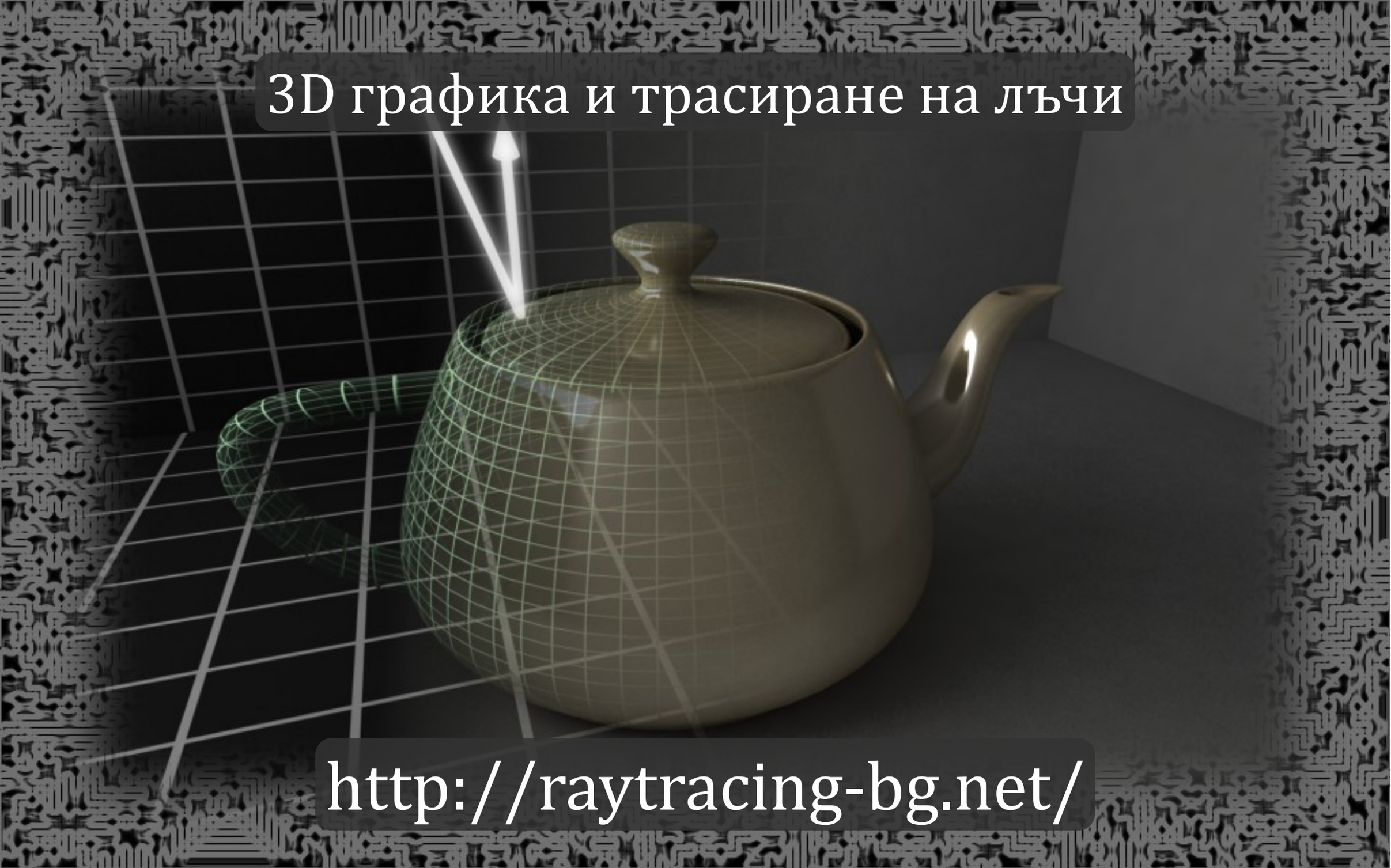
3D графика и трасиране на лъчи



http://raytracing-bg.net/

Тема 4

Въведение в камерата
Пресичане с равнини и кълбета

Съдържание

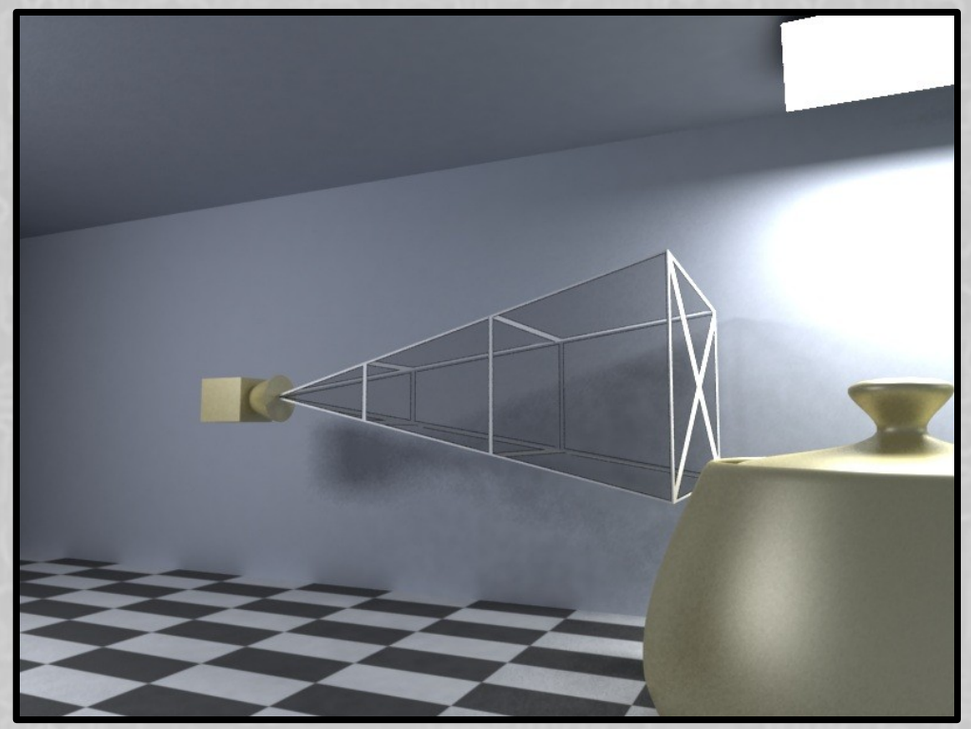
- Реализация на правоъгълна камера
- Реализация на raytracing алгоритъма
- Реализация на пресичания с равнина и сфера
- Шейдъри и текстури
- Сенки
- Домашни :)

Разходка из кода на проекта

- svn://anrieff.no-ip.org/fmiray
- Изчакване в SDL (SDL events)
- Код, който ще ползваме наготово: Color, Vector, Matrix

