителността

- Сплайн повърхнините и NURBS в OpenGL осигуряват автоматично генериране на нормали за възлите за изчисляване на осветеността

Източници на светлина в OpenGL

- Поддържат се поне осем източника на светлина в OpenGL
  - GL_LIGHT0 до GL_LIGHTn
  - за всеки източник се задават се параметрите на осветяване

- Характеристики на източниците на светлина в OpenGL, които могат да бъдат променяни от подразбиращите се стойности
  - ***Color properties***
    - позволяват различни взаимодействия с различни характеристики на материала на повърхностите
  - ***Position properties***
    - управляват позицията и типа на източника на светлина
  - ***Attenuation***
    - управлява естествената тенденция за намаляване на интензитета на светлината с увеличаване на разстоянието до източника

Източници на светлина в OpenGL

- Два типа светлинни източници в OpenGL
  - Точкови източници
    - ***Local*** (***Point***)
  - Насочени източници
    - ***Infinite*** (***Directional***)

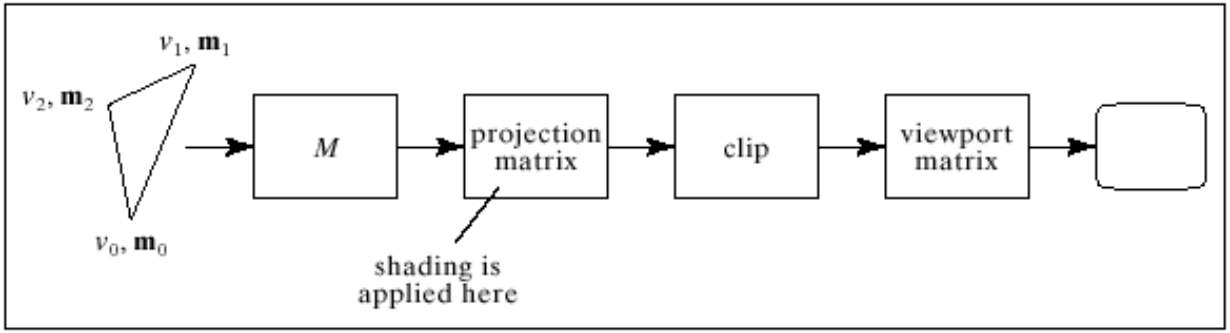
- Типа на светлинния източник се контролира от параметър w
  - $w = 0$ Безкрайно отдалечен източник насочен по $\begin{pmatrix} x & y & z \end{pmatrix}$
  - $w \neq 0$ Локален източник позициониран в $\begin{pmatrix} x/w & y/w & z/w \end{pmatrix}$

Източници на светлина в OpenGL

- Задаване на параметрите на светлинен източник
  - `glLightfv(`*`light, property, value`*`);`
    - *`light`* определя за кой източник са зададените параметри
    - *`property`* определя цвят, позиция и тип, намаляване на интензитета
- Всеки светлинен източник в OpenGL се контролира поотделно
  - задаване на светлинен източник с конкретен номер
    - `glEnable(GL_LIGHT`*`n`*`);`
  - задаване дали осветеността ще се използва при определяне на цветовете на примитивите
    - `glEnable(GL_LIGHTING);`
      - така осветеността може да се разрешава или забранява без да се променят всички отделни компоненти

Осветеност в OpenGL

- Осветеността се определя ***след*** трансформацията ModelView



  - Позволява ротация на светлинния източник заедно със сцената
    - възлите и нормалните вектори се трансформират в координатната система на камерата

Осветеност в OpenGL

- Modelview матрицата променя позицията на източниците на светлина
  - могат да се получат различни ефекти в зависимост от това ***кога*** се задава позицията на светлинния източник
- Три координатни системи, в които може да бъде зададена позиция/посока на източник на светлина
  - ***координати на камерата***
  - ***световни координати***
  - ***координати на модела на обекта***
- За управление и определяне на позицията на светлинния източник могат да се използват push и pop на матрици в матрични стек

Осветеност в OpenGL

- ***Координати на камерата***
  - определят се с единична матрица в ModelView
  - ако се зададе позиция/посока на светлинен източник, той остава фиксиран за равнината на изображението
    - независимо как се трансформират обектите, огледалното осветяване остава едно и също спрямо позицията на наблюдение
- ***Световни координати***
  - само визуализиращата трансформация е в матрицата ModelView
  - позицията/посоката на светлинен източник е фиксирана за сцената
    - аналогично на улична лампа на стълб
- ***Координати на модела на обекта***
  - всяка комбинация на визуализираща и моделиращи трансформации в матрицата ModelView
  - позволява произволно (и анимирано) позициониране на светлинния източник с използване на моделиращи трансформации

