Computer Graphics

Review

Taku Komura

Overview

- Review
- Some additional things

Review

- Graphics pipeline
- Transformation of 3D objects, View transformation
- Projection and Rasterization
- Illumination
- Hidden surface removal
- Texture mapping, bump mapping, environment mapping
- Anti-alising
- Shadows
- Global Illumination
- Curves and surfaces

Graphics Pipeline

Three stages

- Application stage
  - Entirely done in the CPU
  - Loading data, getting user input
- Geometric stage
  - Applying transformation to vertices
  - Computing the attributes for the vertices
- Rasterization stage
  - Per pixel computation
  - Converting the continuous representation to the discrete representations

Which stage do the following processes belong to?

- Getting the input from the user and deciding the view point
- Anti-aliasing
- Transforming the vertex location into that of the canonical view volume
- Computing the color of the pixel by Phong shading

Object Representation

Parameteric surfaces

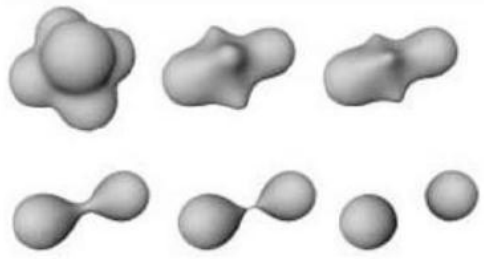
- Bezier, B-spline, NURBS

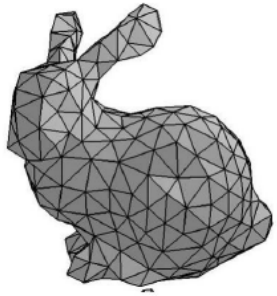
Implicit surfaces

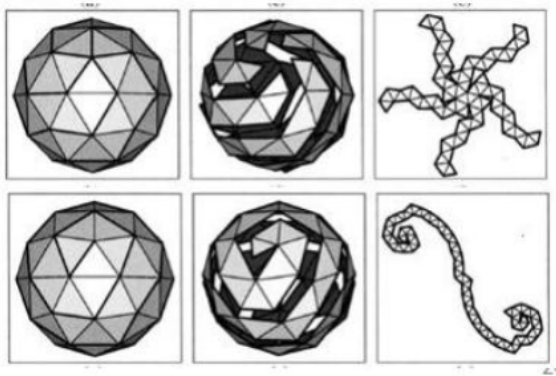
- Metaballs

Polygon mesh

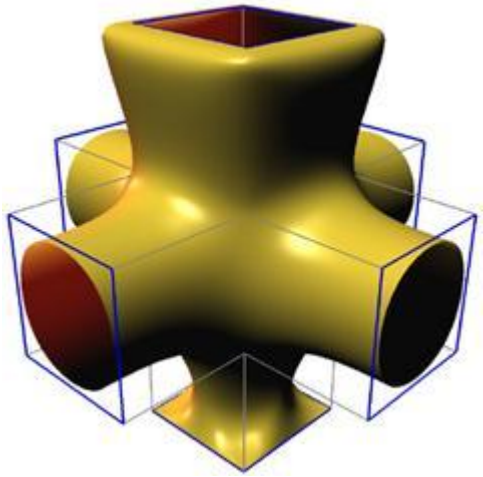
- Polygon format
- Triangle strips
- Subdivision surface

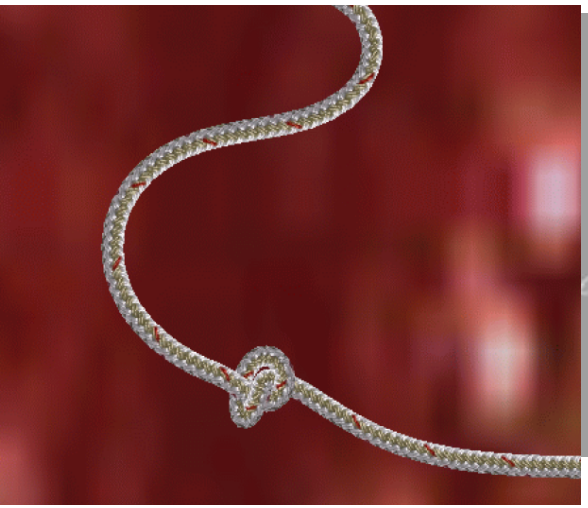


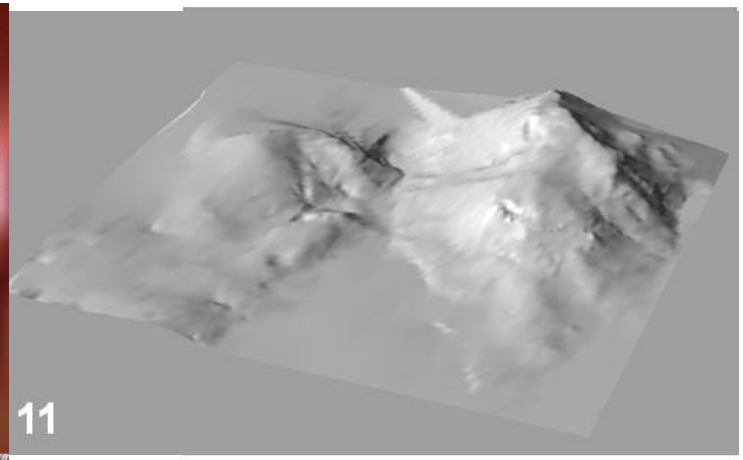


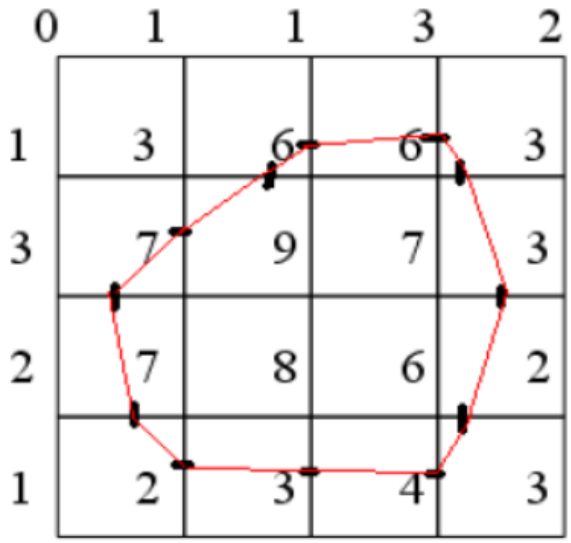


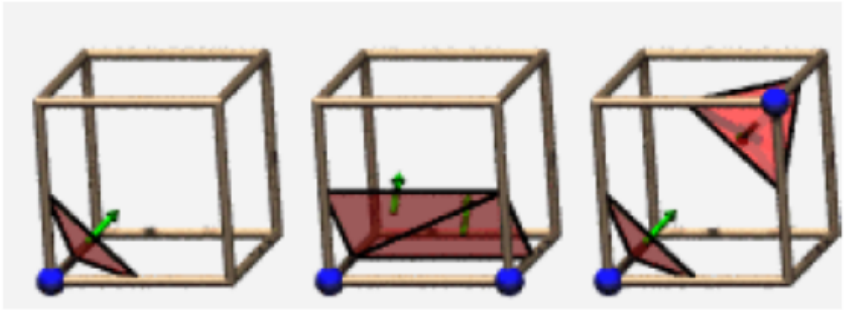
What are the good ways to model the following objects?








0 1 1 3 2
1 3 6 6 3
3 7 9 7 3
2 7 8 6 2
1 2 3 4 3



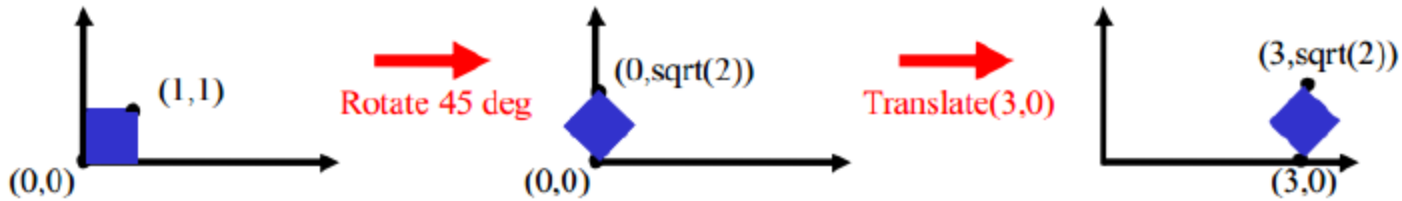
Transformation Matrix

Translation, Rotation and Scaling

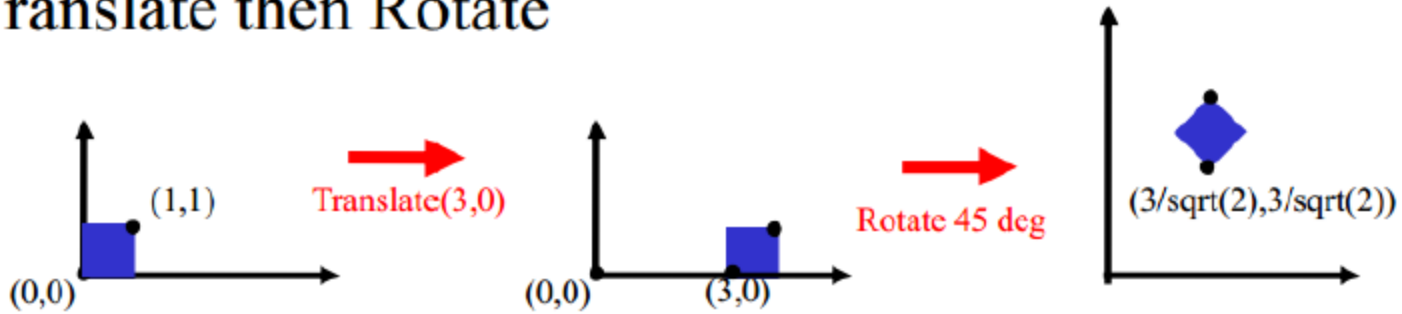
$$T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix concatenation

Rotate then Translate



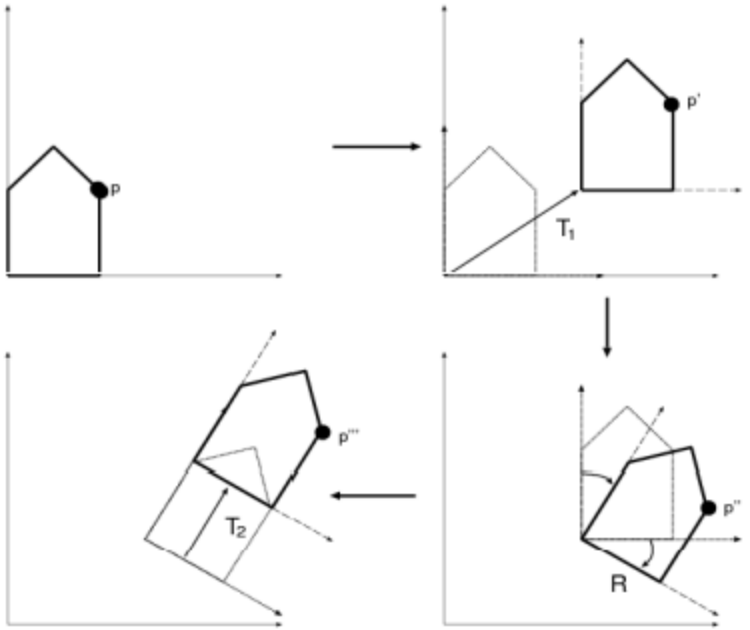
Translate then Rotate



x'= RT x

x'' = TR x

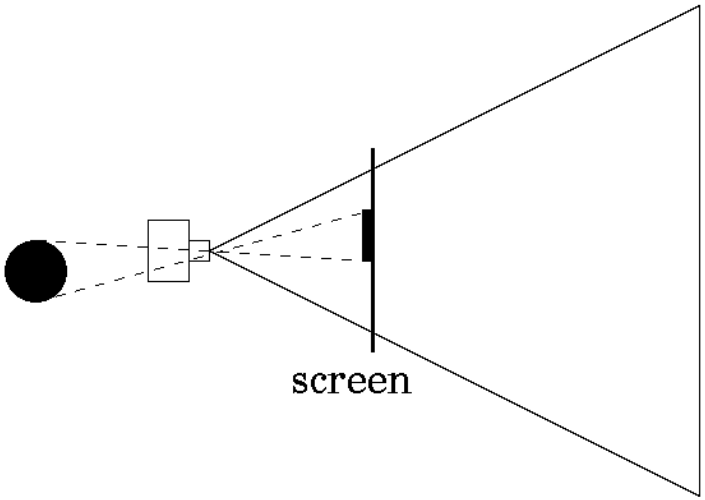
Transformations of coordinate systems - Example

