From Graphics to Visualization

outline

- Introduction
- Light Sources
- Surface Lighting Effects
- Basic (Local) Illumination Models
- Polygon-Rendering Methods
- Texture Mapping
- Transparency and Blending
- Visualization Pipeline

Introduction

- Illumination Model (Lighting/Shading Model)

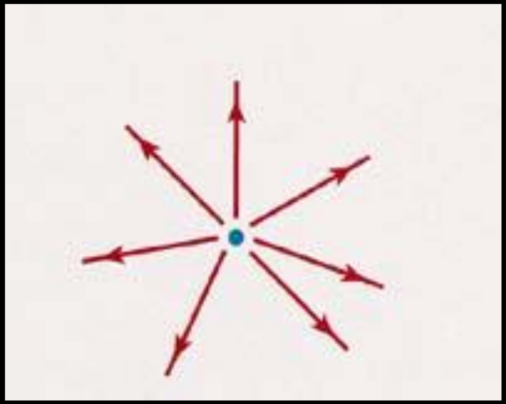
  Calculation of color on an illuminated position on the surface of an object

- Surface Rendering

  A procedure for applying a lighting model to obtain pixels colors for all projected surface positions

Light Sources

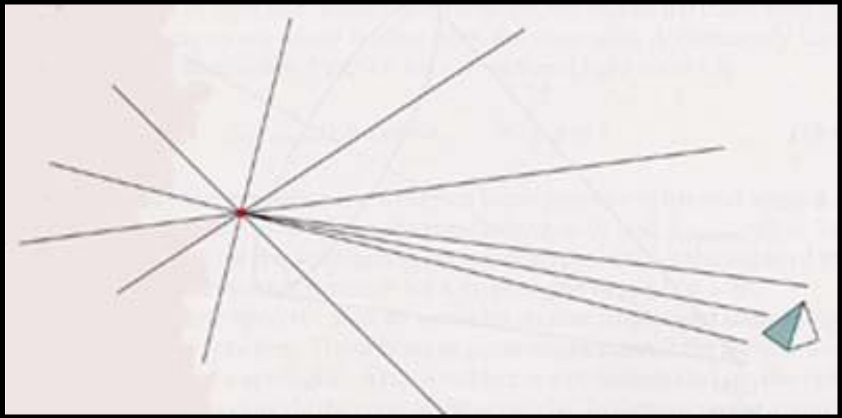
- **Point Source**



Diverging ray paths from a point light source

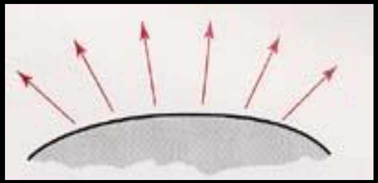
Light Sources

- **Distributed Light Source**

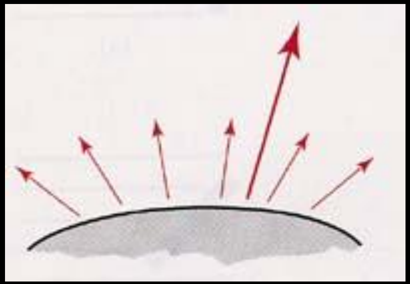


Light rays from an infinitely distant light source illuminate an object along nearly parallel light paths

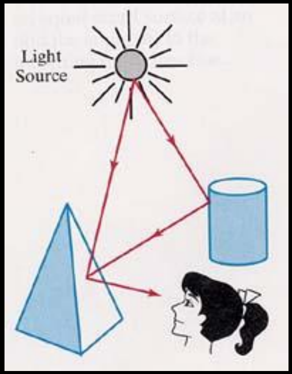
Surface Lighting Effects



Diffuse reflections from a surface (dull/rough surface)



(shiny surface) Specular reflection superimposed on diffuse reflection vectors



(Global Illumination) Surface lighting effects are produced by a combination of illumination from light sources and reflection from other surfaces.

Illumination Models

Rendering methods differ in approximating lighting effects

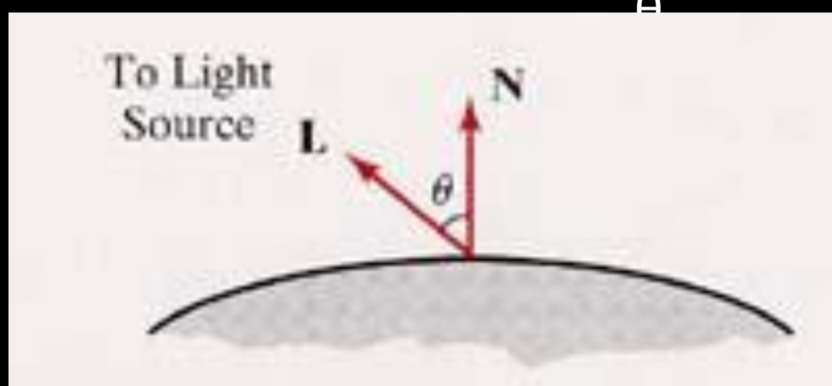
- **Global illumination**: ray tracing, accurate by computationally expensive
- **Local illumination**: relate the illumination of a given scene point directly to the light set, not to any other scene points

Basic (Local)Illumination Models

1) Ambient Light

2) Diffuse Reflection

θ



Angle of incidence θ between the unit light-source direction vector L and the unit normal vector N at a surface position.

