Introduction to Computer Graphics

8. Buffers and Mapping techniques (B)

National Chiao Tung Univ, Taiwan
By: I-Chen Lin, Assistant Professor

Textbook: E.Angel, Interactive Computer Graphics, 5th Ed., Addison Wesley
Ref:Hearn and Baker, Computer Graphics, 3rd Ed., Prentice Hall

Examples of Highly Reflected Models

T1000 from movie “Terminator 2”

Silver Surfer from movie “Fantastic 4:Rise of the Silver Surfer”

How to Handle Highly Specular Surfaces?

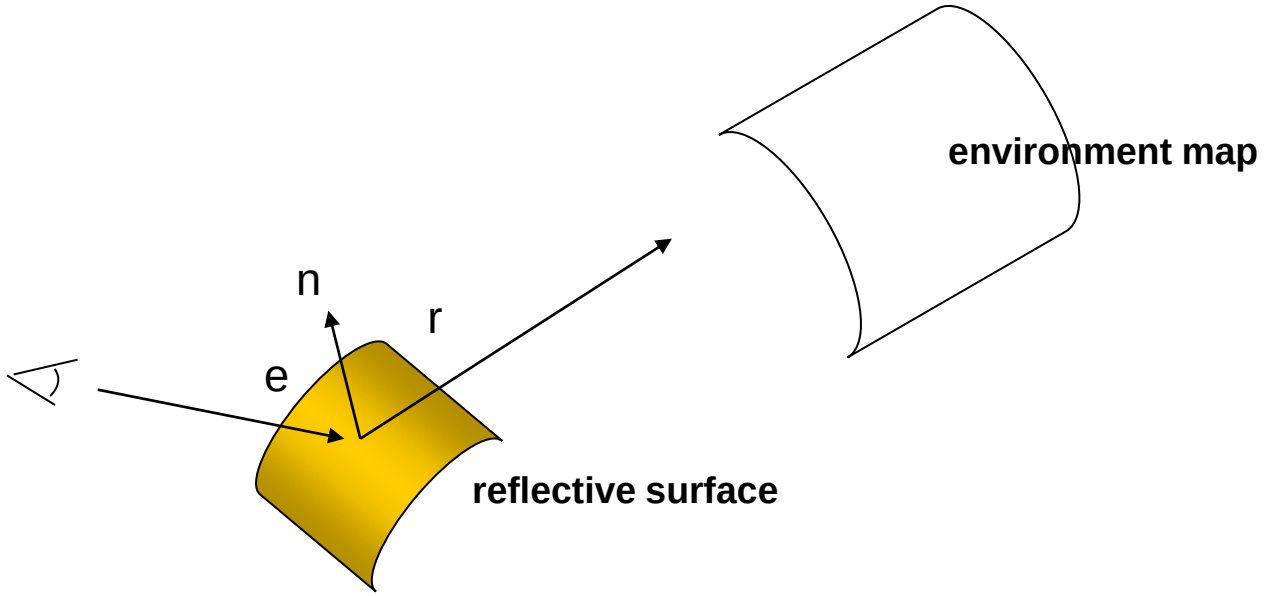
- How to render a flat mirror?
- How to render a mirror-like object in a virtual scene?
- How about rendering such a object in a real scene?

Environment Mapping

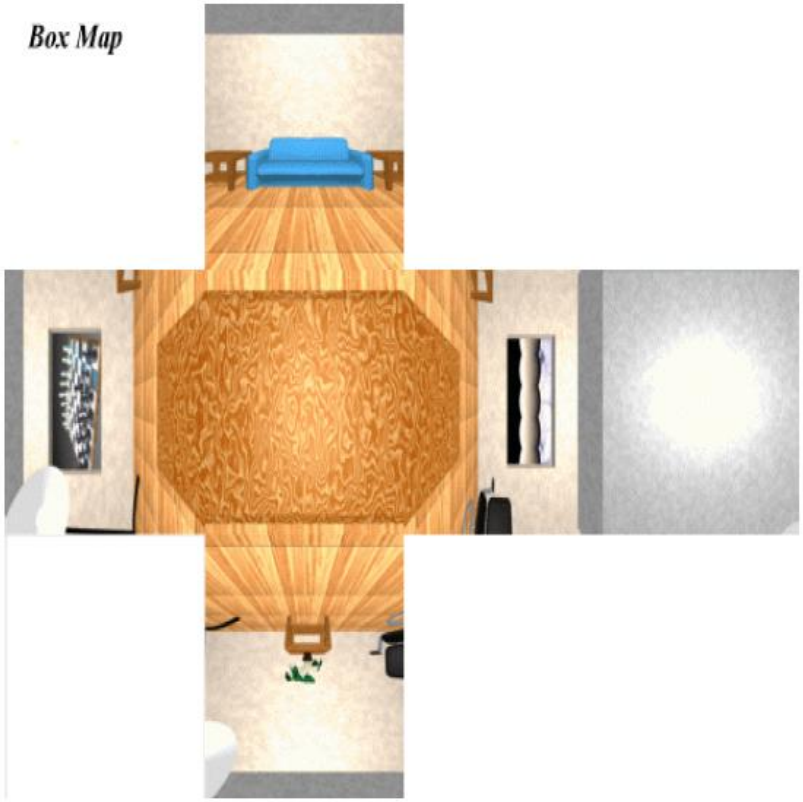
- For real-time applications
- A.k.a reflection mapping
- First proposed by Blinn and Newell.
- A efficient way to create reflections on curved surfaces
  - can be implemented using texture mapping supported by graphics hardware