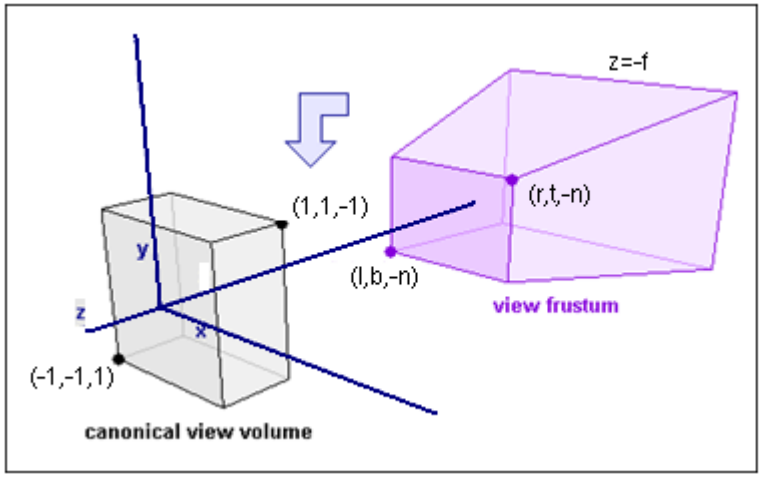


$p' = T_1\, p$
$p'' = T_1\, R\, p$
$p''' = T_1\, R\, T_2\, p$

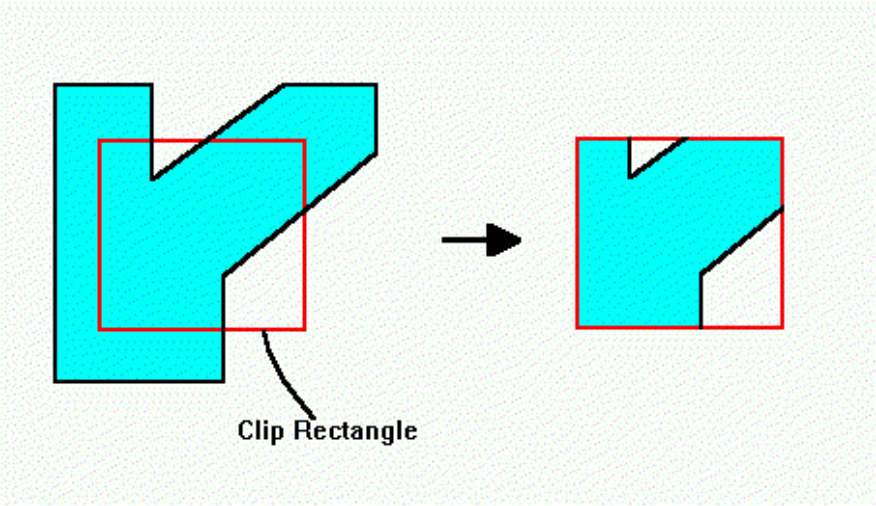
- Concatenate local transformation matrices from left to right
- Can obtain the local – world transformation matrix
- p',p'',p''' are the world coordinates of p after each transformation

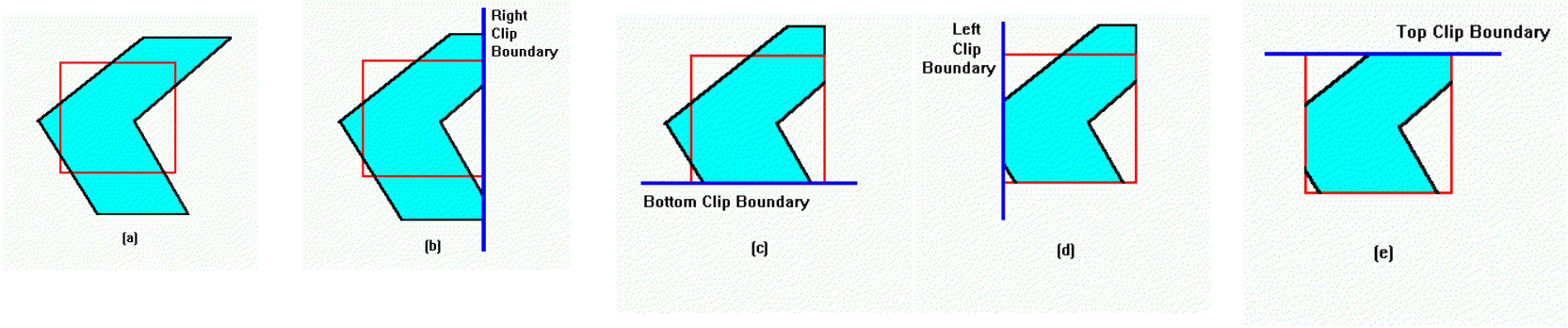
Projection


screen


z=-f
(1,1,-1)
(r,t,-n)
(l,b,-n)
view frustum
y
z
x
(-1,-1,1)
canonical view volume

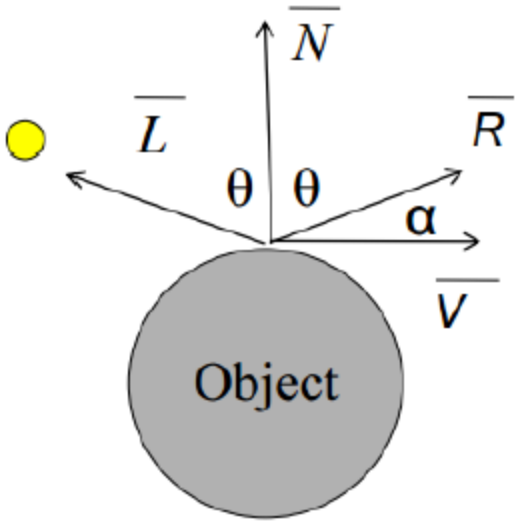
Clipping


Clip Rectangle


(a)
Right
Clip
Boundary
(b)
Bottom Clip Boundary
(c)
Left
Clip
Boundary
(d)
Top Clip Boundary
(e)

Illumination and Shading

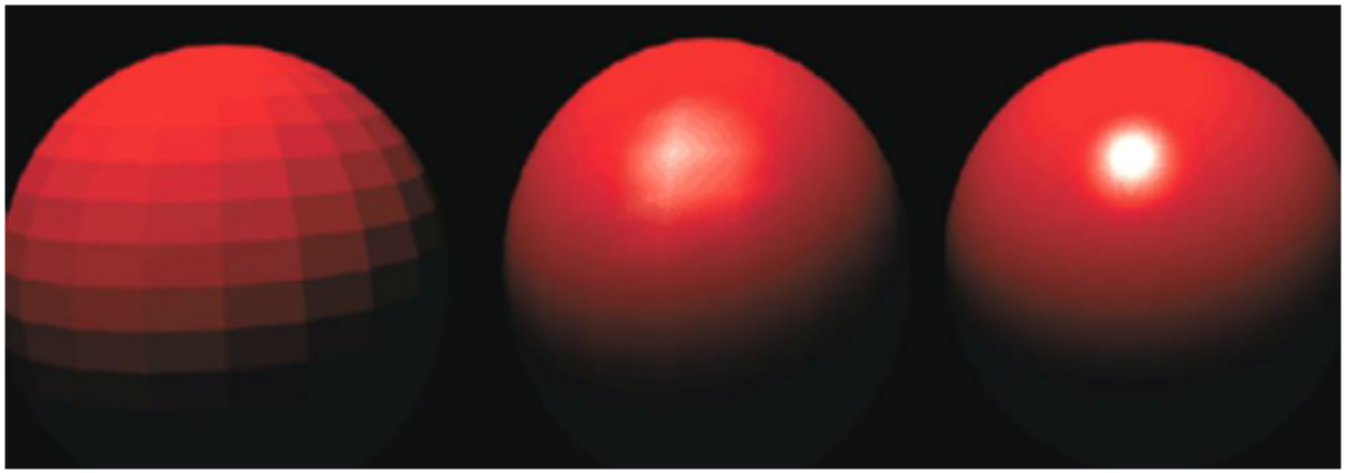
Phong Illumination Model



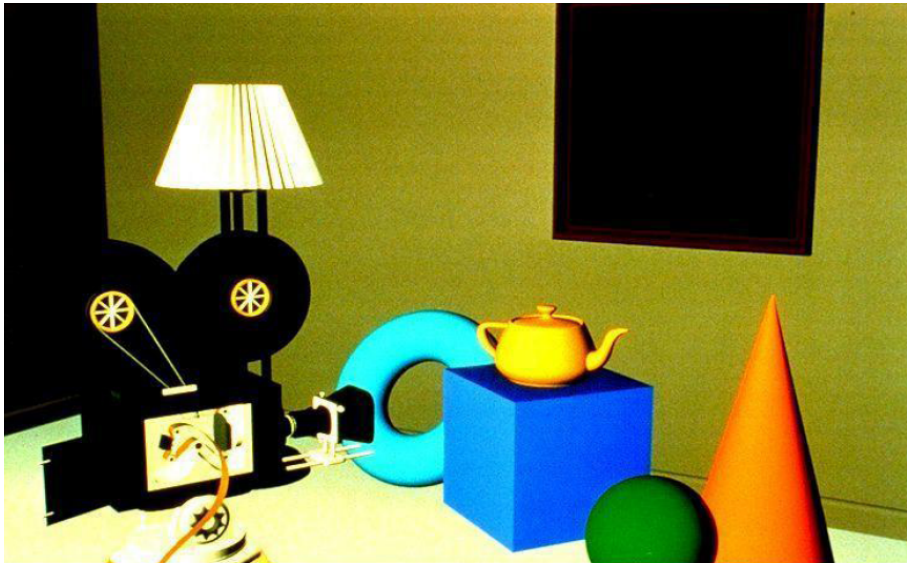
$$I_\lambda = I_a k_a + \sum_{p=1}^{lights} I_p [k_d (\overline{N} \bullet \overline{L}) + k_s (\overline{V} \bullet \overline{R})^n]$$

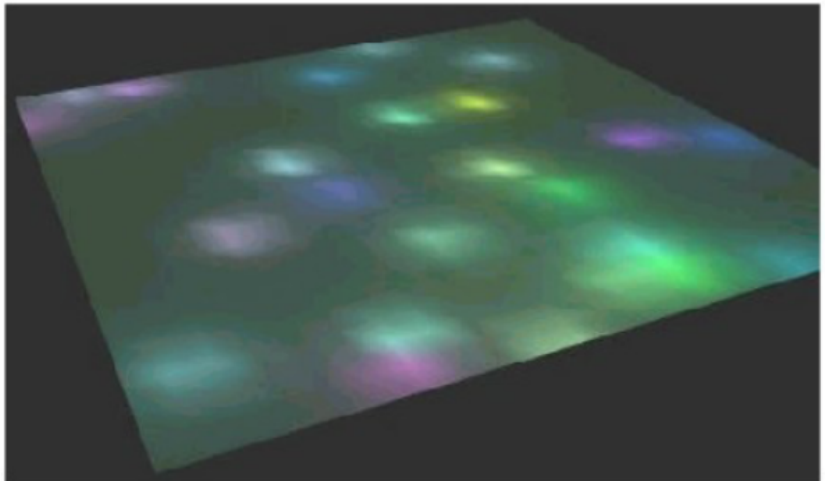
$\cos\theta$ $\cos\alpha$

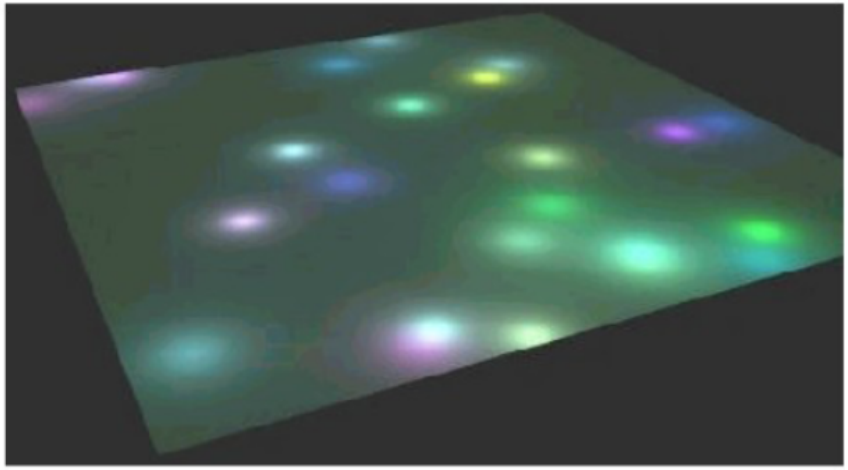
Flat shading, Gouraud shading, Phong shading