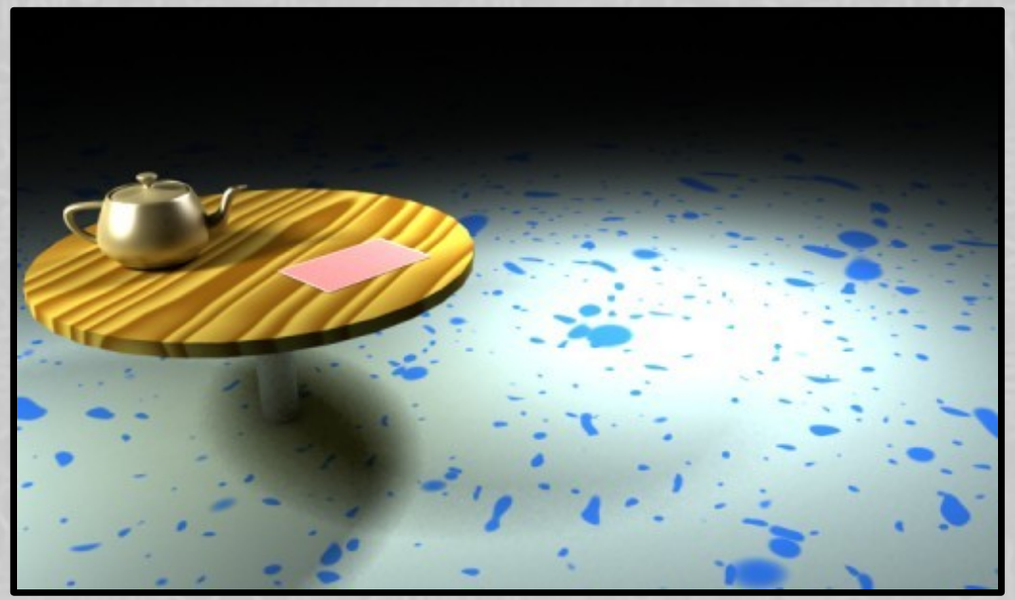
Реализация на камерата

- Ще реализираме стандартна, правоъгълна (rectilinear) камера
- Зрителното поле на камерата е във формата на правоъгълна пирамида
- Нарича се още pinhole camera
  - Няма фокус, DOF-ът е безкраен



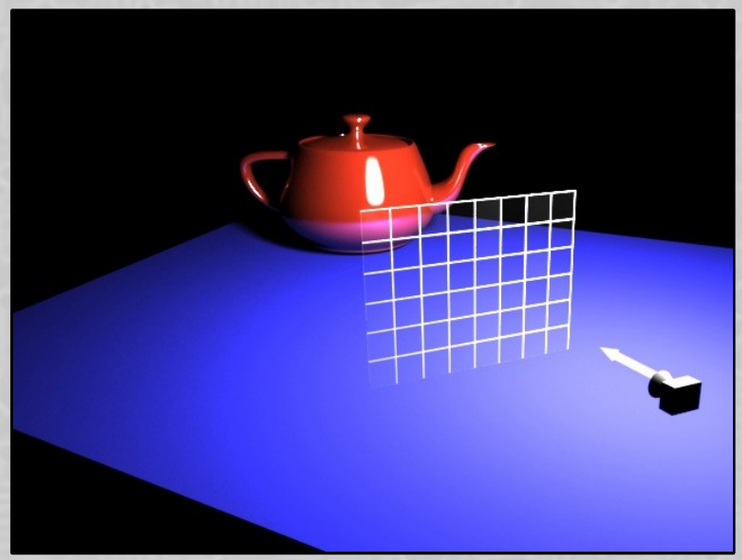
Реализация на камерата

- Ако разположим лист милиметрова хартия пред камерата, при подходяща разделителна способност, всяко квадратче ще отговаря на един пиксел
- Демонстрация [millimetre.m4v]