Basic (Local)Illumination Models

2) Diffuse Reflection

$I_{l,diff} = K_d I_l \cos\theta$

$I_{l,diff} = K_d I_l (N \cdot L)$

k_d : **diffuse-reflection coefficient**, or **diffuse reflectivity.**

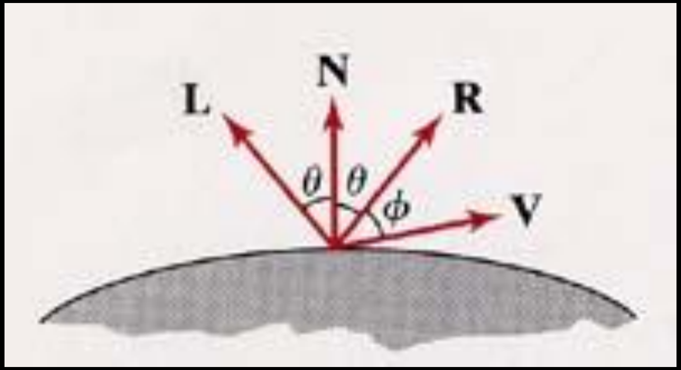
- **Lambertian reflectors**
- **Lambert's cosine law**

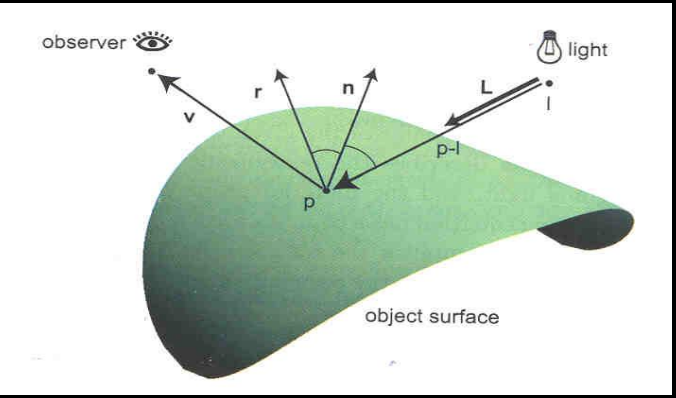
– **Total Diffuse-reflection of a single point-source illumination**

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

Basic (Local)Illumination Models

3) Specular Reflection and *Phong* Model





Specular reflection angle equals angle of incidence θ

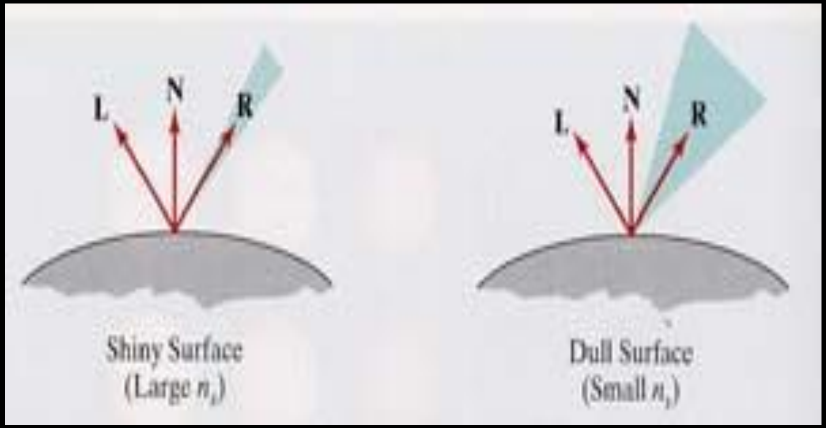
– **Phong Specular-Reflection Model (Phong Model)**

$$I_{l,spec} = W(\theta) I_l \cos^{n_s} \Phi \tag{2-4}$$

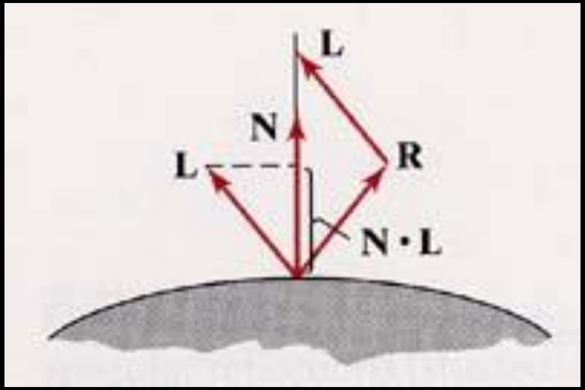
$$I_{l,spec} = \begin{cases} k_s I_l (V \cdot R)^{n_s}, & \text{if } \mathrm{V} \cdot \mathrm{R} > 0 \text{ and } \mathrm{N} \cdot \mathrm{L} > 0 \\ 0.0, & \text{if } \mathrm{V} \cdot \mathrm{R} \le 0 \text{ or } \mathrm{N} \cdot \mathrm{L} \le 0 \end{cases} \tag{2-5}$$

Basic (Local)Illumination Models

3) Specular Reflection and *Phong* Model



Modeling specular reflections (shaded area) with parameter n_s

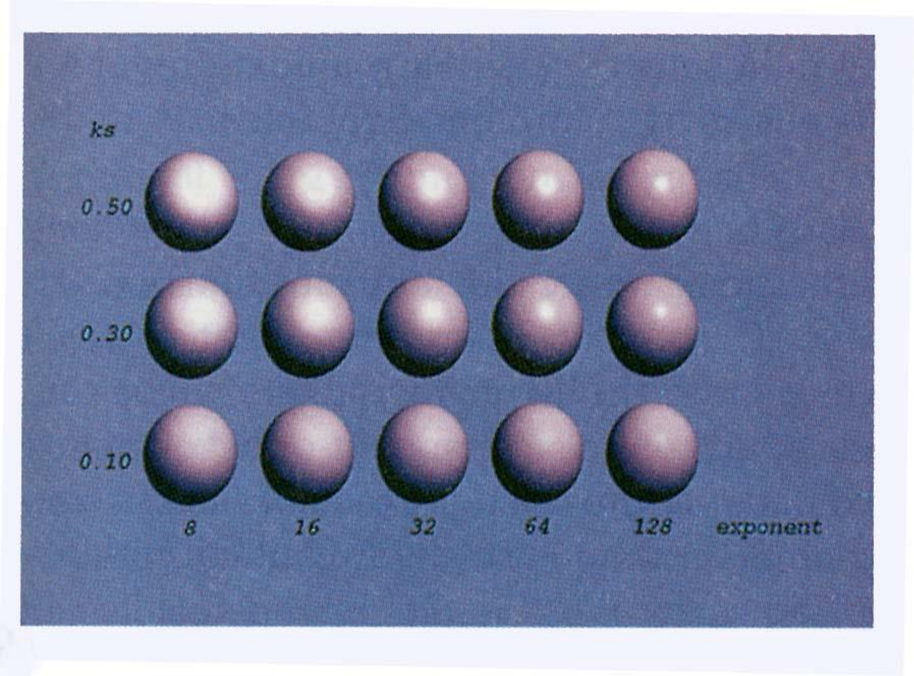


The projection of either L or R onto the direction of the normal vector N has a magnitude equal to N˙L.