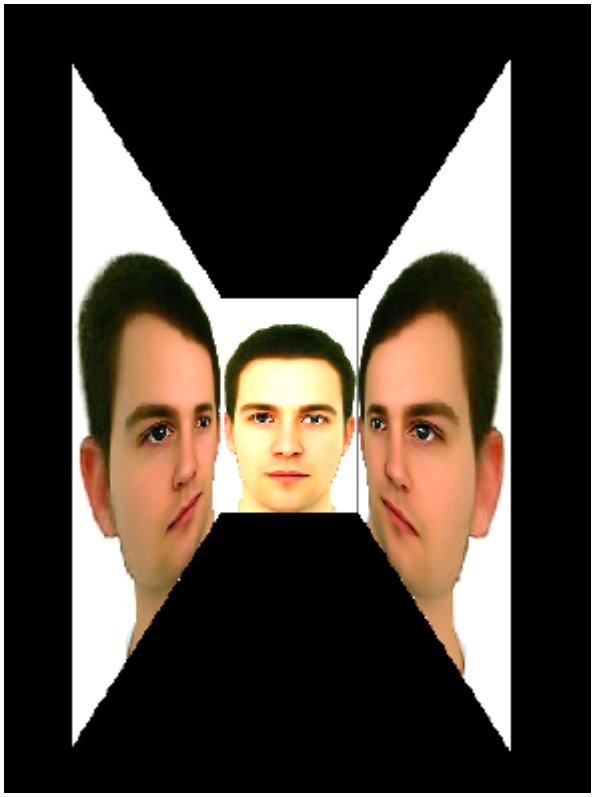
Illumination and Shading



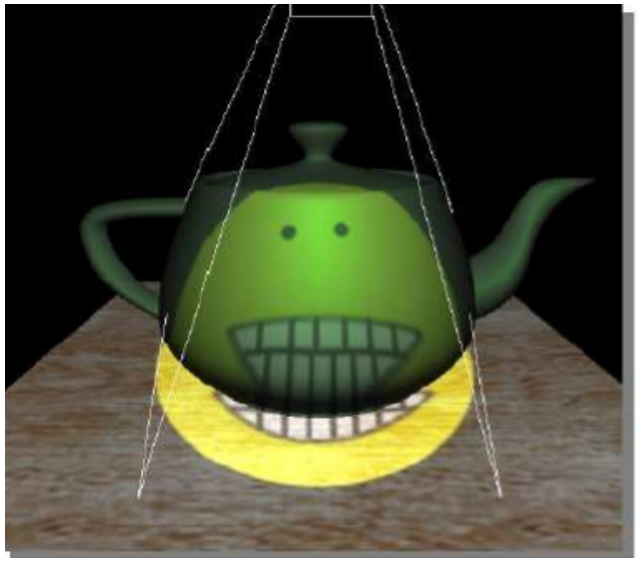


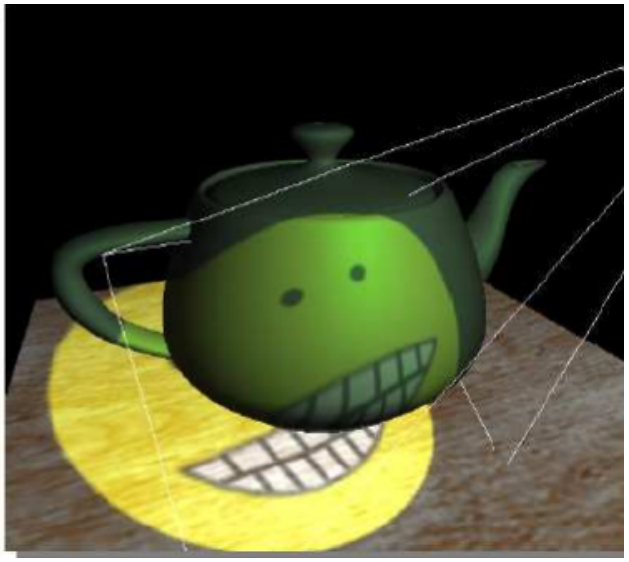


Texture Mapping



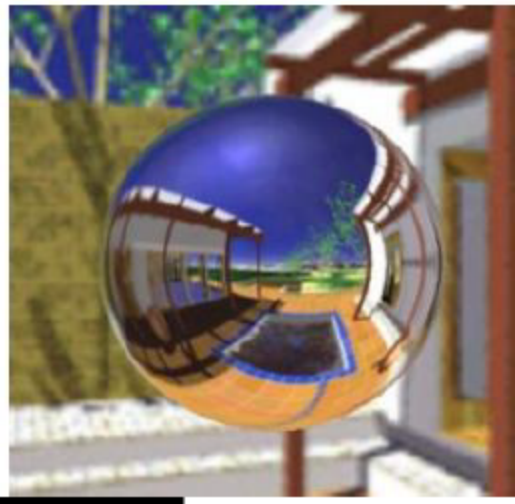
Texture Mapping

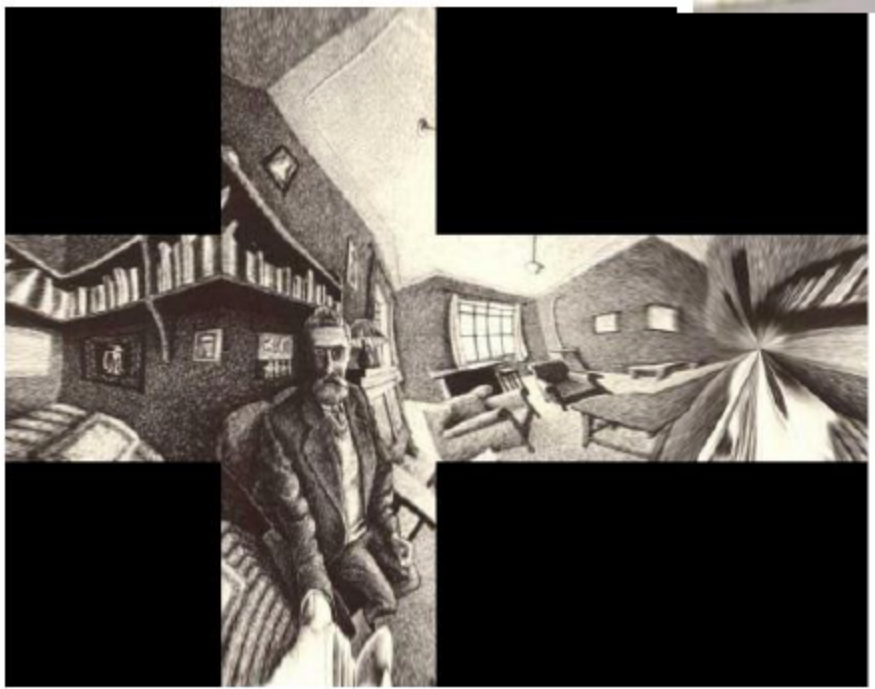




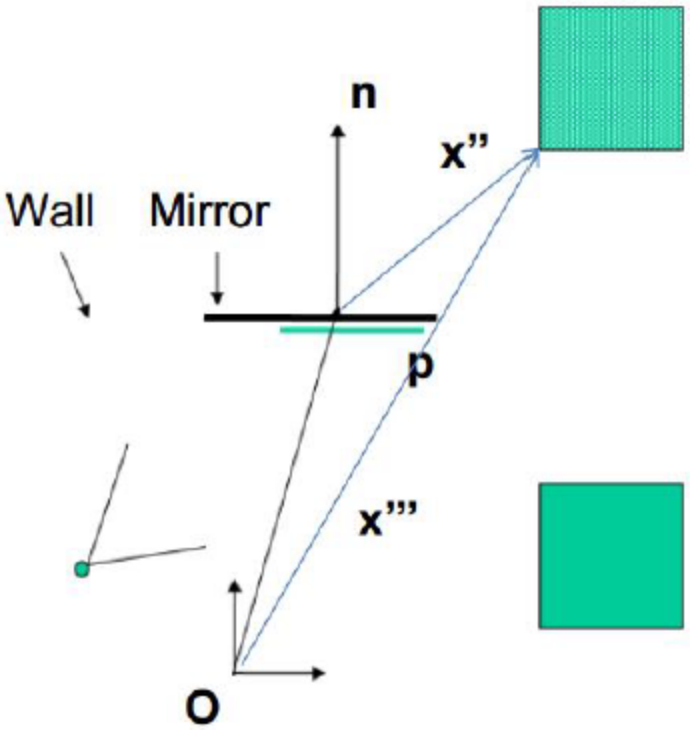
What is the mapping function?

