Fog and Cloud Effects

Karl Smeltzer
Alice Cao
John Comstock

Goal

- Explore methods of rendering scenes containing fog or cloud-like effects through a variety of different techniques
- Atmospheric effects make rendered scenes
    - become more natural and realistic
    - create a better sense of depth
    - impact the viewer to wonder whether they are indeed seeing a photo or a rendered picture

Natural Variation

- Real environmental fog and clouds vary greatly in size, shape, definition, density, etc.
- Not feasible to judge one rendering method as the best or most realistic

Enclosed Fog vs. Overall Fog

Fog Enclosed in a Volume

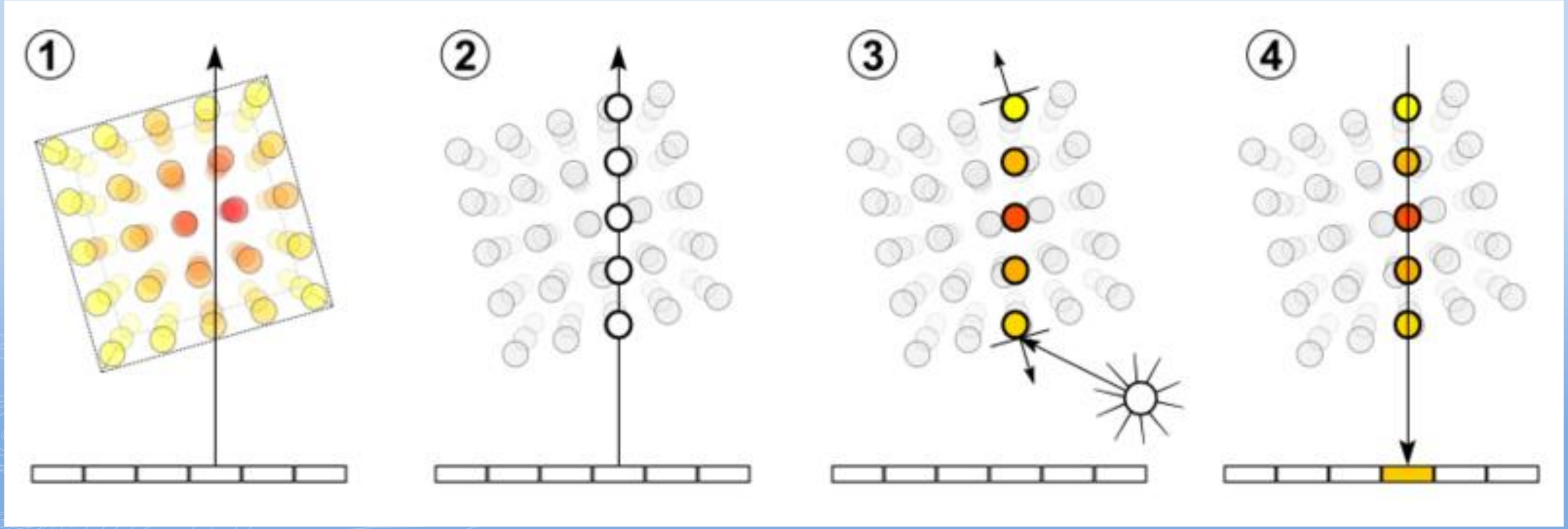
Fog Around the Entire Scene

Fog Rendering Techniques

- Traditional texture mapping
- Two-dimensional noise texturing
- Volumetric ray casting
- Pixel fog
- Single scattering model

Volumetric Ray Casting

- For each pixel
  - Shoot a ray through a three dimensional volume
  - Take samples within the volume at various points
  - Compute the color for each sample point
  - Composite the various samples into a final color