$$R = (2N \cdot L)N - L$$

Basic (Local)Illumination Models

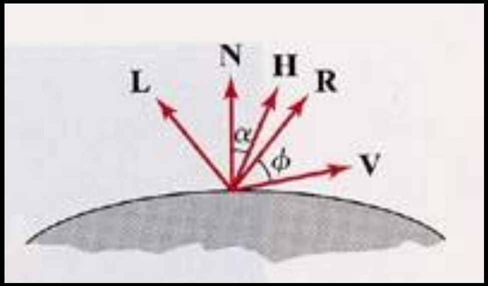
3) Specular Reflection and *Phong* Model



Specular reflections from a spherical surface for varying specular parameter values and a single light source

Basic (Local)Illumination Models

3) Specular Reflection and *Phong* Model



Halfway vector H along the bisector of the angle between L and V.

$$H = \frac{L + V}{|L + V|}$$

Color Intensity Calculations

- ***Phong Algorithm:***

$I = I_a + I_d + I_s$

I_a: ambient reflection

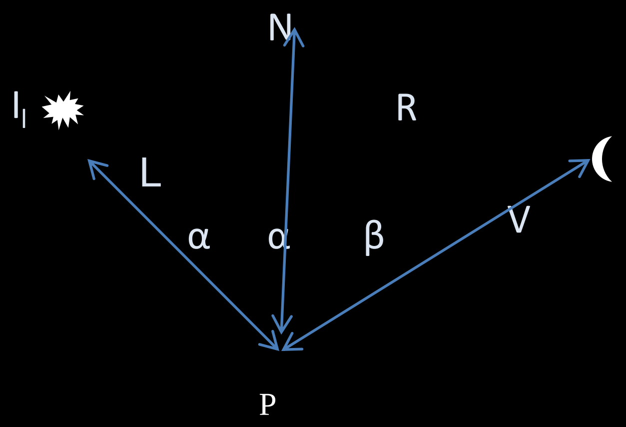
I_d: diffuse reflection

I_s: specular reflection

$I_a = k_a I_e$

$I_d = k_d I_l \cos\alpha$ here $\cos\alpha = (L \cdot N)$

$I_s = k_s I_l \cos^m\beta$ here $\cos\beta = (R \cdot V)$



L, N, R, V are vectors

- ***Multiple light sources:***

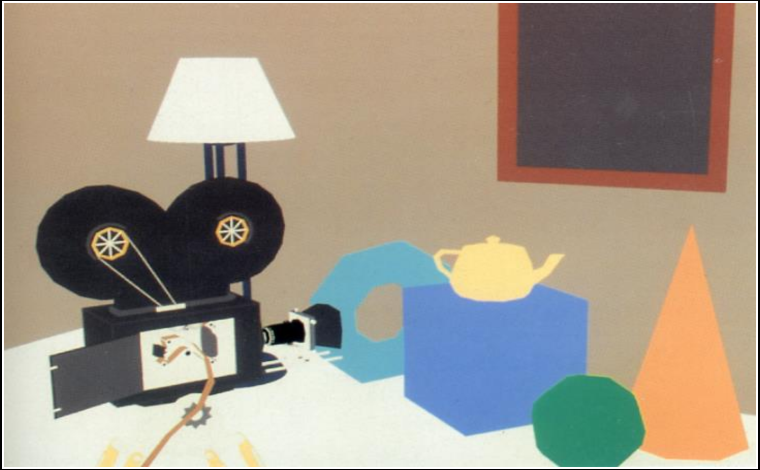
$$I = I_a + \sum_{i=1}^{n} \frac{I_{d_i} + I_{s_i}}{r_i + C}$$

Here r_i is the distance to the light i and C is a constant

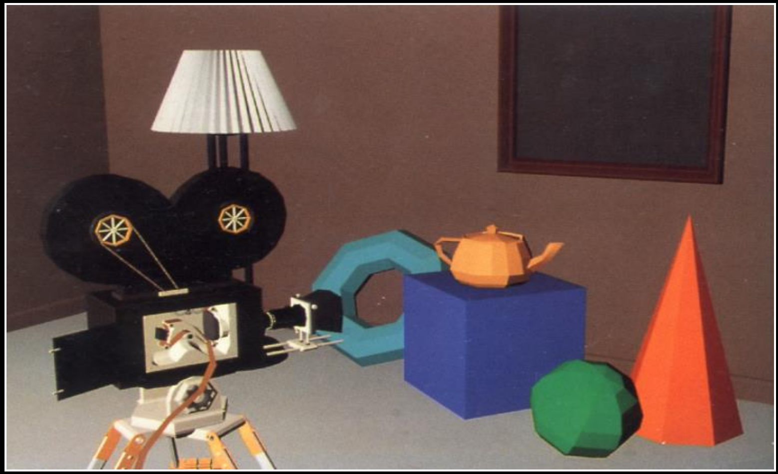
Shading

- Technique To Render Solid Surfaces
- Determines How Surfaces Will Be Filled
- Process for Computing the Color Intensity Value for Each Pixel Contained in a Polygon
- The Most Common Shading Techniques Are:
  - *Flat Shading* *glShadeModel (GL_FLAT);*
  - *Gouraud Shading* *glShadeModel (GL_SMOOTH);*
  - *Phong Shading* *(OpenGL by default doesn't do phong shading)*