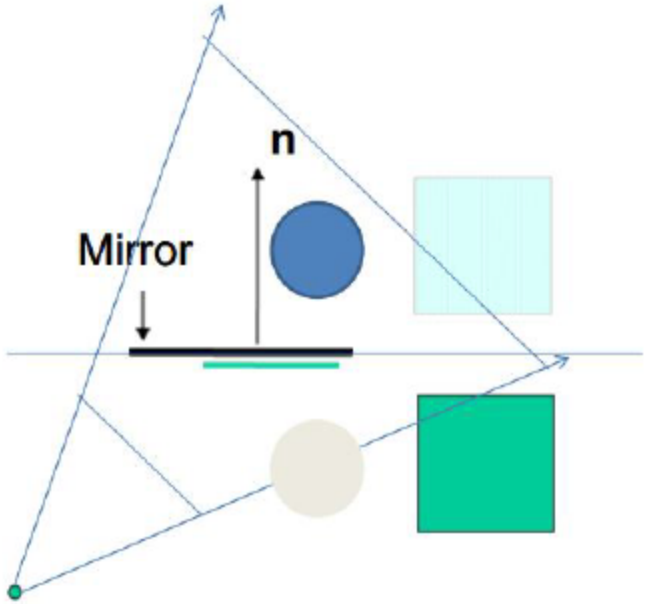
Environment Mapping

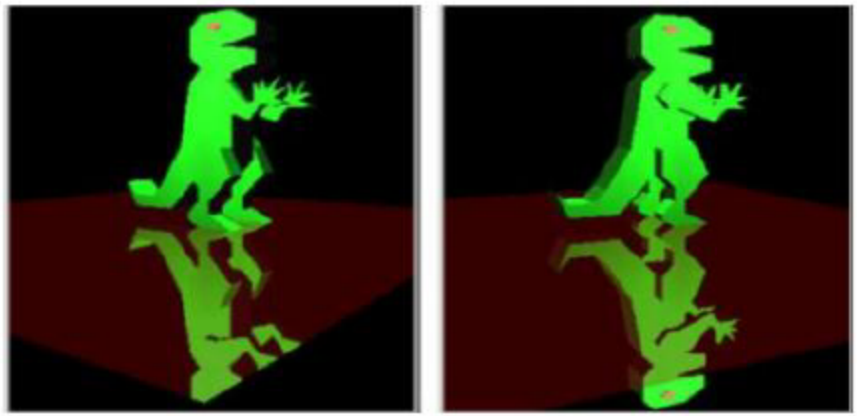




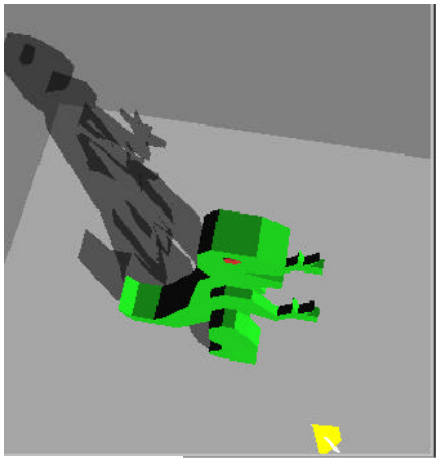
Mirrored world

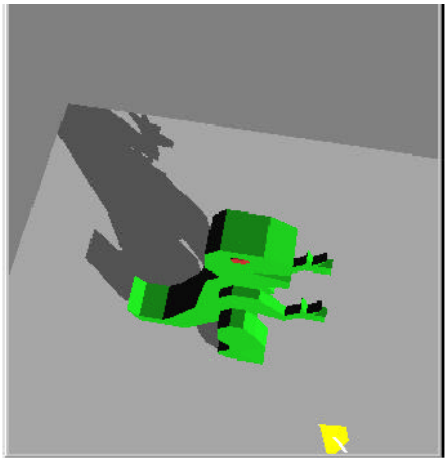


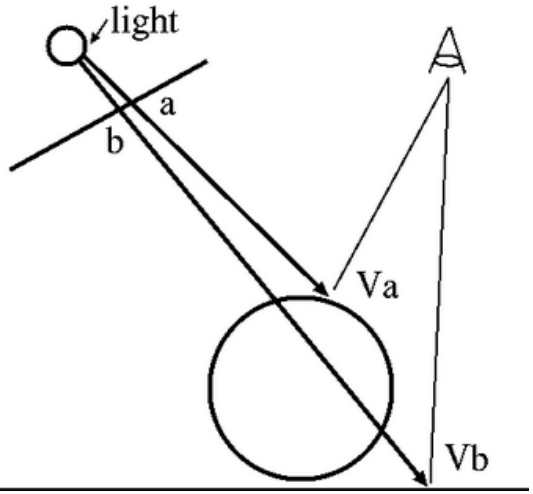


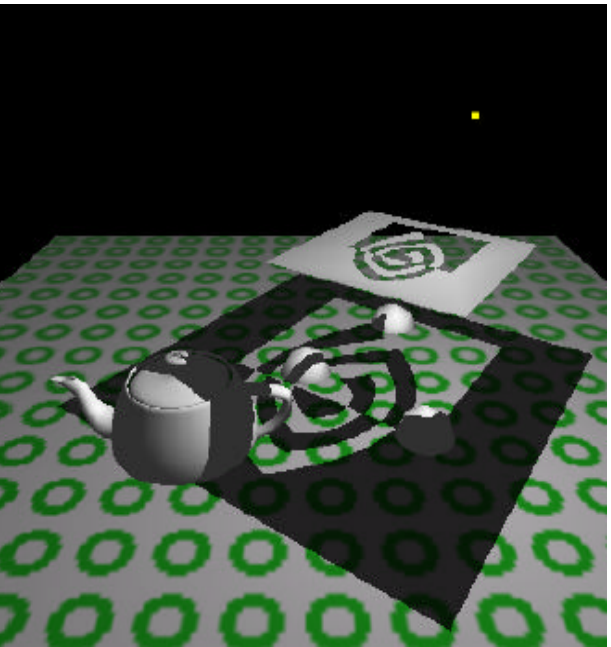


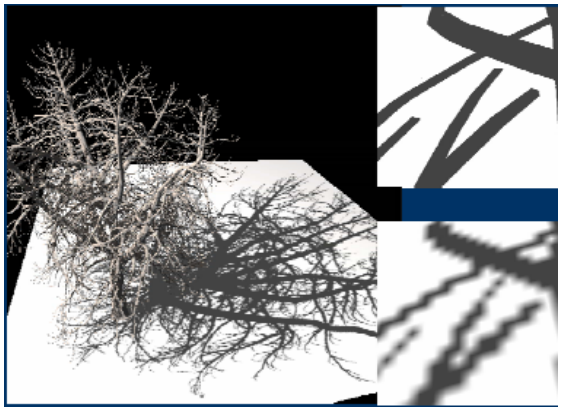
Shadows

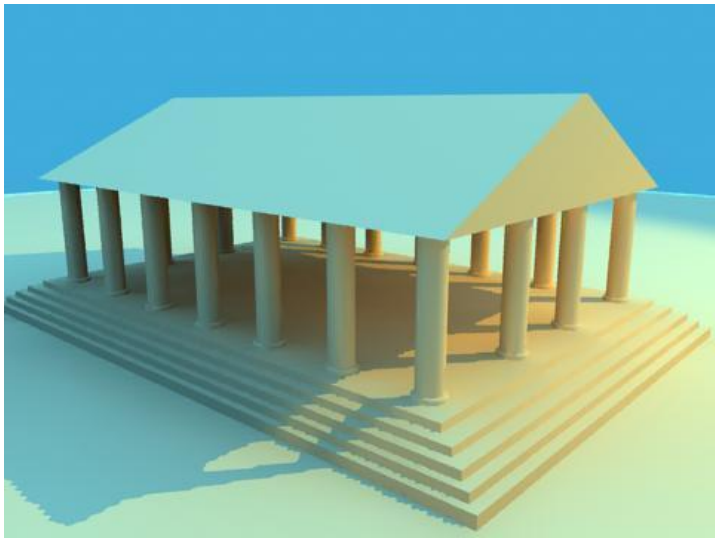




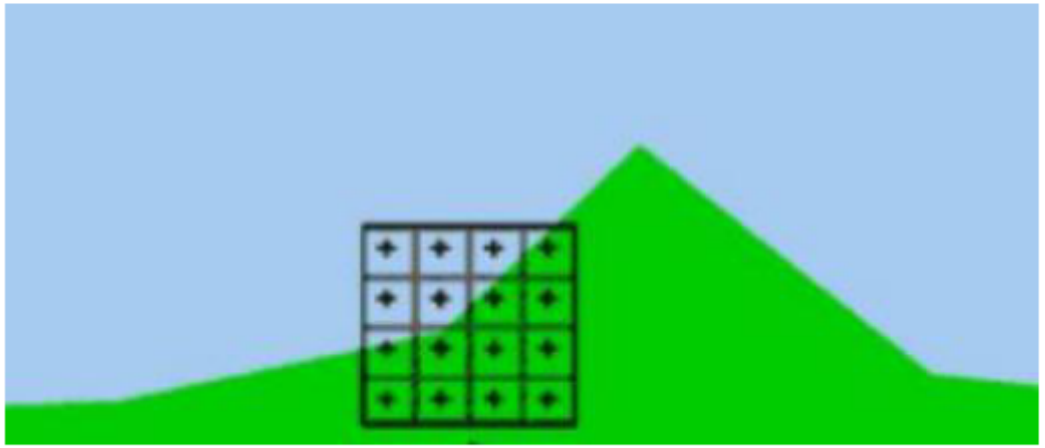


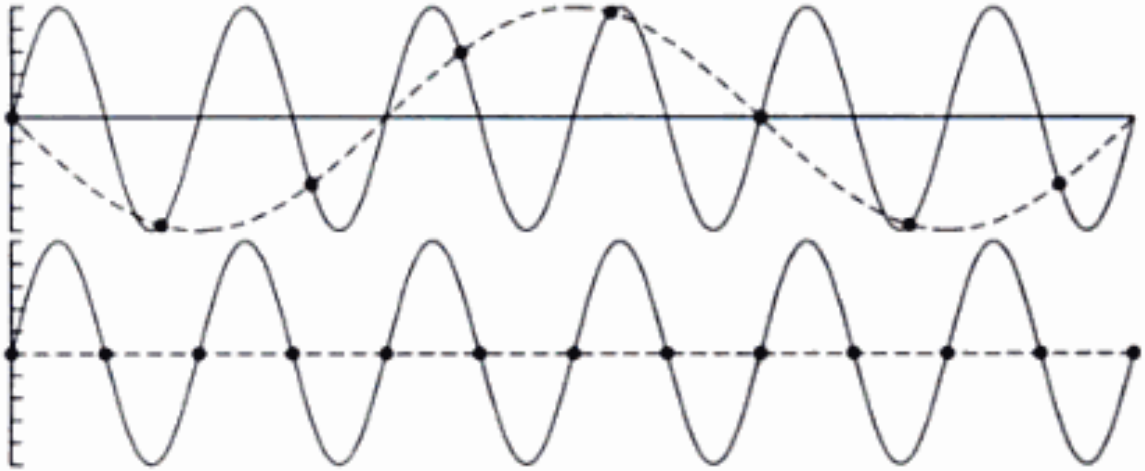


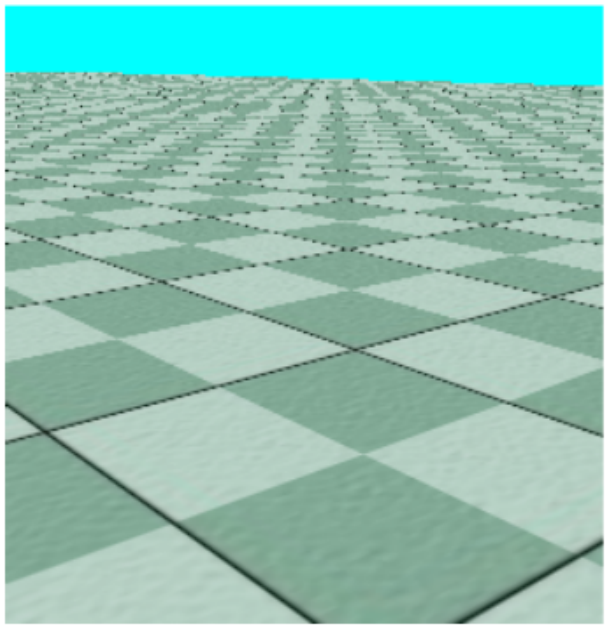


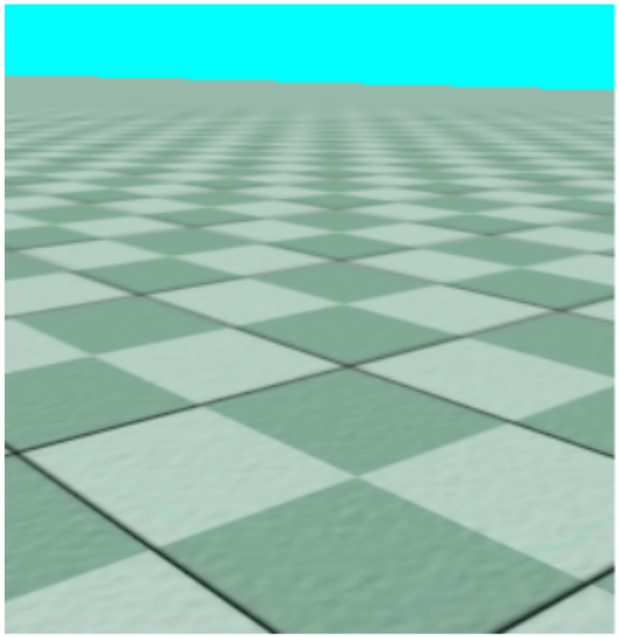


Anti-aliasing

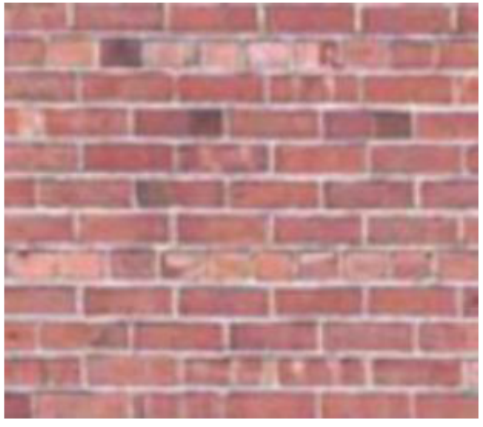




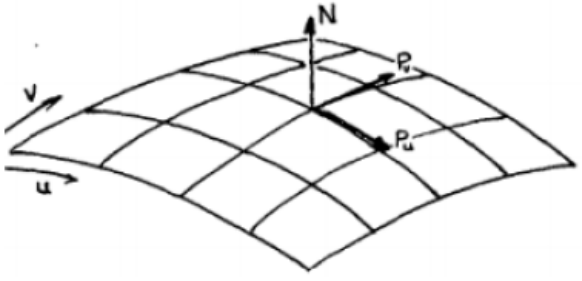


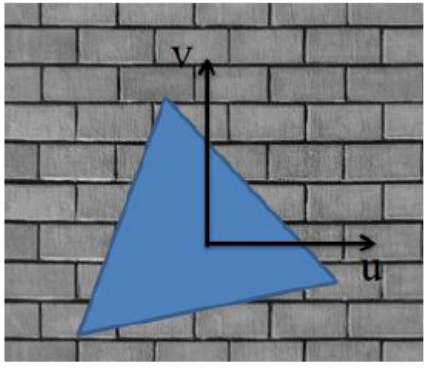


Bump mapping



$$\mathbf{n}' = \mathbf{n} + \frac{F_u(\mathbf{n} \times \mathbf{P_v}) - F_v(\mathbf{n} \times \mathbf{P_u})}{||\mathbf{n}||}$$