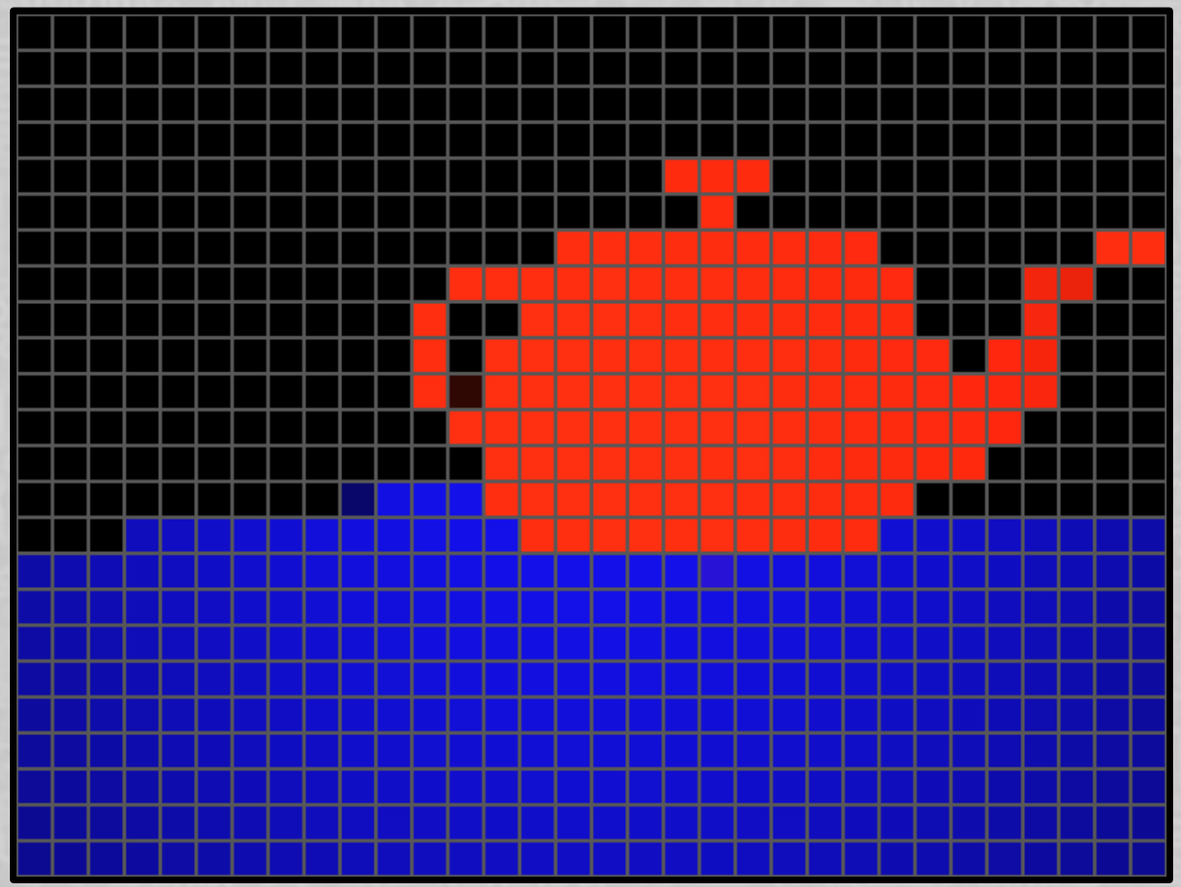
Реализация на камерата

- Най-простият възможен рейтрейсър: изстрелваме лъч през всеки пиксел и оцветяваме пиксела в зависимост от ударения обект
  - Например: черно, ако не ударим нищо; червено за чайник; синьо за равнина
- Демонстрация [primitive.m4v]



Реализация на камерата

- Резултатната картинка:



Реализация на камерата

- Ще симулираме листа „милиметрова хартия“
- Трябват ни просто три от краищата на листа (горен ляв, горен десен, долен ляв)
- Намиране на произволен пиксел (по зададени X, Y)
  - За X направлението, интерполираме между горен ляв и горен десен край
  - За Y направлението, интерполираме между горен ляв и долен ляв край
  - `DestP = TopLeft + (TopRight-TopLeft)*(X/screenW) + (BottomLeft-TopLeft)*(Y/screenH)`

Реализация на камерата

- Как генерираме трите краища?
  - Ще предполагаме, че камерата се намира в (0,0,0) и гледа в посока на оста Z
  - Ще разположим лист със зададеното отношение (aspect ratio), на разстояние 1 от камерата (Z = 1)
  - Ще мащабираме листа, така че диагоналния ъгъл да стане колкото искаме (FOV)
    - За наше удобство, ще задаваме FOV-а в градуси
  - Ще приложим трите ротации (roll, pitch и yaw) върху краищата

Генериране на лъчи

- Генерирането на лъчи след това става по споменатата формула, като трябва да транслираме лъча да тръгва от позицията на камерата
  - Самия лъч задаваме като двойка (начало, посока), като посоката е единичен вектор.