Shading Techniques



No Shading



Flat Shading

Gouraud Shading

Phong Shading

Flat Shading

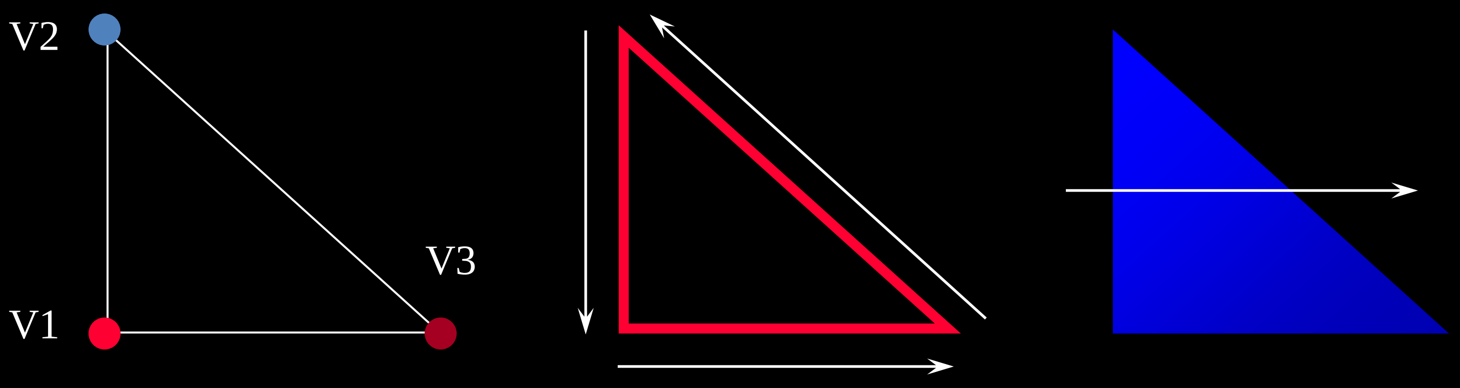
Flat Shading

- Constant Shading Or Flat Shading
- The Simplest and Cheapest and Therefore Fastest Shading Method
- Filling An Entire Polygon with One Color Intensity
- This Model is Only Valid (Realistic) If:
  - *The light source is imagined to be at infinity*
  - *The viewer is at infinity*
  - *The polygon is not an approximation to a curved surface*

Gouraud Shading

Gouraud Shading

- Also called Smooth shading
- Color Interpolation Algorithm
  - *Interpolation along polygon edges*
  - *Interpolation across polygon surfaces*