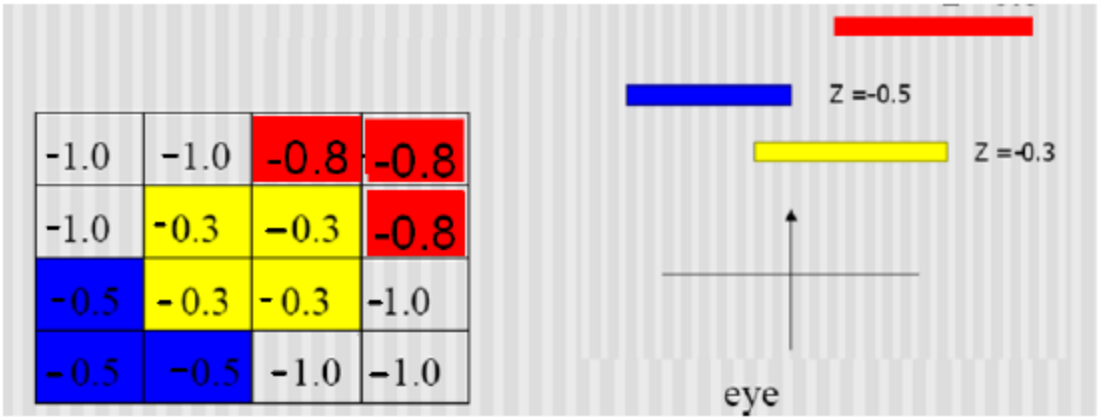


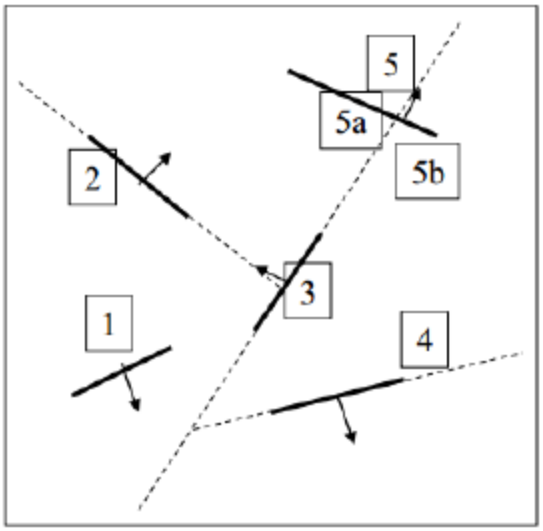


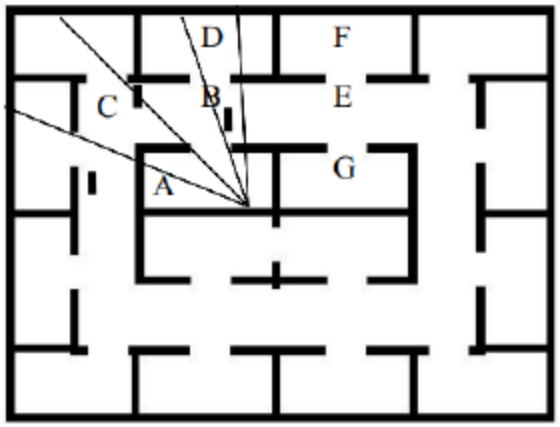
$$\begin{aligned} \frac{\partial P}{\partial x} &= \frac{\partial P}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial P}{\partial v}\frac{\partial v}{\partial x} \\ \frac{\partial P}{\partial y} &= \frac{\partial P}{\partial u}\frac{\partial u}{\partial y} + \frac{\partial P}{\partial v}\frac{\partial v}{\partial y}. \end{aligned}$$