Осветеност в OpenGL

- Задаване на източник на светлина
  - максимум 8 светлинни източника
    - `GL_LIGHT0, GL_LIGHT1, …, GL_LIGHT7`
  - характеристики
    - позиция
    - фонова, дифузна и огледална компоненти
    - намаляване на интензитета
    - локална или отдалечена позиция на наблюдение
    - осветяване на предни и задни стени
    - глобална фонова светлина

Осветеност в OpenGL

- Позиция
  - насочен или позиционен източник
    - пример за структурата `LightInfo` инициализирана с единствен източник на бяла светлина

```
// Lighting
static int numActiveLights;
typedef struct _LightInfo {
  GLfloat xyz[4];
  GLfloat *rgb;
  int enable;
} LightInfo;

// Light information
LightInfo linfo[] = {
   {{ 0.0f,  0.0f,  2.0f, 0.0f}, white},
};
const int MAX_LIGHTS = (sizeof(linfo) / sizeof(linfo[0]));
```

Осветеност в OpenGL

- Инициализация в OpenGL с LightInfo

```
glLightfv(GL_LIGHT0 + num, GL_SPECULAR, dim);
glLightfv(GL_LIGHT0 + num, GL_POSITION, linfo[num].xyz);
glLightfv(GL_LIGHT0 + num, GL_DIFFUSE,  linfo[num].rgb);
```

- Задаване на изчисления с използване на осветеност

```
glEnable(GL_LIGHTING);
```

  - за дадени източници

```
glEnable(GL_LIGHT0 + num);
```

- Задаване на характеристики на материала

```
glMaterialfv(GL_FRONT, GL_AMBIENT,  matAmb);
glMaterialfv(GL_FRONT, GL_DIFFUSE,  matDiff);
glMaterialfv(GL_FRONT, GL_SPECULAR, matSpec);
glMaterialfv(GL_FRONT, GL_EMISSION, matEmission);
glMaterialf(GL_FRONT, GL_SHININESS, 100.0);
```

Осветеност в OpenGL – пример

```
#include <stdlib.h>
#include <GL/glut.h>
/*  Initialize z-buffer, projection matrix, light source, and lighting model.
 *  Do not specify a material property here. */
void init(void)
{
   GLfloat ambient[] = { 0.0, 0.0, 0.0, 1.0 };
   GLfloat diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat specular[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat position[] = { 0.0, 3.0, 2.0, 0.0 };
   GLfloat lmodel_ambient[] = { 0.4, 0.4, 0.4, 1.0 };
   GLfloat local_view[] = { 0.0 };

   glClearColor(0.0, 0.1, 0.1, 0.0);
   glEnable(GL_DEPTH_TEST);
   glShadeModel(GL_SMOOTH);

   glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
   glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
   glLightfv(GL_LIGHT0, GL_POSITION, position);
   glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);
   glLightModelfv(GL_LIGHT_MODEL_LOCAL_VIEWER, local_view);

   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
}
```

Осветеност в OpenGL – пример

```
/*  Draw twelve spheres in 3 rows with 4 columns.
 *  The spheres in the first row have materials with no ambient reflection.
 *  The second row has materials with significant ambient reflection.
 *  The third row has materials with colored ambient reflection.
 *  The first column has materials with blue, diffuse reflection only.
 *  The second column has blue diffuse reflection, as well as specular
 *  reflection with a low shininess exponent.
 *  The third column has blue diffuse reflection, as well as specular
 *  reflection with a high shininess exponent (a more concentrated highlight).
 *  The fourth column has materials which also include an emissive component */

void display(void)
{
   GLfloat no_mat[] = { 0.0, 0.0, 0.0, 1.0 };
   GLfloat mat_ambient[] = { 0.7, 0.7, 0.7, 1.0 };
   GLfloat mat_ambient_color[] = { 0.8, 0.8, 0.2, 1.0 };
   GLfloat mat_diffuse[] = { 0.1, 0.5, 0.8, 1.0 };
   GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat no_shininess[] = { 0.0 };
   GLfloat low_shininess[] = { 5.0 };
   GLfloat high_shininess[] = { 100.0 };
   GLfloat mat_emission[] = {0.3, 0.2, 0.2, 0.0};

   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Осветеност в OpenGL – пример

```
/*  draw sphere in first row, first column
 *  diffuse reflection only; no ambient or specular */
   glPushMatrix();
   glTranslatef (-3.75, 3.0, 0.0);
   glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
   glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
   glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
   glutSolidSphere(1.0, 16, 16);
   glPopMatrix();