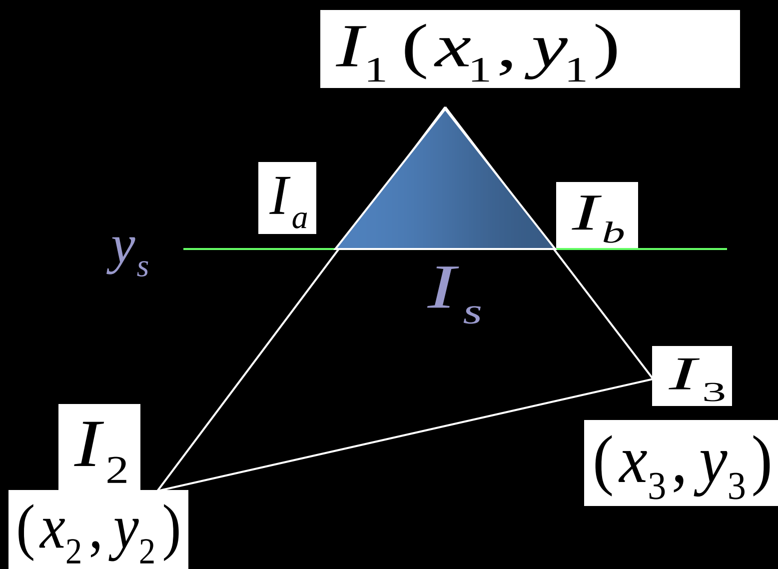


Color Values Given On A Per Vertex Basis

Interpolation Along The Edges

Interpolation Across The Surface

Gouraud Shading Illustration



$$I_a = \frac{1}{y_1 - y_2}\left[I_1(y_s - y_2) + I_2(y_1 - y_s)\right]$$

$$I_b = \frac{1}{y_1 - y_3}\left[I_1(y_s - y_3) + I_3(y_1 - y_s)\right]$$

$$I_s = \frac{1}{x_b - x_a}\left[I_a(x_b - x_s) + I_b(x_s - x_a)\right]$$

$$y_s = j + 1$$

$$I_{a,j+1} = I_{a,j} + \Delta I_a$$

$$\Delta I_a = \frac{1}{y_1 - y_2}(I_1 - I_2)$$

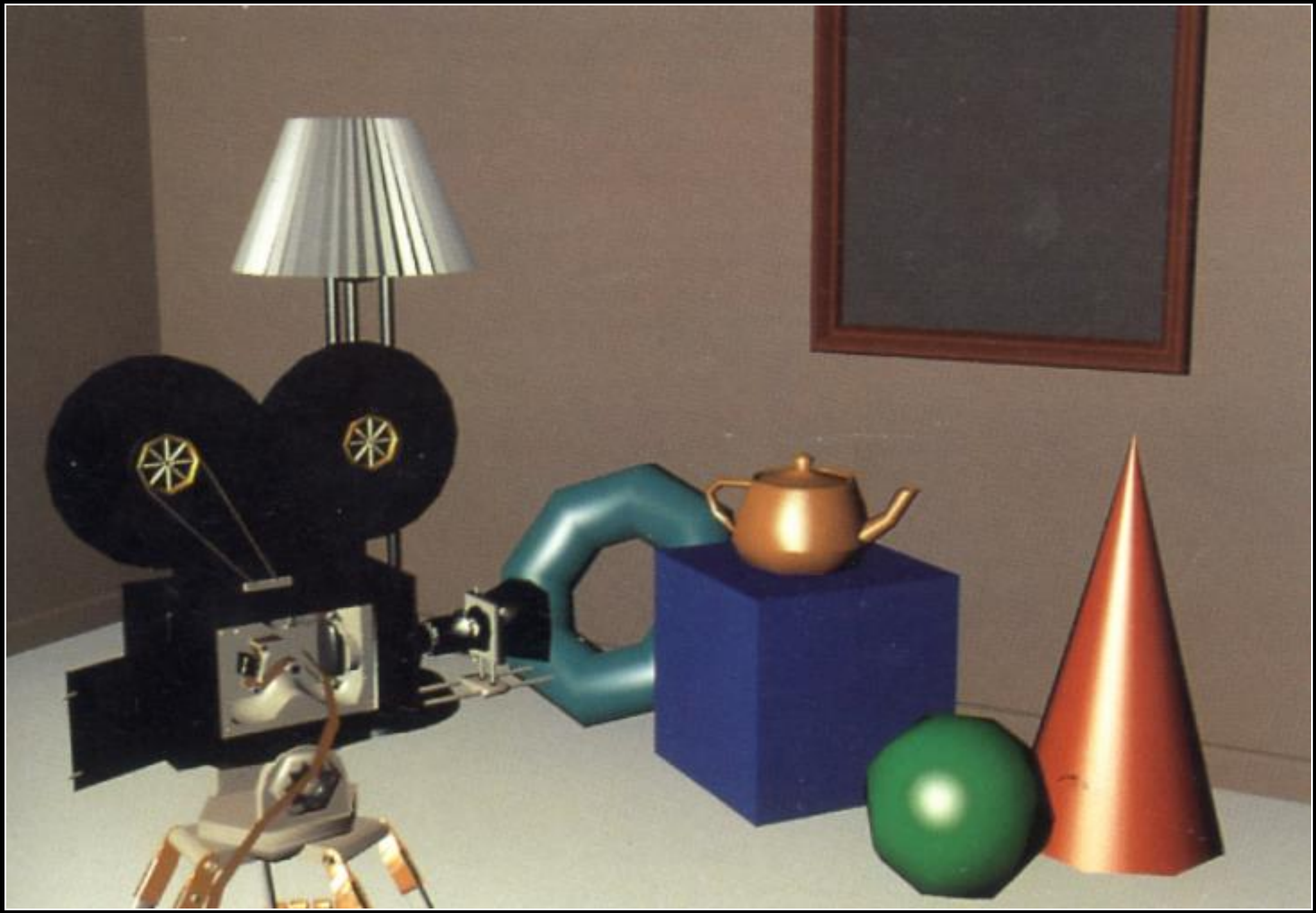
$$I_{b,j+1} = I_{b,j} + \Delta I_b$$

$$\Delta I_b = \frac{1}{y_1 - y_3}(I_1 - I_3)$$

$$I_{i+1,s} = I_{i,s} + \Delta I_s$$

$$\Delta I_s = \frac{1}{x_b - x_a}(I_b - I_a)$$

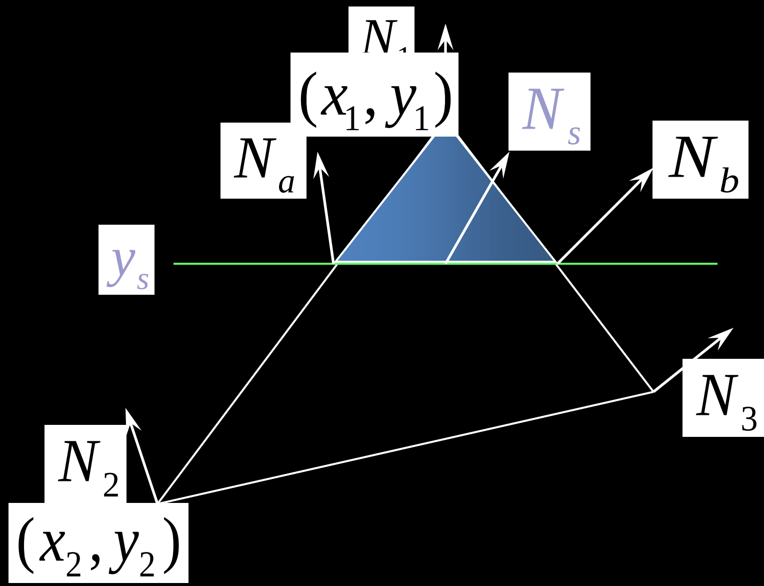
Phong Shading



Phong Shading

- An Interpolation Process Similar To Gouraud Shading
- Interpolation Over Normal Vector Instead of Vertex Color
  - *Normal vectors tell about an objects orientation*
  - *Surface orientation is important in respect to the position of*
    - *The observer/viewer of a scene*
    - *The source of lighting*
- Creates greater realism than Gouraud shading
  - *Specially when combined with an illumination model*
  - *Usually implemented through application software*
  - *Very computing intense*

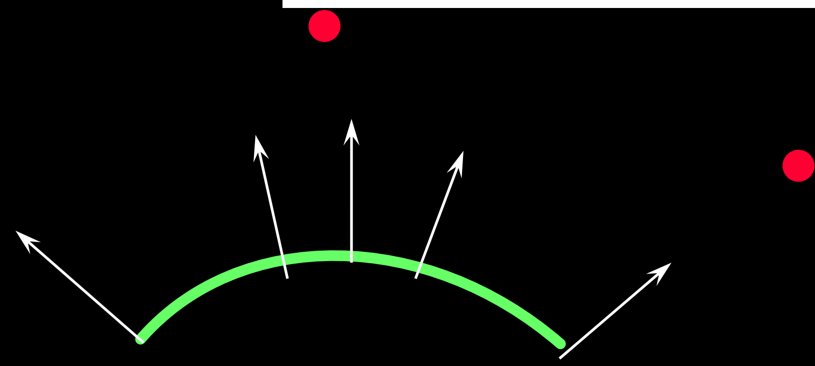
Phong Shading Illustration



$$\vec{N}_a = \frac{1}{y_1 - y_2}\left[\vec{N}_1(y_s - y_2) + \vec{N}_2(y_1 - y_s)\right]$$

$$\vec{N}_b = \frac{1}{y_1 - y_3}\left[\vec{N}_1(y_s - y_3) + \vec{N}_3(y_1 - y_s)\right]$$

$$\vec{N}_s = \frac{1}{x_b - x_a}\left[\vec{N}_a(x_b - x_s) + \vec{N}_b(x_s - x_a)\right]$$