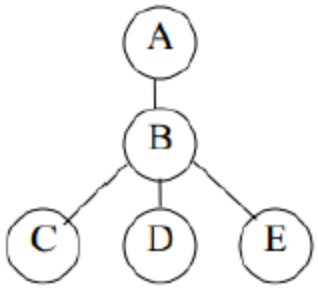
Hidden Surface Removal

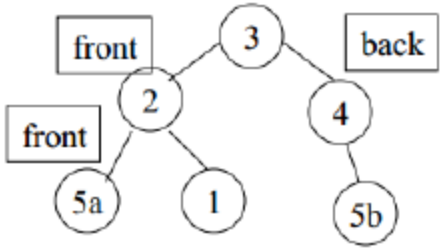
Z-buffer, BSP trees, Portal culling

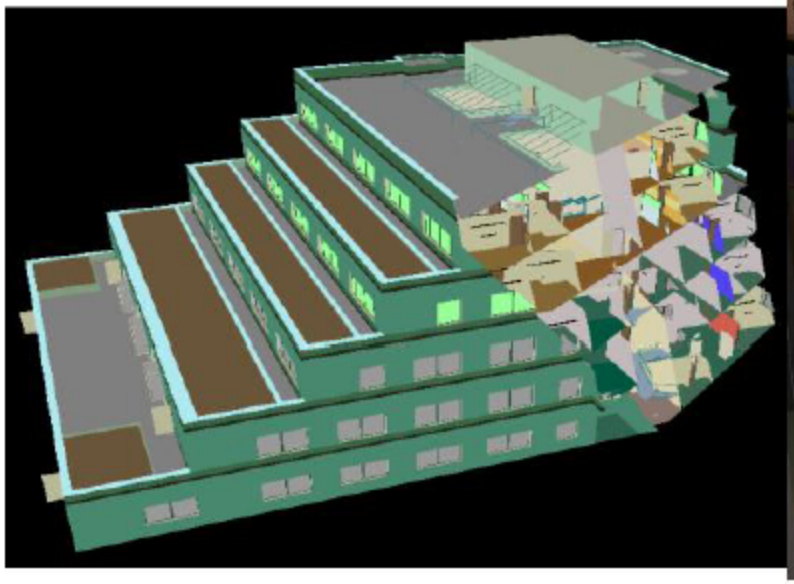








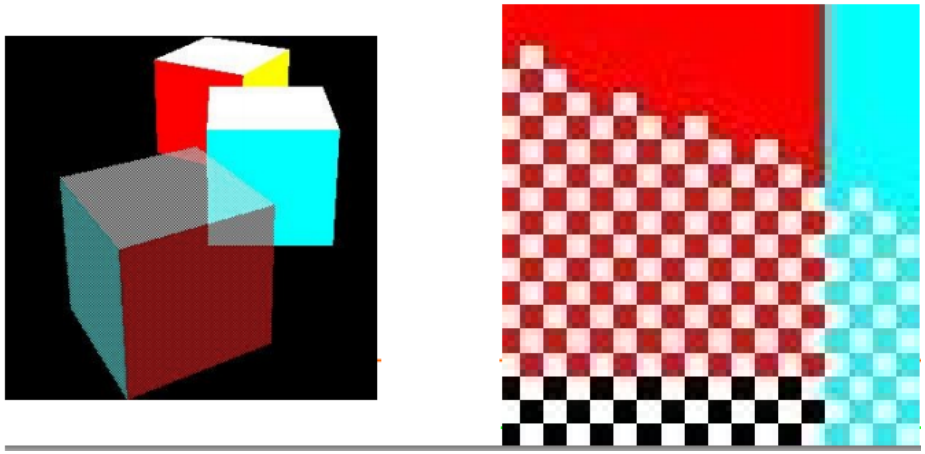


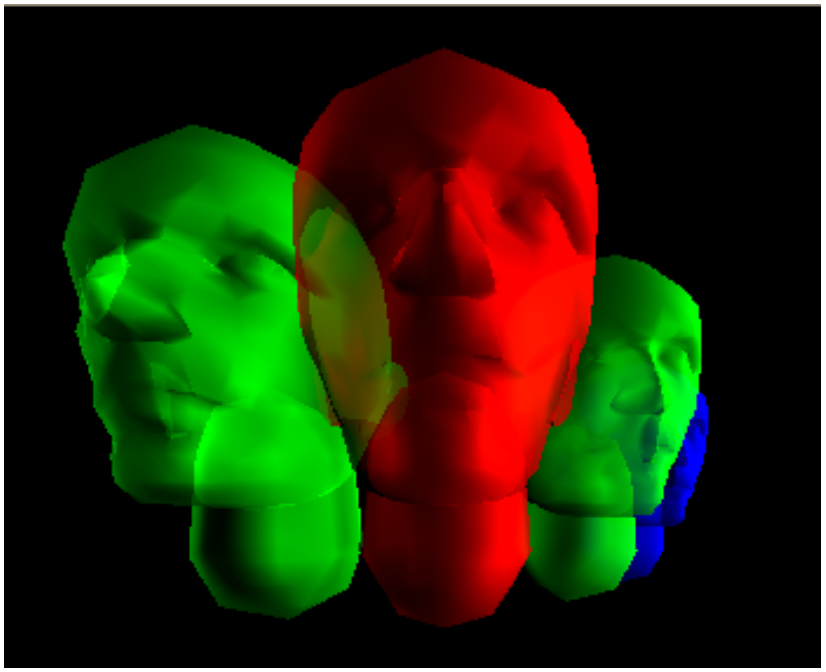


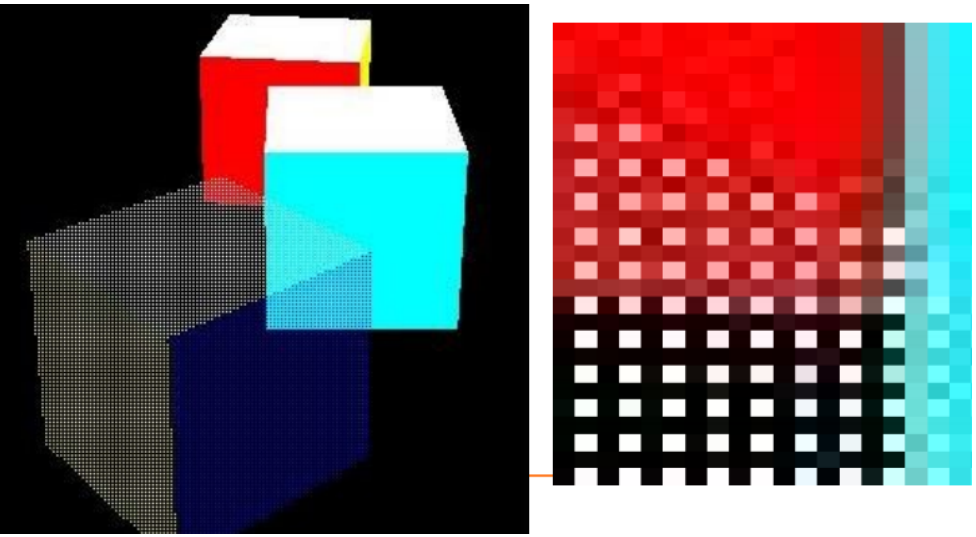

100
12

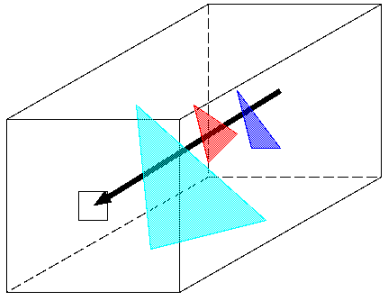
Transparency

alpha = 0.5

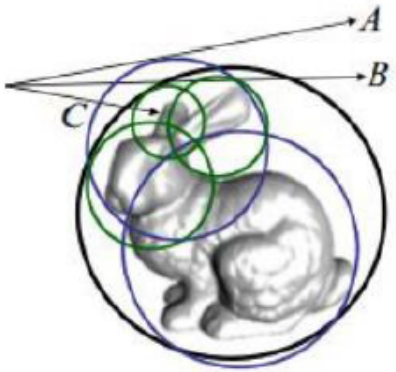


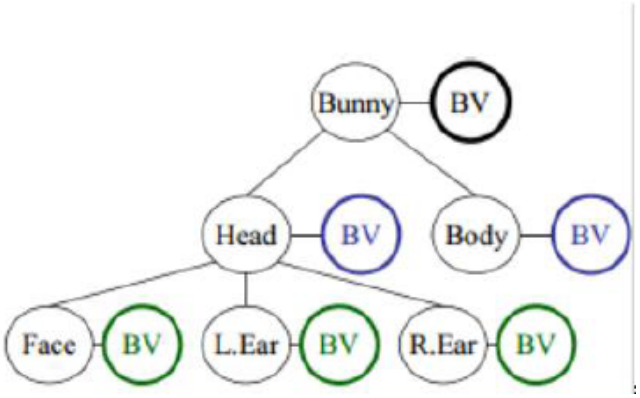


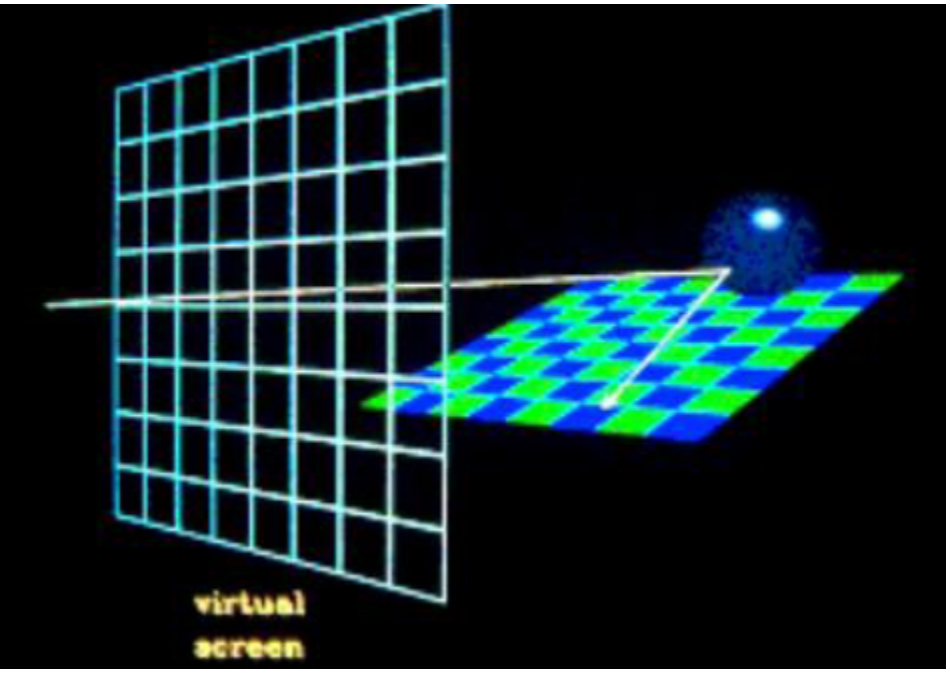




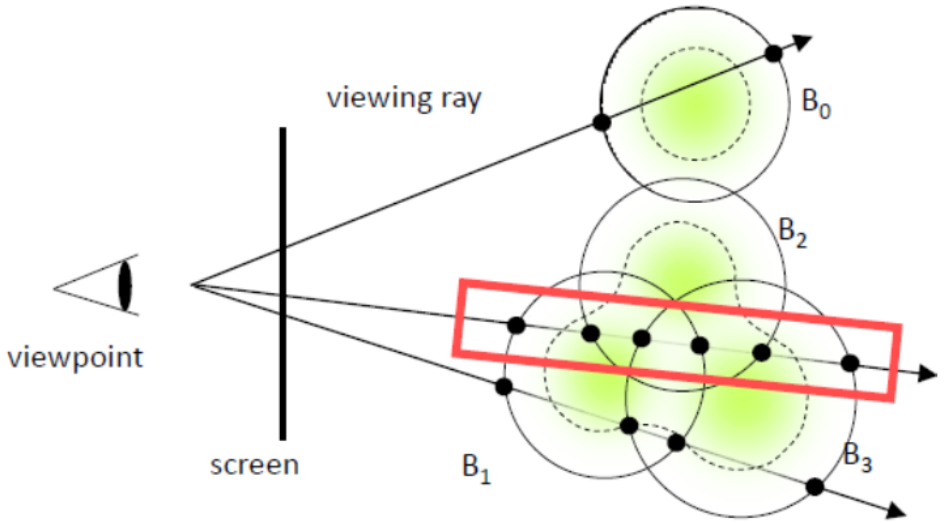
Ray Tracing







- How to make a bounding sphere hierarchy?


viewing ray
B_0
B_2
viewpoint
screen
B_1
B_3

Light Transport Notations

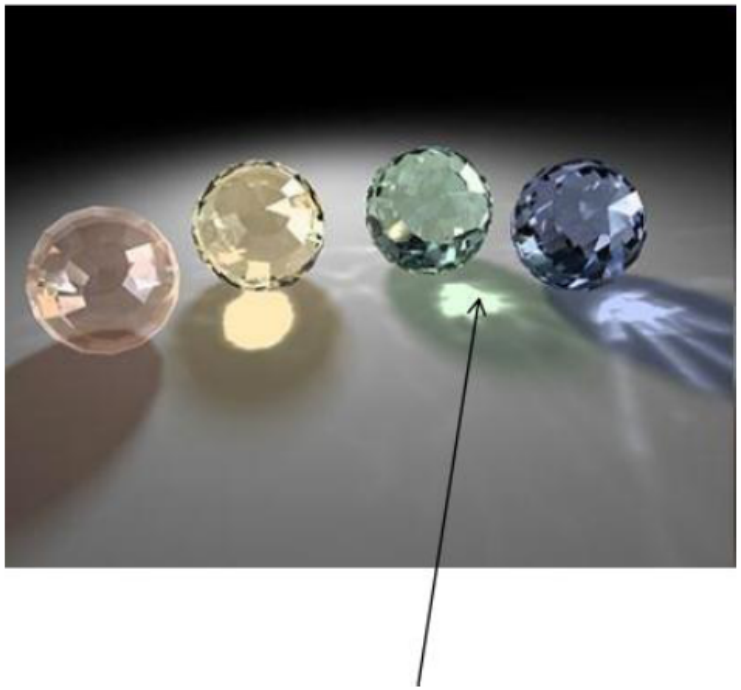
L a light source

E the eye

S a specular reflection

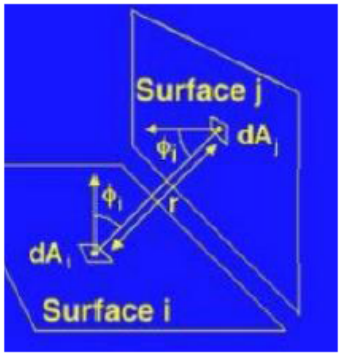
D a diffuse reflection

LDDE



LSDE

Radiosity

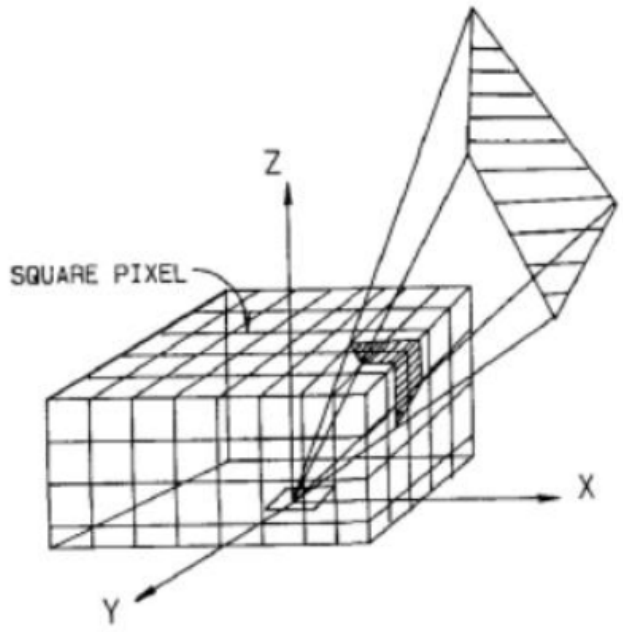


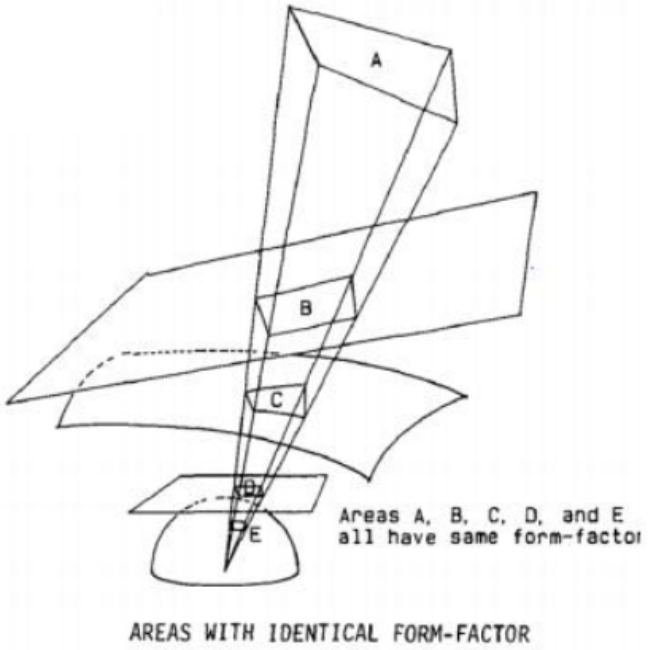
$$B_j = E_j + \rho_j \sum_{i=1}^{N} B_i F_{i,j}$$

(a)

$$\begin{pmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \dots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \dots & -\rho_2 F_{2N} \\ \vdots & \vdots & \dots & \vdots \\ -\rho_N F_{N1} & -\rho_N F_{N2} & \dots & 1-\rho_N F_{NN} \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{pmatrix}$$

Hemicube





Projection, Z-buffer, symmetry

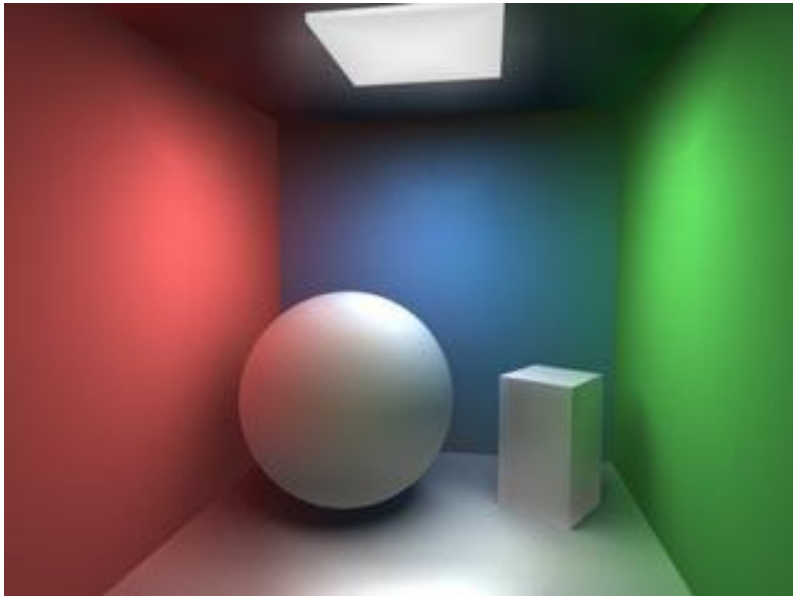
For computing the radiosity, you use an iterative approach

$$B_j = E_j + \rho_j \sum_{i=1}^{N} B_i F_{i,j}$$

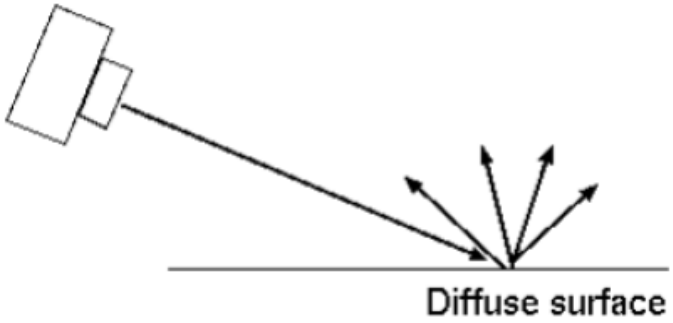
$$B_j^0 = E_j$$

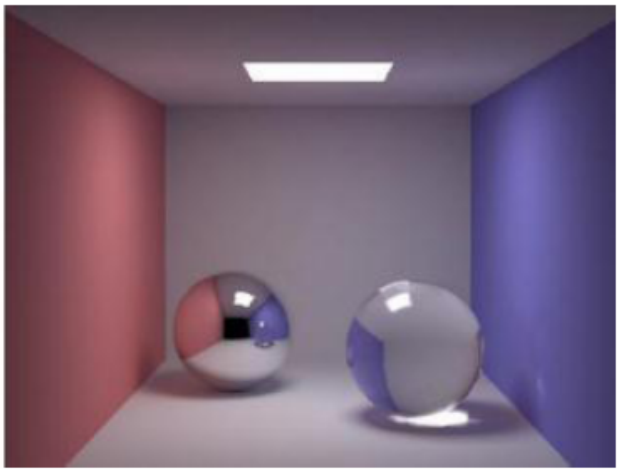
$$B_j^{k+1} = \rho_j \sum_{i=1}^{N} B_i^k F_{i,j}$$