Рендерирането на сцената

- ```
  for each y in rows:
      for each x in columns:
          ray = camera.getRay(x, y)
          vfb[x, y] = raytrace(ray)
  ```
- `raytrace()` е функция, която, по даден лъч, намира цвета на първия ударен обект по протежение на лъча
  ```

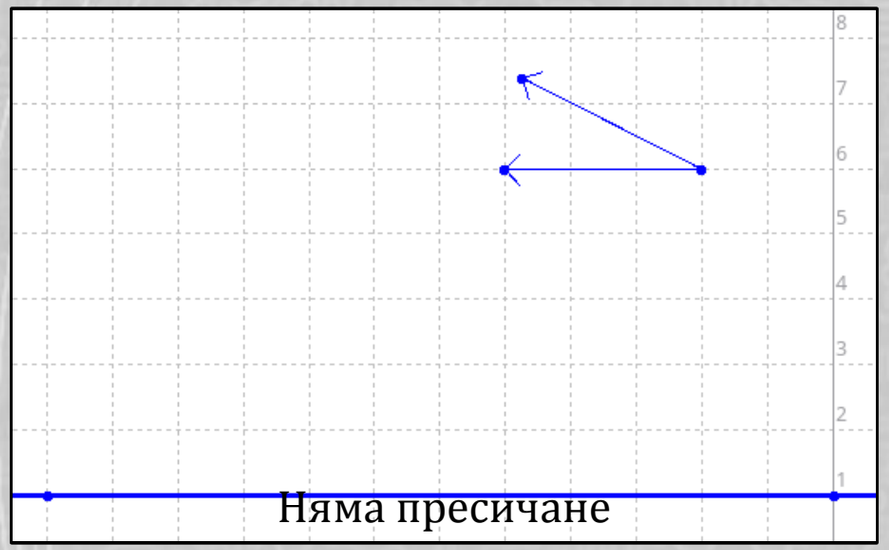
Raytracing алгоритъма

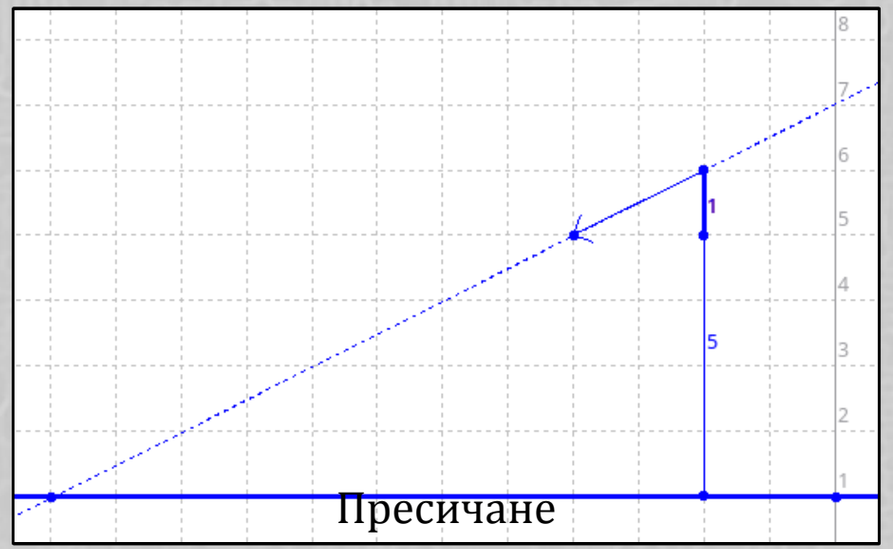
- **Function** raytrace(ray):