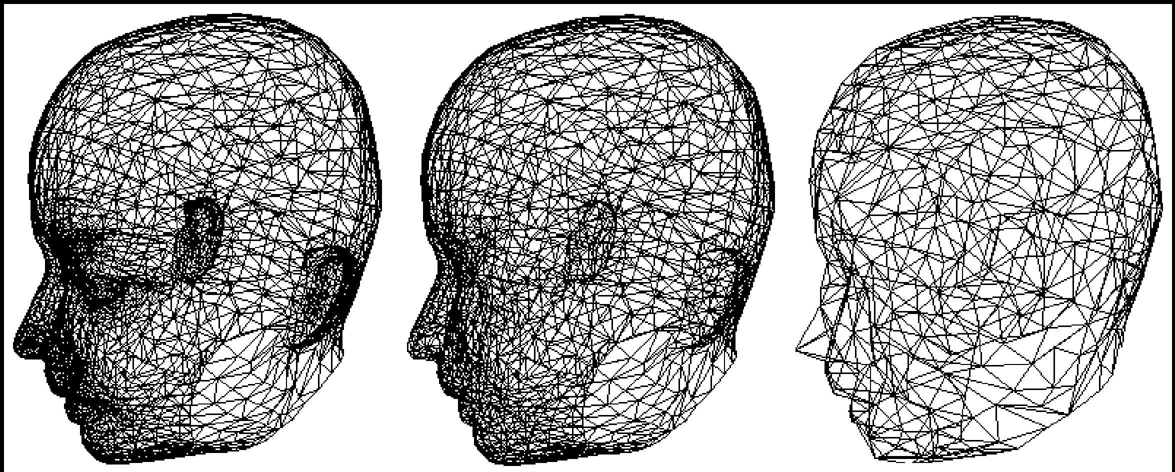
Fill-Area Primitives

- **Fill (Filled) Area**

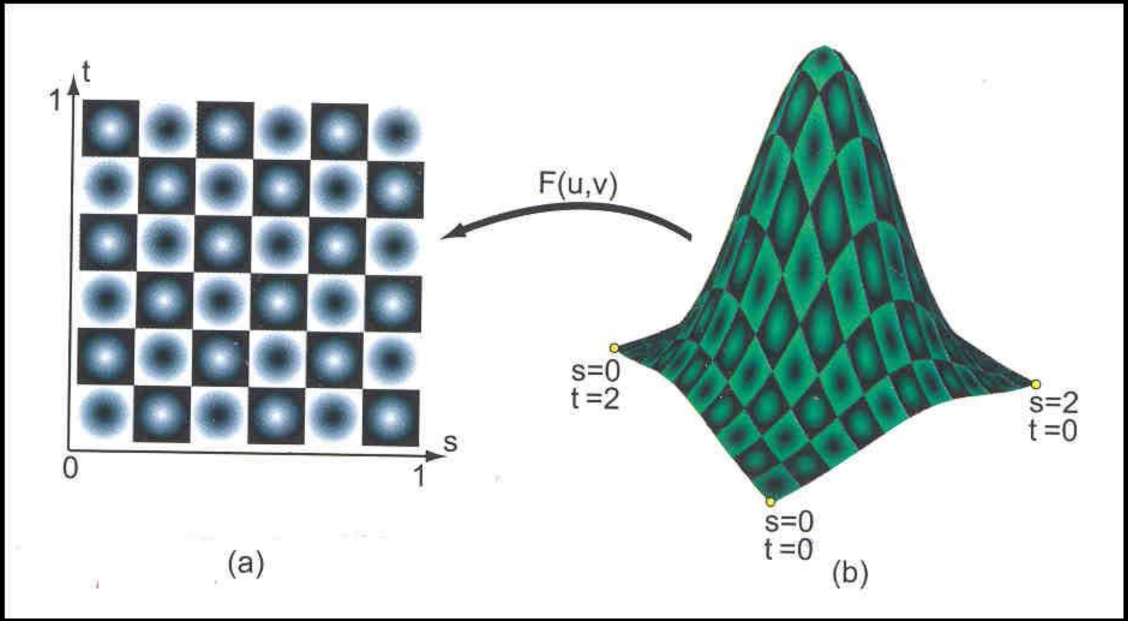
  -- An area filled with some solid color or a pattern

- **Surface Tessellation**

  -- Approximating a curved surface with polygon facets (a polygon mesh)