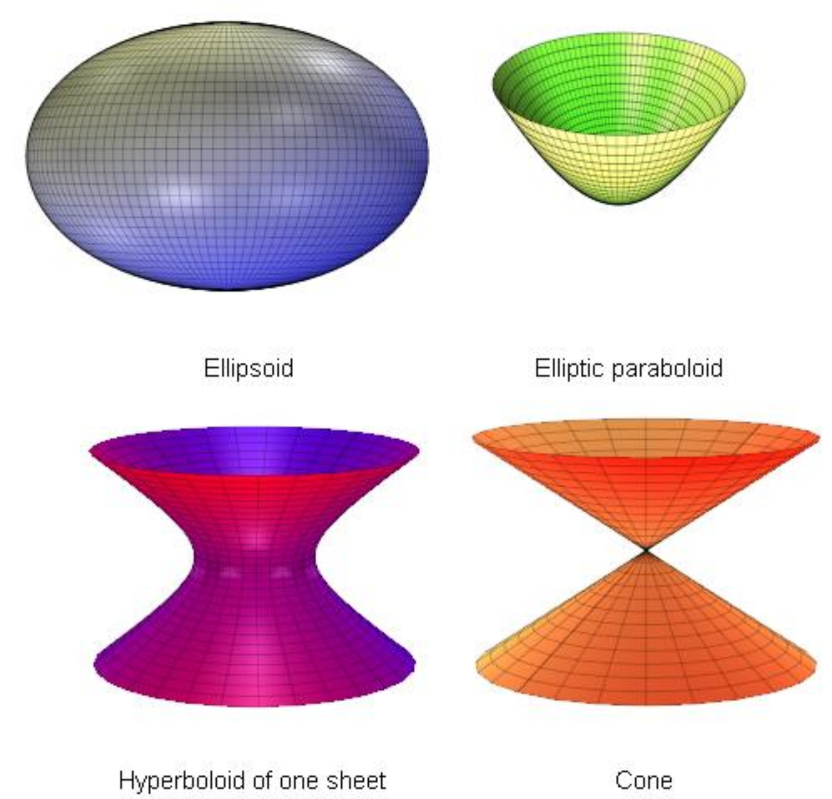


Volumes using Quadric Surfaces

- The basic general form of a three-dimensional quadric surface function:

$AX^2 + BY^2 + CZ^2 + DXY + EXZ + FYZ + GX + HY + IZ + J = 0$

- This implies that any quadric surface anywhere in 3D space can be defined using ten numbers, A through J.



Quadric Surfaces (cont.)

- **This formula can then be reduced through substitution to a more familiar form:**

  $A_ct^2 + B_ct + C_c = 0$ where

  $A_c = AX_d^2 + BY_d^2 + CZ_d^2 + DX_dY_d + EX_dZ_d + FY_dZ_d$

  $B_c = 2AX_oX_d + 2BY_oY_d + 2CZ_oZ_d + DX_oY_d + DY_oX_d + EX_oZ_d + EZ_oX_d + FY_oZ_d + FZ_oY_d + GX_d + HY_d + IZ_d$

  $C_c = AX_o^2 + BY_o^2 + CZ_o^2 + DX_oY_o + EX_oZ_o + FY_oZ_o + GX_o + HY_o + IZ_o + J$

  $R(t) = R_dt + R_o$

- **This allows us to generalize two calculations:**
  - **Visibility**

    We can use $B_c^2 - 4A_cC_c$ to determine whether an intersection exists and then solve the complete quadratic equation for t_1 and t_2, to substitute back into our $R(t)$ above.

  - **Normal at any point**

    Given the normal defined as the following:

    $N_x = 2AX + DY + EZ + G$

    $N_y = 2BY + DX + FZ + H$

    $N_z = 2CZ + EX + FY + I$

    With the value for t, from the visibility calculation above, we can define

    $P_x = X_dt + X_o$

    $P_y = Y_dt + Y_o$

    $P_z = Z_dt + Z_o$

    We can now substitute the P values in for X, Y, and Z in the normal equation for the normal at any point.

Pixel Fog

- Apply a fog factor to each pixel to determine how much of a pixel is obscured by fog. The fog factor is calculated by linearly interpolating the accumulated color and the fog color along a ray at various sample points.
- This can be applied to the complete rendered scene and not just a single piece of geometry
- DirectX uses the following formula to compute the fog coefficient:

$$f = e^{-(\rho * d * n)}$$

with p = density, d = camera distance, and n = Perlin noise factor.