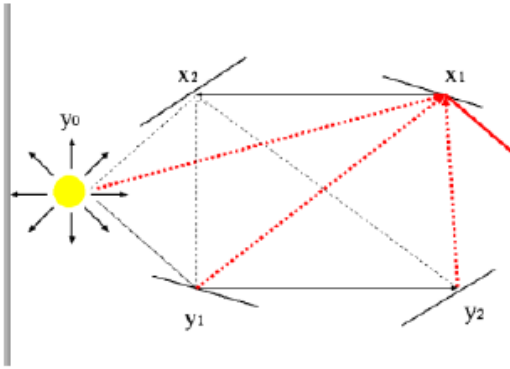
$$B = \sum_k B^k$$



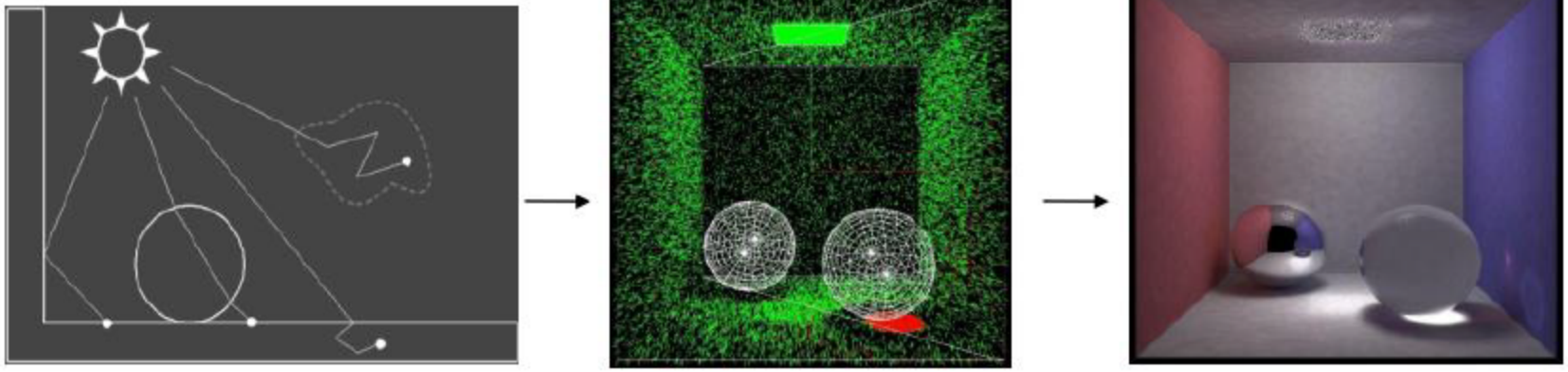
Path Tracing

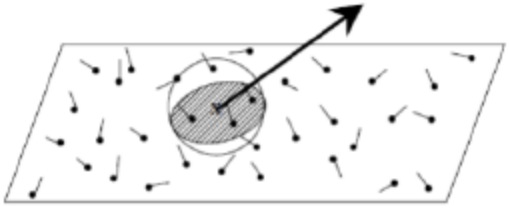


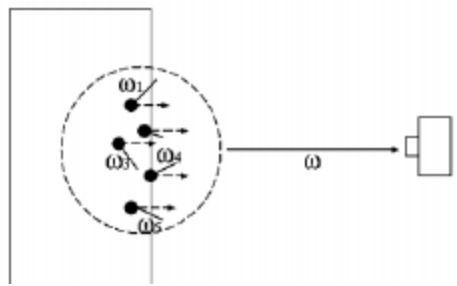




Photon Mapping

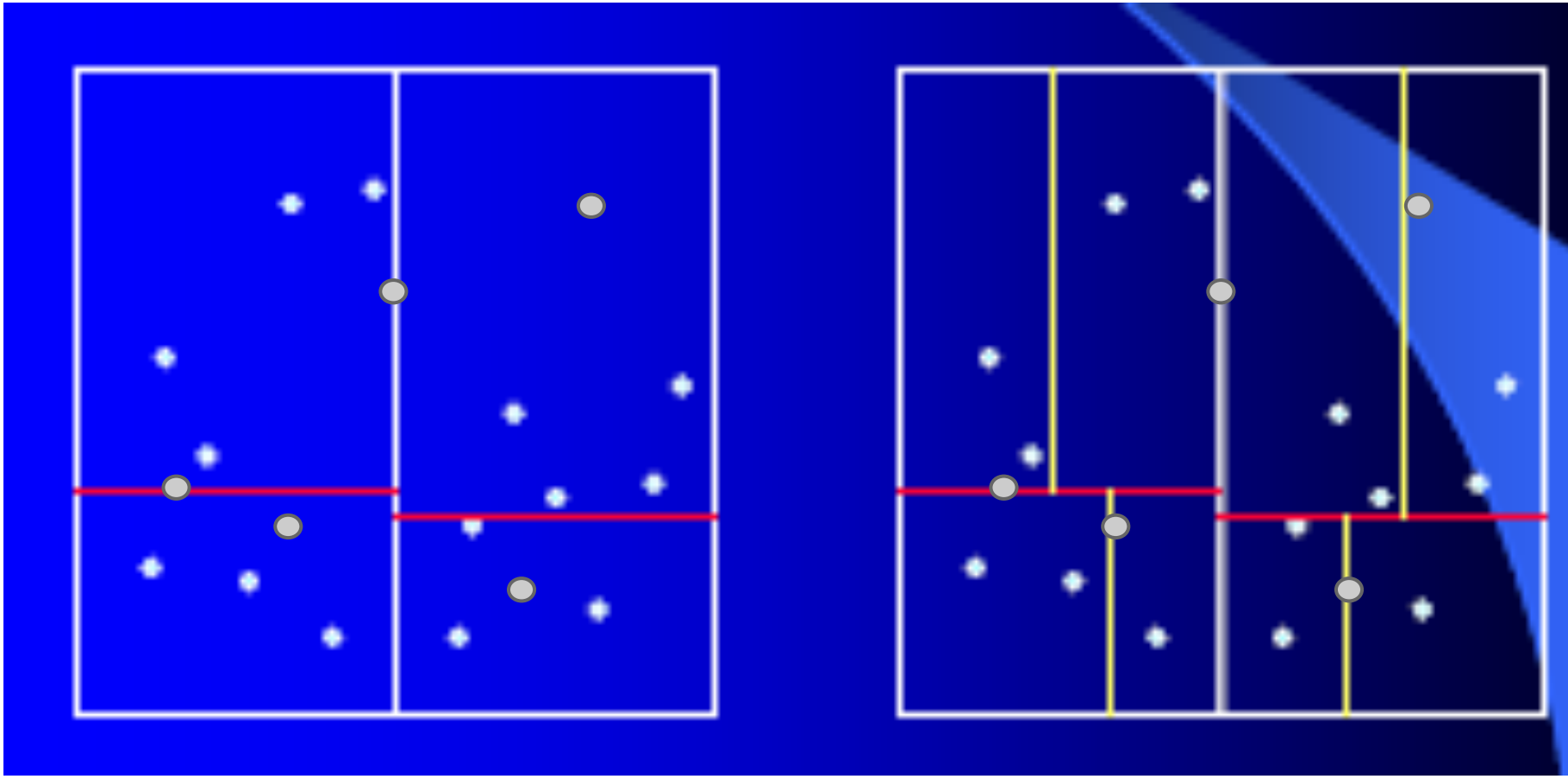




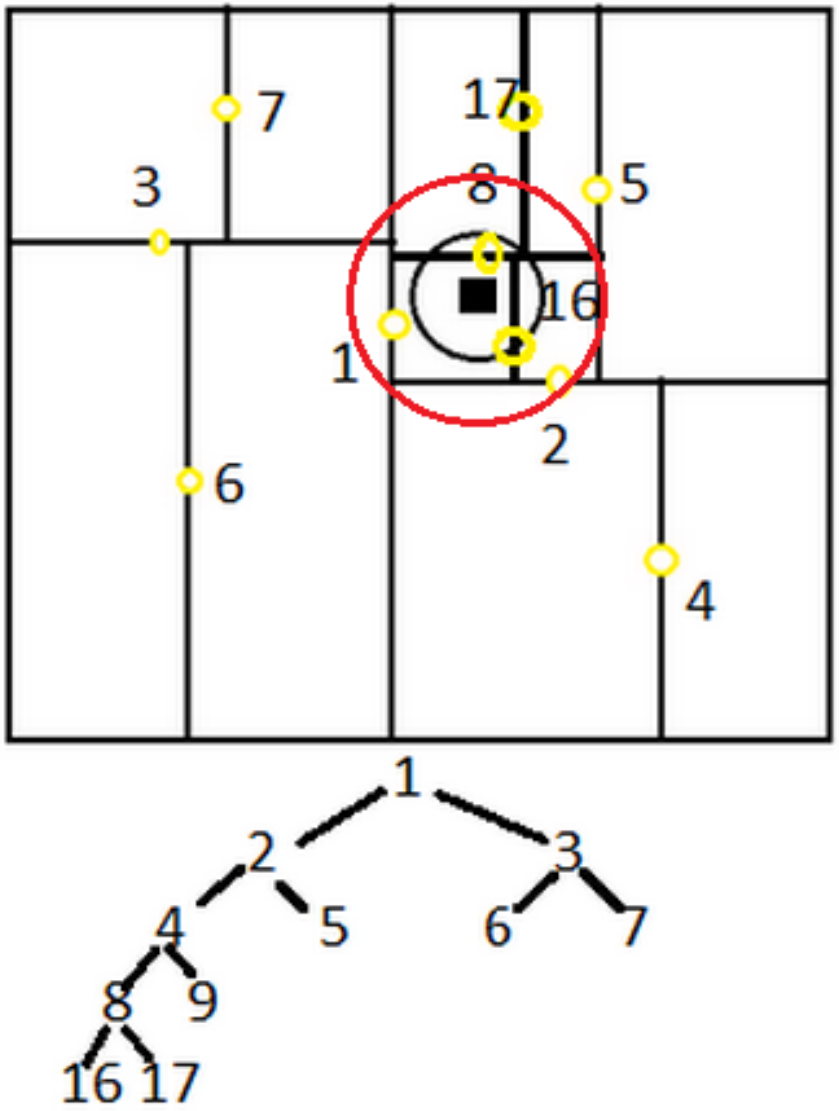


$$L_r(x,\vec{\omega}) = \sum_{p=1}^{N} f_r(x,\overrightarrow{\omega_p},\vec{\omega}) \frac{\Delta\Phi_p(x,\overrightarrow{\omega_p})}{\Delta A}$$

KD-Tree



KD-tree



- For 2 nearest neighbours, what is the order that the photons are added in the KD-tree?
- What are the photons in the heap when the algorithm ends?