```
    closestDist = +∞
    closestNode = None
    for each node in sceneNodes:
        intersectionInfo = intersect(ray, node)
        if intersectionInfo.distance < closestDist:
            closestDist = intersectionInfo.distance
            closestNode = intersectionInfo.node
    if closestDist == +∞:
        return backgroundColor
    return closestNode.computeColor()
```

Пресичане с равнина

- Ще реализираме обекта „равнина“. Равнината ще е успоредна на XZ координатната равнина, ще е зададено само разстоянието по y от XZ
- Имаме два случая:





Пресичане със сфера

- Разстоянието от центъра на сферата O до произволна точка T е: $\texttt{sqrt}((T_x-O_x)^2 + (T_y-O_y)^2 + (T_z-O_z)^2)$
- Позицията на всяка точка по лъча е: `start + p * dir`
  - Където $p > 0$ е параметър
  - Разстоянието от всяка точка на лъча до сферата се получава по горната формула, като заместим `T = start + p * dir`, за някое конкретно p
  - Търсим такова p, че разстоянието да е точно R (радиуса на сферата)

Пресичане със сфера

- $sqrt(((start_x + p*dir_x)-O_x)^2 + ((start_y + p*dir_y)-O_y)^2 + ((start_z + p*dir_z)-O_z)^2) = R \Leftrightarrow$
  - (нека H = start - O):
- $sqrt((H_x + p*dir_x)^2 + (H_y + p*dir_y)^2 + (H_z + p*dir_z)^2) = R \Leftrightarrow$
- $(H_x + p*dir_x)^2 + (H_y + p*dir_y)^2 + (H_z + p*dir_z)^2 = R^2 \Leftrightarrow$