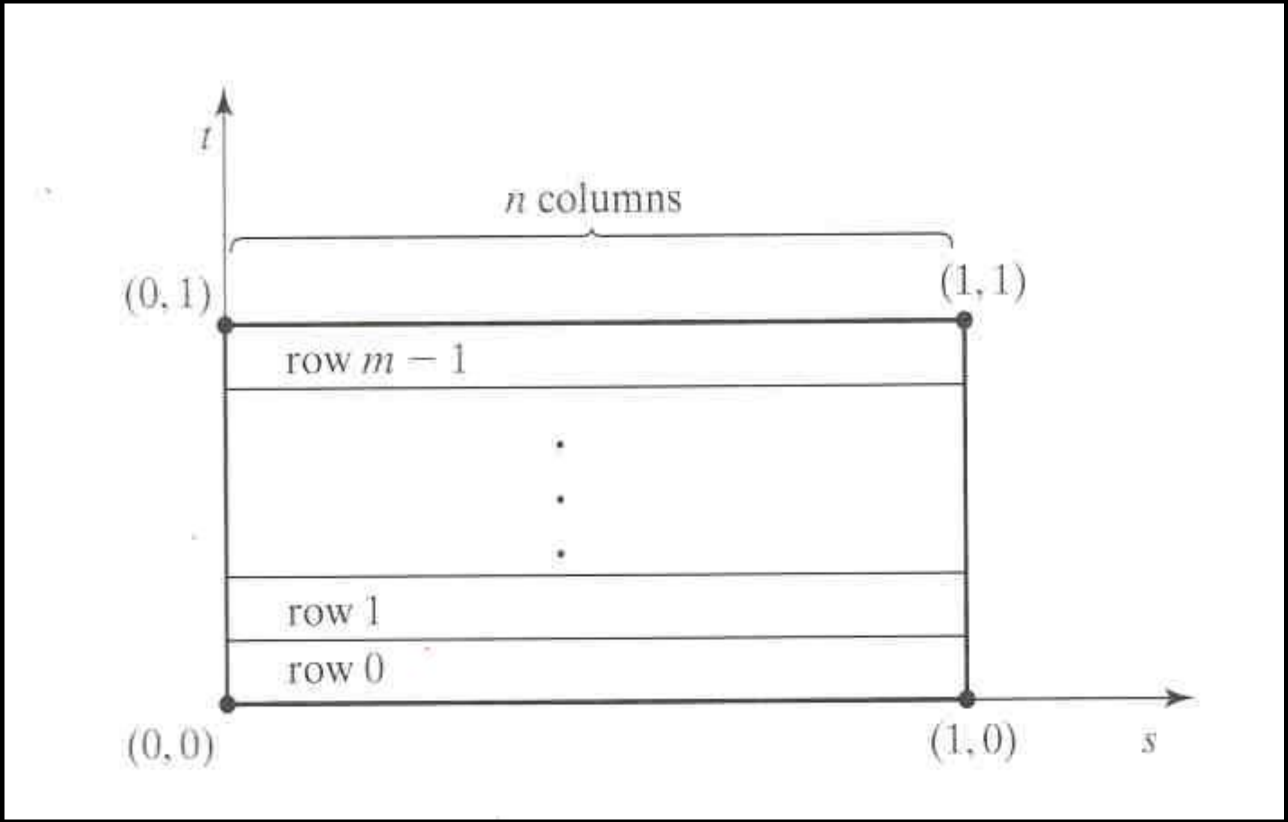


Texture Mapping



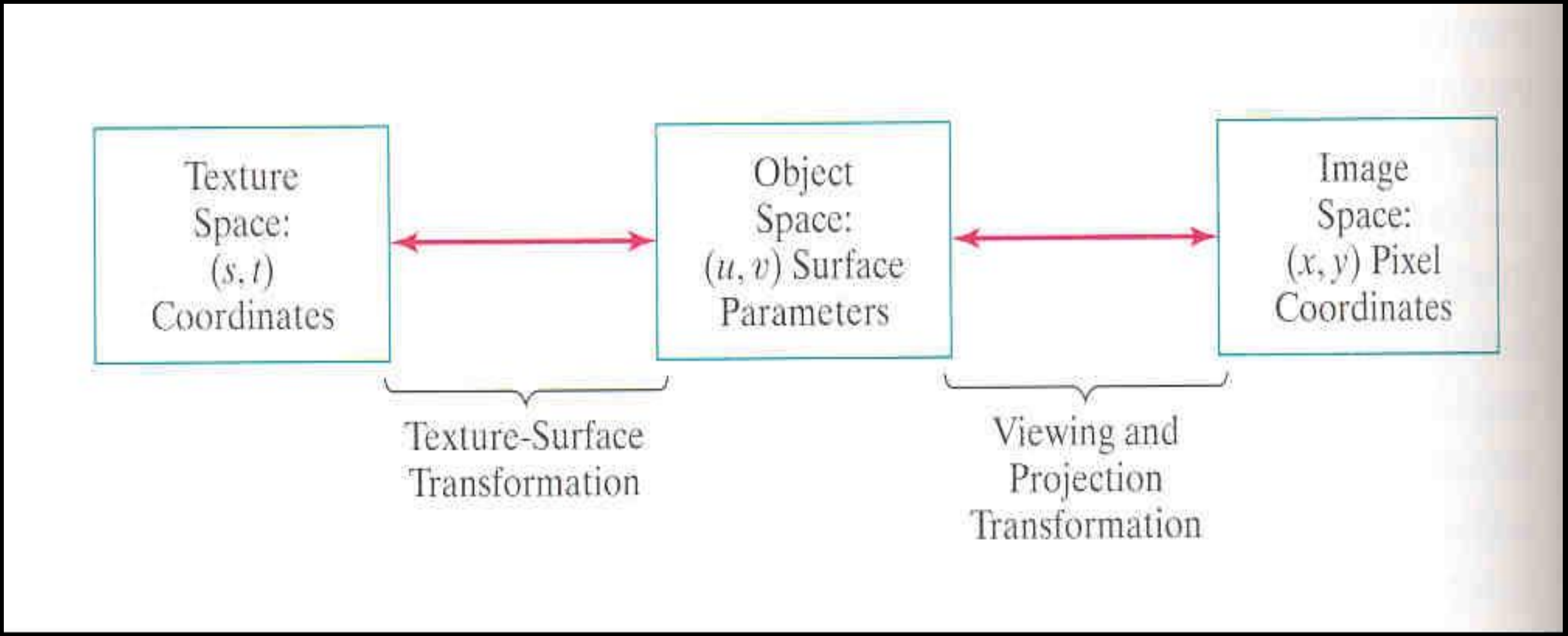
Texture Mapping
(a) Texture Image; (b) Texture-mapped object