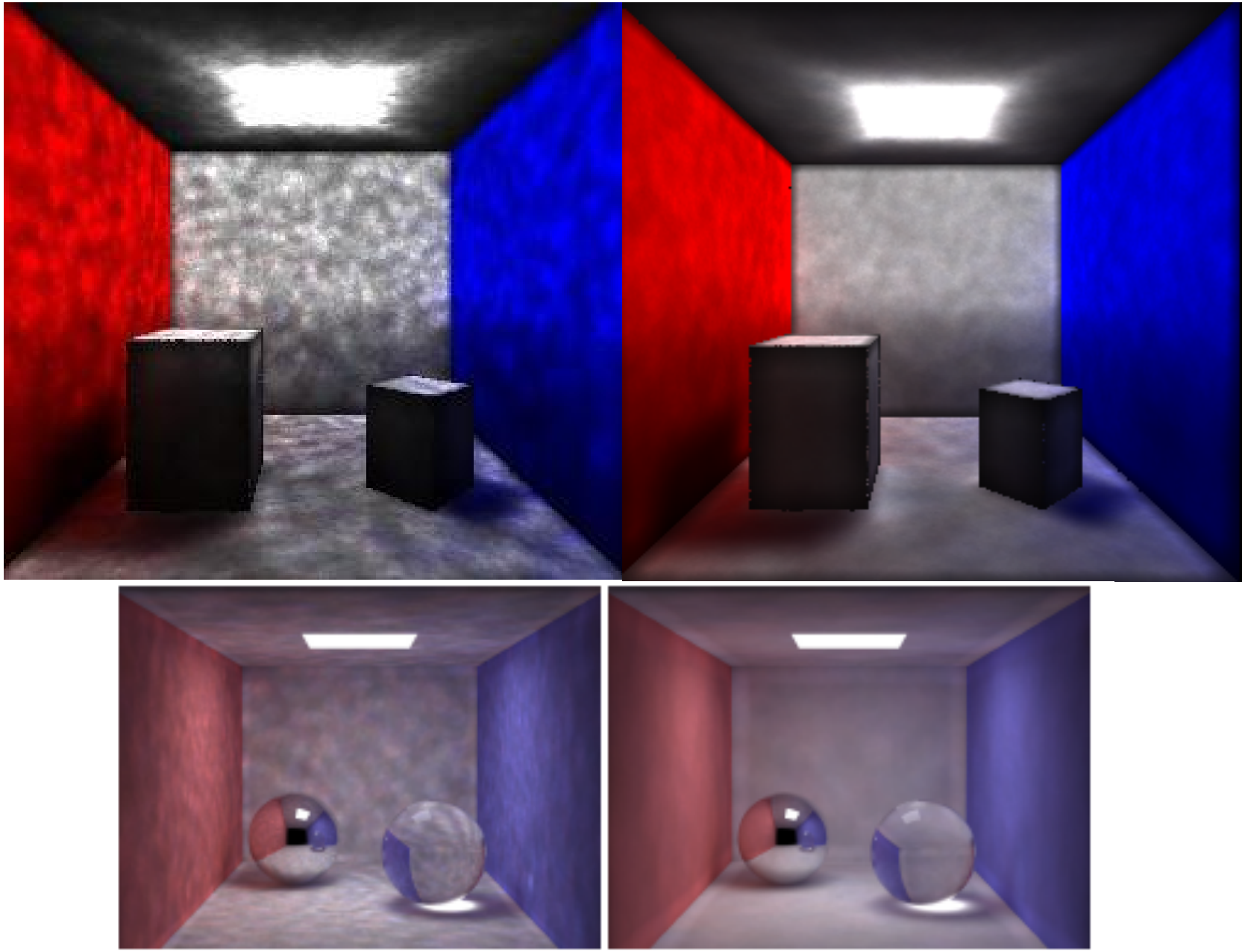
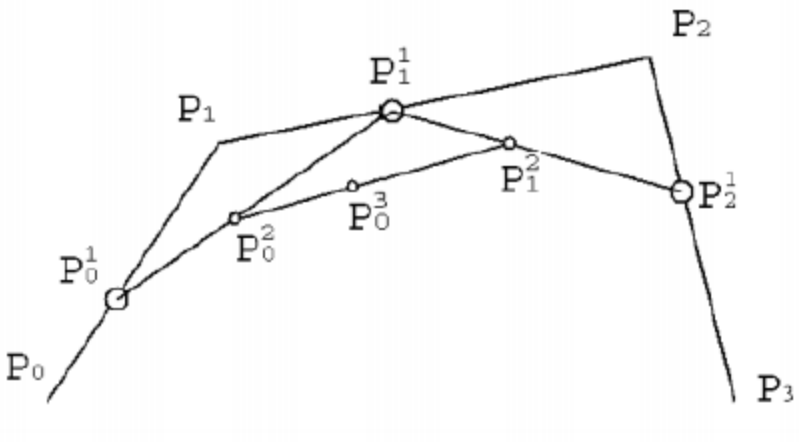
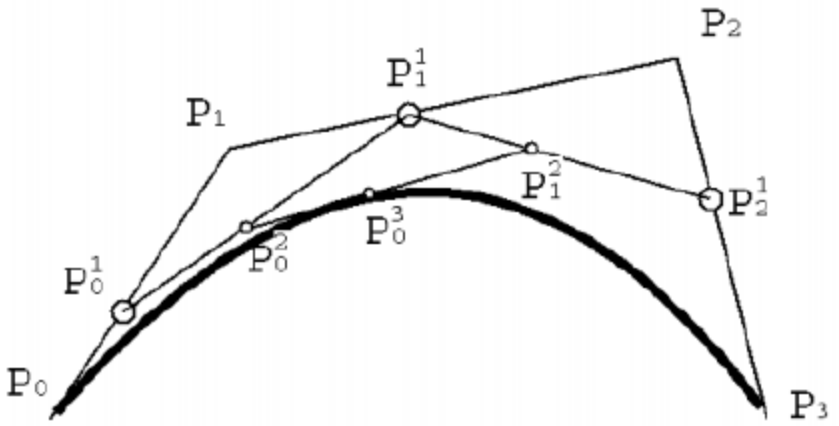
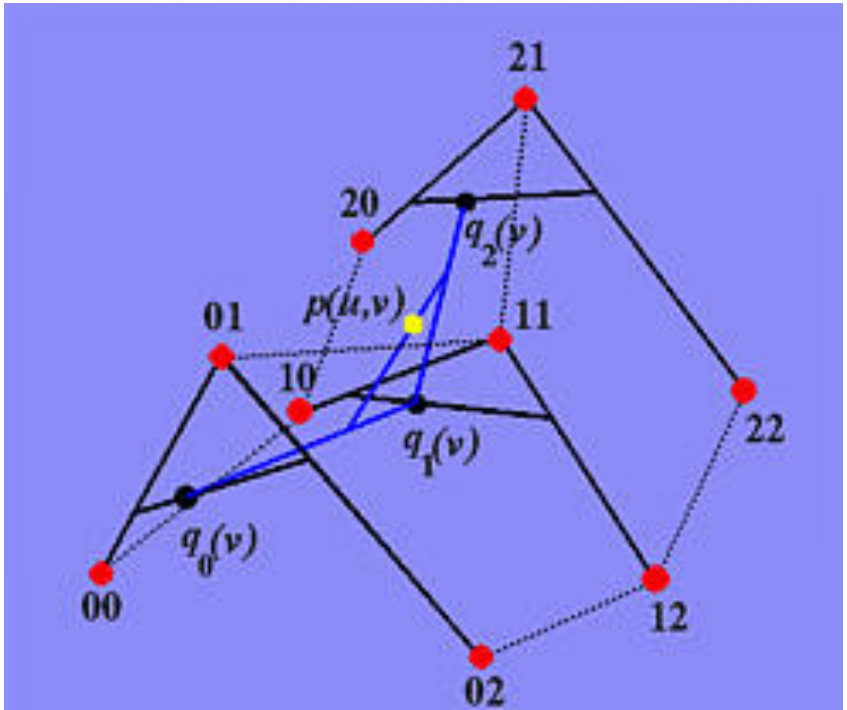


Figure 20: Global photon map radiance estimates visualized directly using 100 photons (left) and 500 photons (right) in the radiance estimate.

Parametric Curves and Surfaces



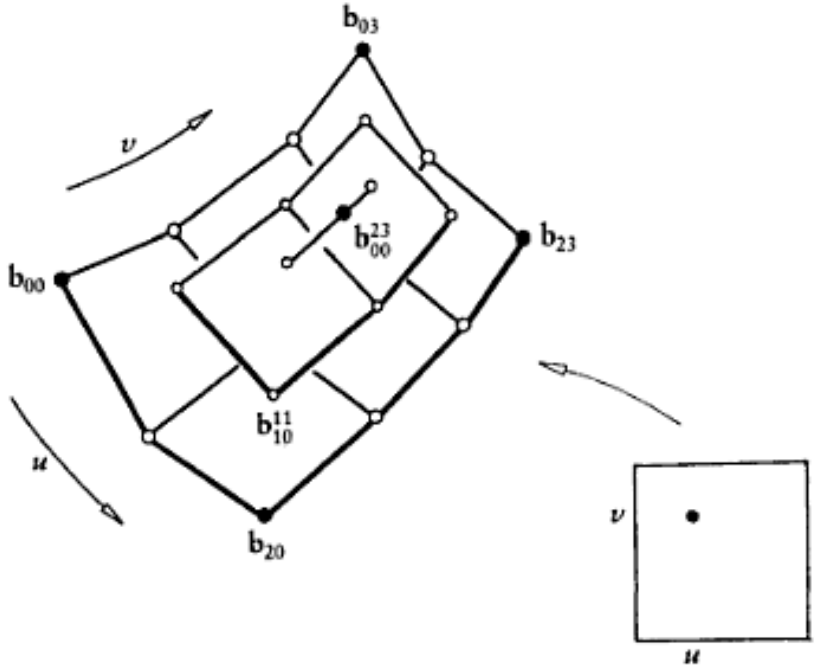




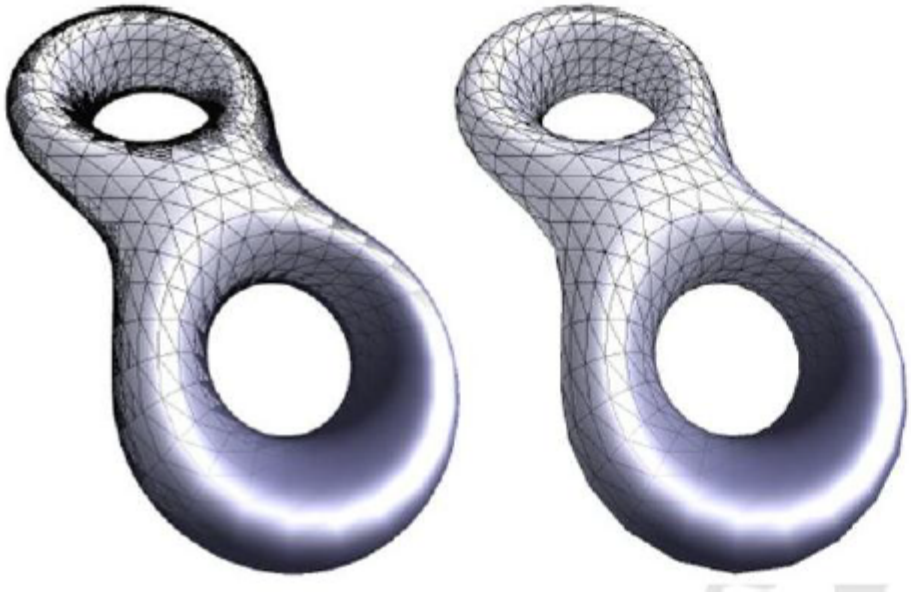
http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/surface/bezier-de-casteljau.html

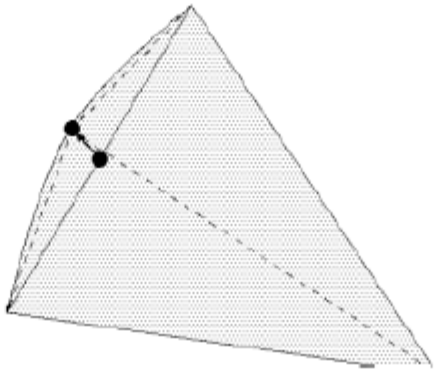
Direct de Casteljau

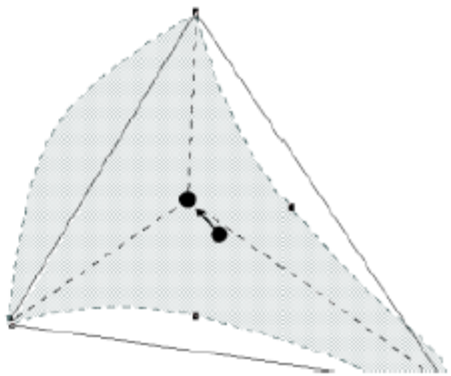
- Given (u,v), in every adjacent quad area, sample a point by bilinear interpolation
- Iterate this process



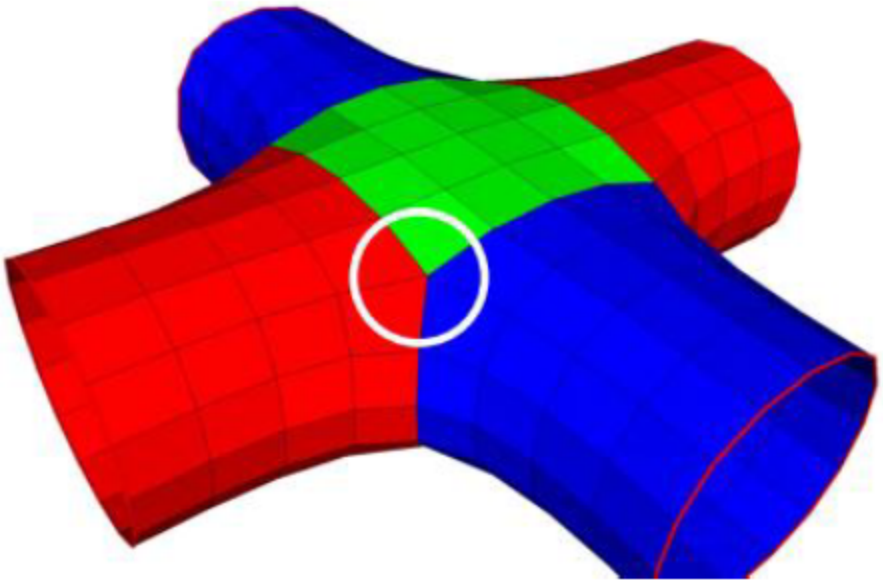
Adaptive Tesselation, On-the-fly Tesselation

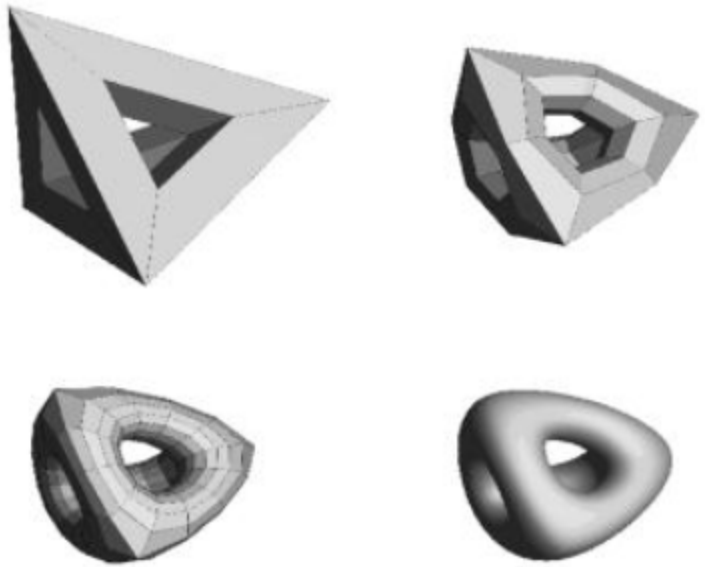






NURBS, Subdivision Surface





Good Luck!!