/*  draw sphere in first row, second column
 *  diffuse and specular reflection; low shininess; no ambient */
   glPushMatrix();
   glTranslatef (-1.25, 3.0, 0.0);
   glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, low_shininess);
   glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
   glutSolidSphere(1.0, 16, 16);
   glPopMatrix();
```

Осветеност в OpenGL – пример

```
/*  draw sphere in first row, third column
 *  diffuse and specular reflection; high shininess; no ambient */
   glPushMatrix();
   glTranslatef (1.25, 3.0, 0.0);
   glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);
   glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
   glutSolidSphere(1.0, 16, 16);
   glPopMatrix();

/*  draw sphere in first row, fourth column
 *  diffuse reflection; emission; no ambient or specular reflection */
   glPushMatrix();
   glTranslatef (3.75, 3.0, 0.0);
   glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
   glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
   glMaterialfv(GL_FRONT, GL_EMISSION, mat_emission);
   glutSolidSphere(1.0, 16, 16);
   glPopMatrix();
```

Осветеност в OpenGL – пример

```
/*  draw sphere in second row, first column
 *  ambient and diffuse reflection; no specular */
   glPushMatrix();
   glTranslatef (-3.75, 0.0, 0.0);
   glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
   glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
   glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
   glutSolidSphere(1.0, 16, 16);
   glPopMatrix();

/*  draw sphere in second row, second column
 *  ambient, diffuse and specular reflection; low shininess */
   glPushMatrix();
   glTranslatef (-1.25, 0.0, 0.0);
   glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, low_shininess);
   glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
   glutSolidSphere(1.0, 16, 16);
   glPopMatrix();
```

Осветеност в OpenGL – пример

```
/*  draw sphere in second row, third column
 *  ambient, diffuse and specular reflection; high shininess */
   glPushMatrix();
   glTranslatef (1.25, 0.0, 0.0);
   glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);
   glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
   glutSolidSphere(1.0, 16, 16);
   glPopMatrix();

/*  draw sphere in second row, fourth column
 *  ambient and diffuse reflection; emission; no specular */
   glPushMatrix();
   glTranslatef (3.75, 0.0, 0.0);
   glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
   glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
   glMaterialfv(GL_FRONT, GL_EMISSION, mat_emission);
   glutSolidSphere(1.0, 16, 16);
   glPopMatrix();
```

Осветеност в OpenGL – пример

```
/*  draw sphere in third row, first column
 *  colored ambient and diffuse reflection; no specular */
   glPushMatrix();
   glTranslatef (-3.75, -3.0, 0.0);
   glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient_color);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
   glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
   glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
   glutSolidSphere(1.0, 16, 16);
   glPopMatrix();

/*  draw sphere in third row, second column
 *  colored ambient, diffuse and specular reflection; low shininess */
   glPushMatrix();
   glTranslatef (-1.25, -3.0, 0.0);
   glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient_color);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, low_shininess);
   glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
   glutSolidSphere(1.0, 16, 16);
   glPopMatrix();
```

Осветеност в OpenGL – пример

```
/*  draw sphere in third row, third column
 *  colored ambient, diffuse and specular reflection; high shininess */
   glPushMatrix();
   glTranslatef (1.25, -3.0, 0.0);
   glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient_color);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);
   glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
   glutSolidSphere(1.0, 16, 16);
   glPopMatrix();

/*  draw sphere in third row, fourth column
 *  colored ambient and diffuse reflection; emission; no specular */
   glPushMatrix();
   glTranslatef (3.75, -3.0, 0.0);
   glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient_color);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
   glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
   glMaterialfv(GL_FRONT, GL_EMISSION, mat_emission);
   glutSolidSphere(1.0, 16, 16);
   glPopMatrix();

   glFlush();
}
```

Осветеност в OpenGL – пример

```
void reshape(int w, int h)
{
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   if (w <= (h * 2))
      glOrtho (-6.0, 6.0, -3.0*((GLfloat)h*2)/(GLfloat)w,
         3.0*((GLfloat)h*2)/(GLfloat)w, -10.0, 10.0);
   else
      glOrtho (-6.0*(GLfloat)w/((GLfloat)h*2),
         6.0*(GLfloat)w/((GLfloat)h*2), -3.0, 3.0, -10.0, 10.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)
{
   switch (key) {
      case 27:
         exit(0);
         break;
   }
}
```

Осветеност в OpenGL – пример

```
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize (600, 450);
   glutCreateWindow(argv[0]);
   init();
   glutReshapeFunc(reshape);
   glutDisplayFunc(display);
   glutKeyboardFunc (keyboard);
   glutMainLoop();
   return 0;
}
```

КРАЙ

Следваща тема:

Визуализиране с алгоритъм с трасиране на лъчи