Pixel Fog (cont.)

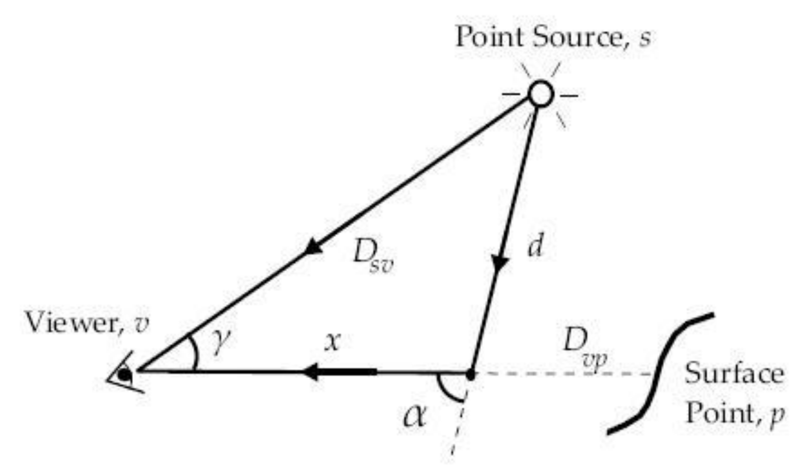
- This method is popular and used in both the OpenGL and the DirectX models
  - only have to apply a fog density value once to each pixel
- However, this method has many short comings:
  - glow around light sources are missing
  - object shading tends to be incorrect
  - may look two-dimensional in 3D space

Single Scattering Model

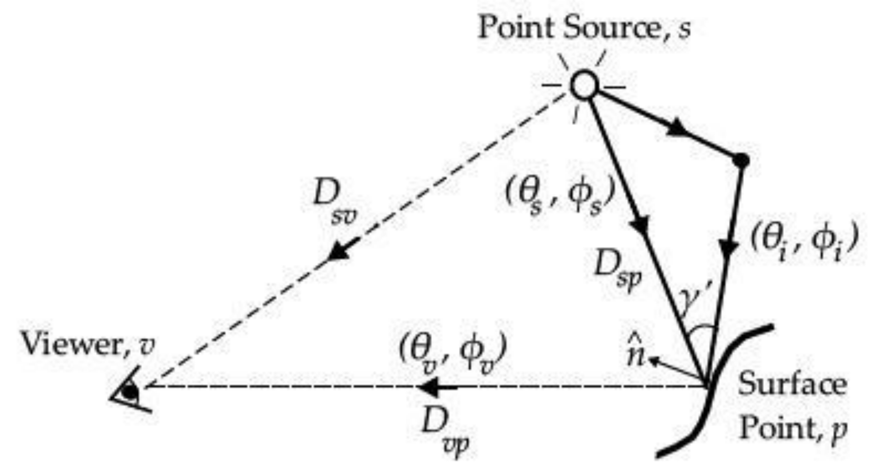
- The single scattering model improves upon the pixel fog formula by:
  - adding the glow around point light sources
  - softening the diffuse radiance on reflected objects
  - brightening dark regions
  - dimming and diffusing specular highlights
  - creating a noticeable loss of contrast and color saturation

Single Scattering Model (cont.)

- Our current Phong illumination model calculates light based on the following:



- Calculating a single scatter point and use a Point Spread Formula for each pixel



Single Scattering Model (cont.)

- From a simple Phong illumination model we can add the single scatter point and Point Spread Formula to create the following:

Implementation

- Volumetric ray casting with Perlin noise in a couple quadric surfaces
  - Sphere
  - Cone
  - Cylinder
  - Additional volumes (time permitting)
- Pixel fog with Perlin noise and Gaussian Filter
- The single scattering model with 2D texture lookup (time permitting)

Potential Problems

- We are unable to achieve any functioning, satisfactory results
- The results don’t simulate real fog very well, so our results look really fake
  - Possible 2D vs. 3D appearance
  - Real fog refracts light off each water droplet and is significantly more complex than our models