- $p^2(dir_x^2+dir_y^2+dir_z^2) + 2*p(H_xdir_x + H_ydir_y + H_zdir_z) + (H_x^2 + H_y^2 + H_z^2 - R^2) = 0 \Leftrightarrow$
- $p^2 * dir.length^2 + p * (2 * dir \cdot H) + (H.length^2 - R^2) = 0$
- Квадратно уравнение!
  - 0 решения – лъчът не пресича сферата
  - 1 или 2 решения – лъчът допира или пресича сферата

Осветление

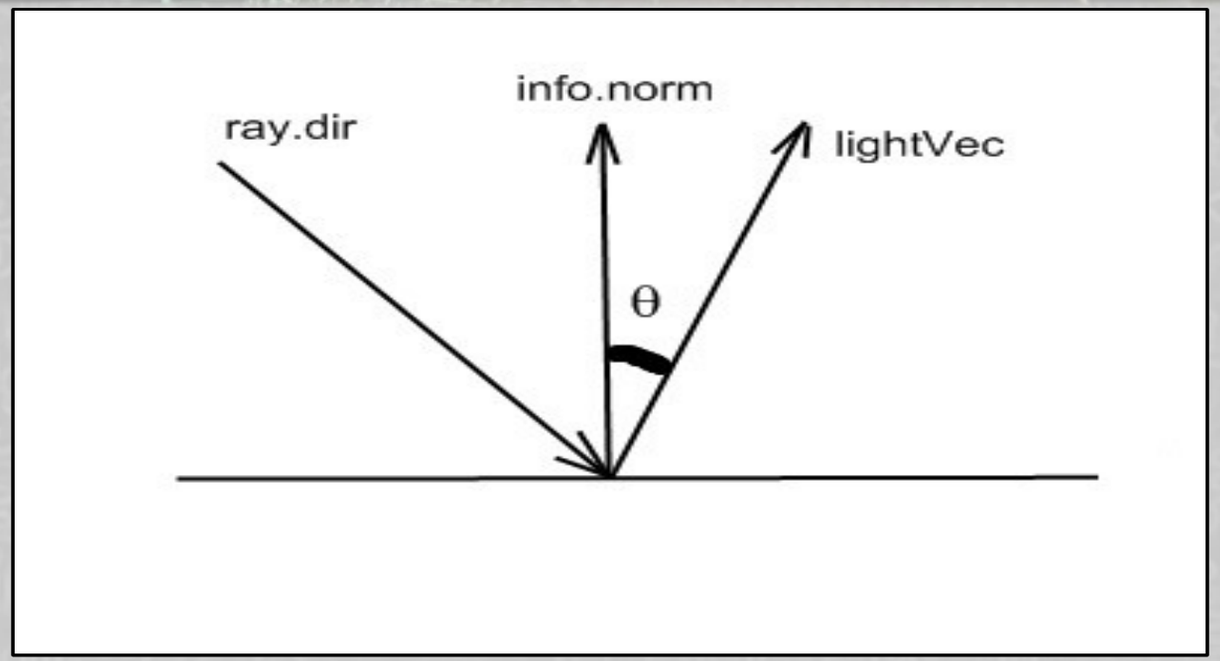
- Ще използваме най-простия модел за лампа – точкова
- Точковата лампа има две характеристики
  - Позиция
  - Интензитет
- Точковата лампа разпръсква светлина във всички посоки
- Осветеността на дадена точка намалява с квадрата на отдалечеността ѝ от лампата (1/(r*r))
  - Защо?

Материали

- Ще напишем 2 прости материала
  - Ламберт (или дифузен) модел
    - За матови повърхности, които не хвърлят отблясъци
  - Фонг (Phong) модел
    - За „лъскави“ повърхности

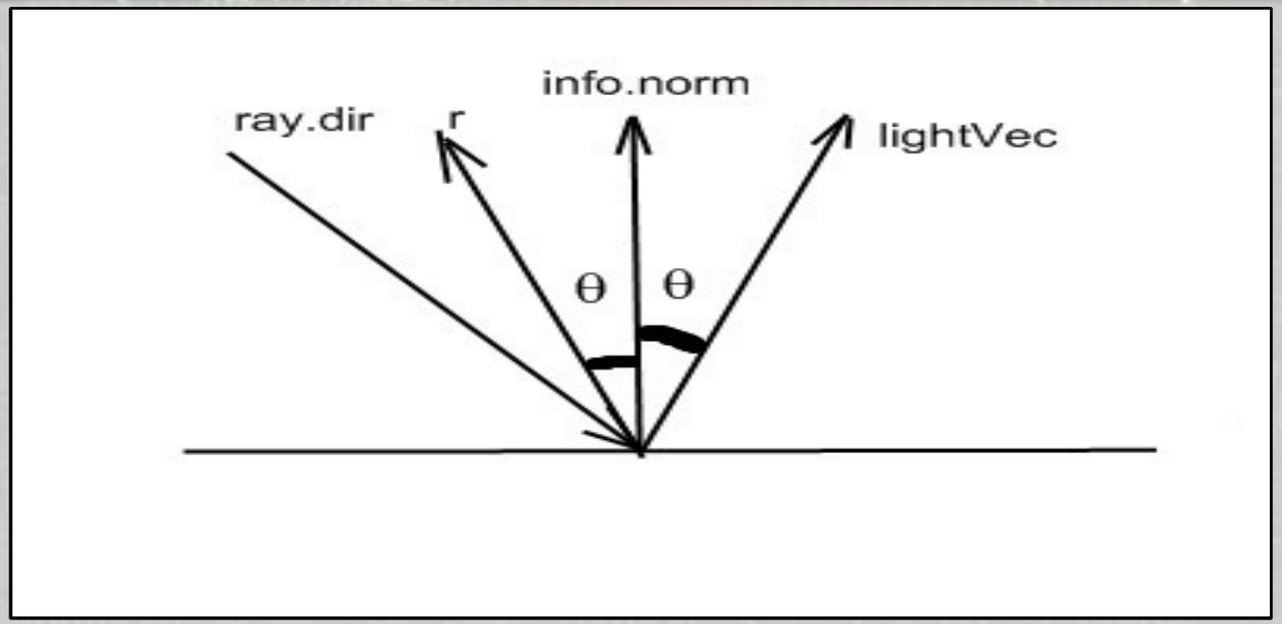
1 картинка струва поне 1000 думи

Ламберт



- OutColor = materialColor * max(0, cos(theta)) * lightColor
- cos(theta) = dot(lightVec, info.norm)

Фонг (Phong)



- R е отражението на lightVec спрямо нормалата info.norm
- OutColor = pow(dot(-ray.dir, r), exponent) * lightColor
- Exponent контролира „колко лъскав е материала“

Домашни

- Дадени са 5 задачи – условията се намират на сайта на курса (http://raytracing-bg.net/)
- Краен срок – 12 по обяд на 6 Ноември 2009.