Texture Mapping



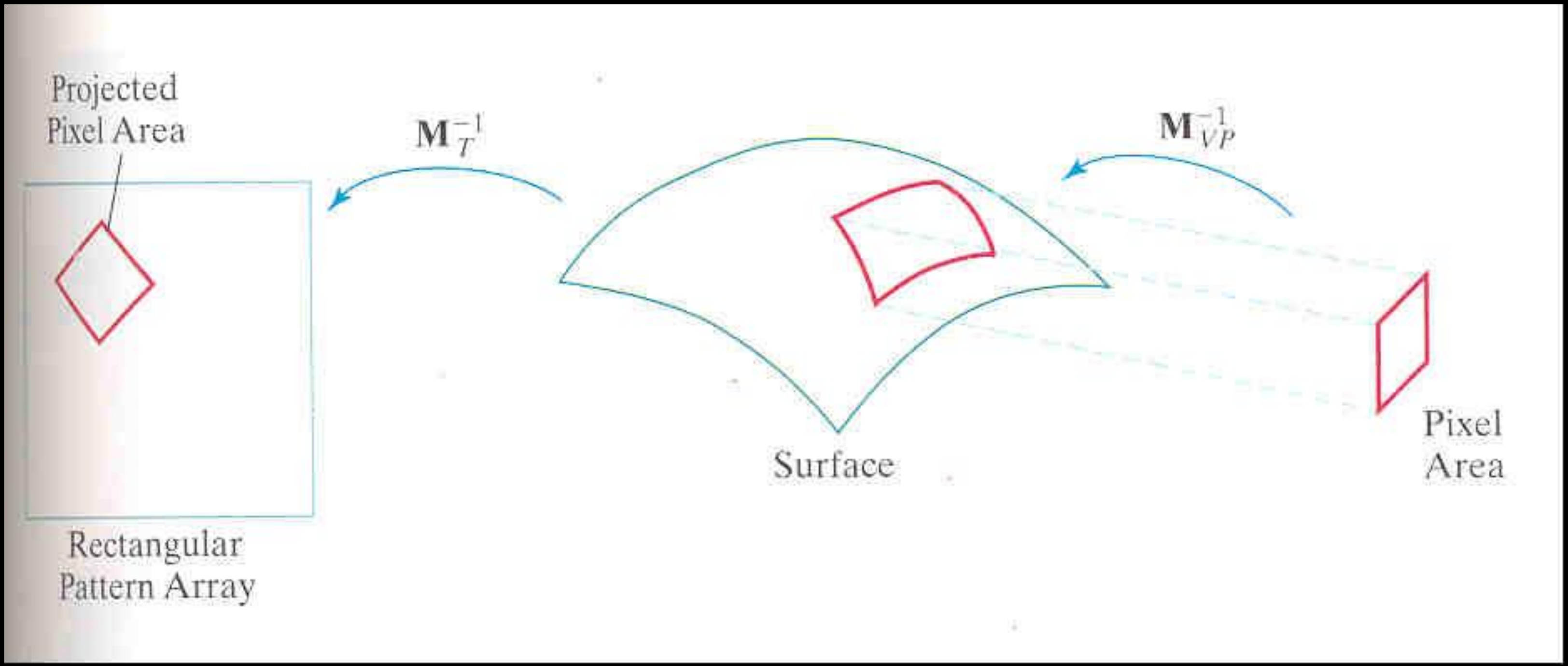
Texture Image

Texture Mapping



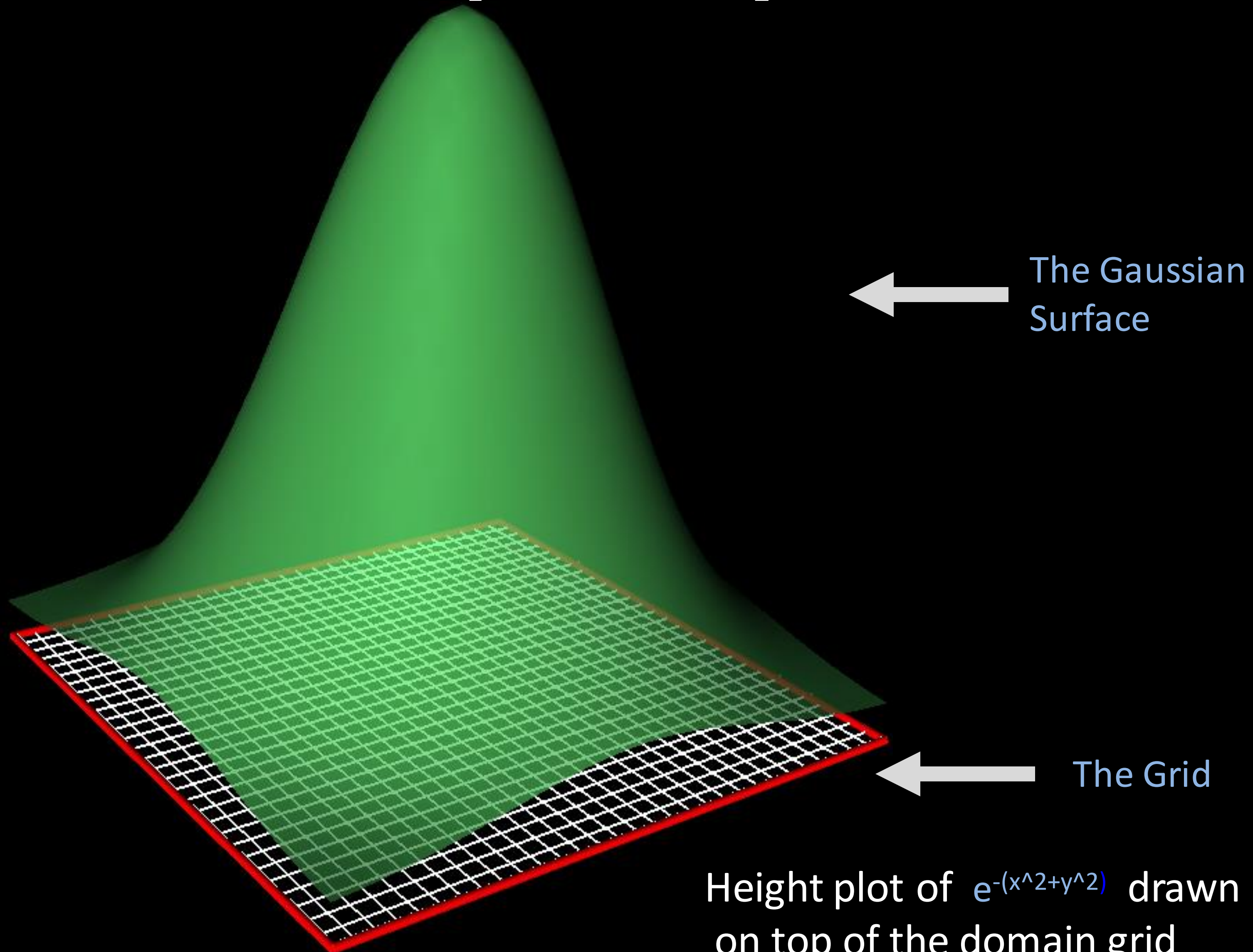
Texture Mapping: Coordinates Transformations

Texture Mapping



Texture Mapping

Transparency and Blending



Height plot of $e^{-(x^2+y^2)}$ drawn on top of the domain grid

Visualization Pipeline

$$f(x, y) = e^{-(x^2+y^2)}$$

Continuous data

Data Acquisition

float data[N_x,N_y]

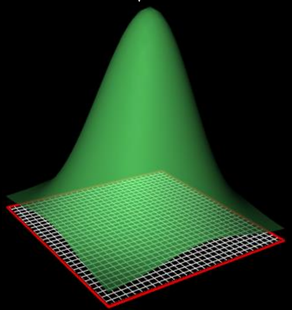
Discrete dataset

Data Mapping

Class Quad

Geometric object

Rendering



Displayed image