Environment Mapping

- Assume the environment is far away and there's no self-reflection
- The reflection at a point can be solely decided by the reflection vector.



Environment Mapping



Pictures from lecture notes of Computer Graphics course, UNC

Sphere Mapping

- The image texture is taken from a perfectly reflective sphere.
- Assume the size of the sphere →0. Map the rays to the environment

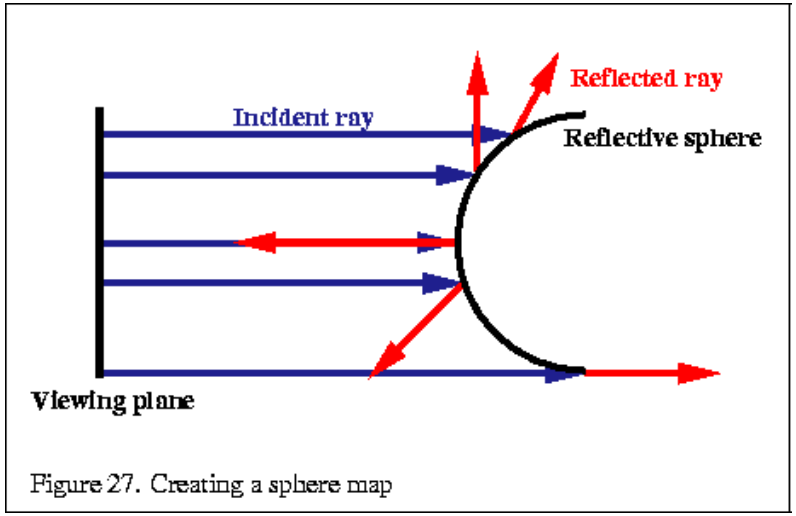
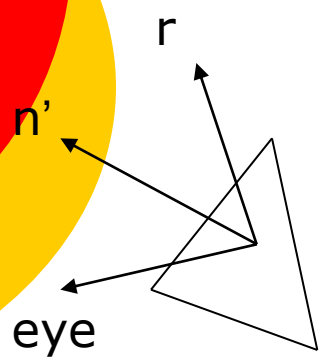


Figure 27. Creating a sphere map


Figure 31. Sphere map generated from image cube faces in Figure 30

Pictures from OpenGL tutorial. http://www.opengl.org

Sphere Mapping



- To access the sphere map texture
  - Compute the reflection vector *r* on the object surface as usual

    ($r = (rx,ry,rz) = e' - 2(n' \cdot e')n'$)

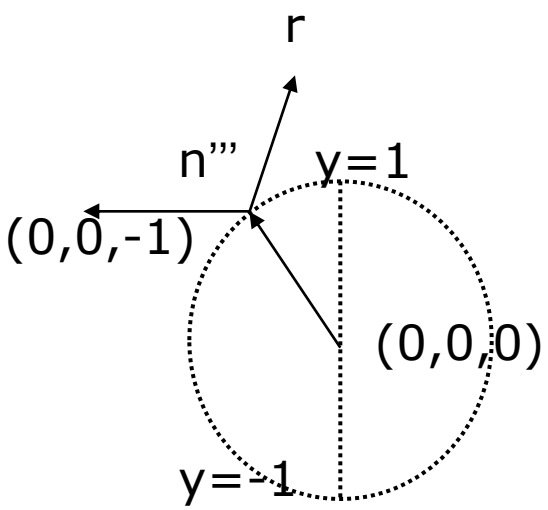
  - Access the texture: compute the sphere normal in the local space $n'' = (rx,ry,rz) + (0,0,-1)$

$$n''' = \left(\frac{r_x}{m}, \frac{r_y}{m}, \frac{r_z - 1}{m} \right) \qquad m = \sqrt{r_x^{\,2} + r_y^{\,2} + (r_z - 1)^2}$$

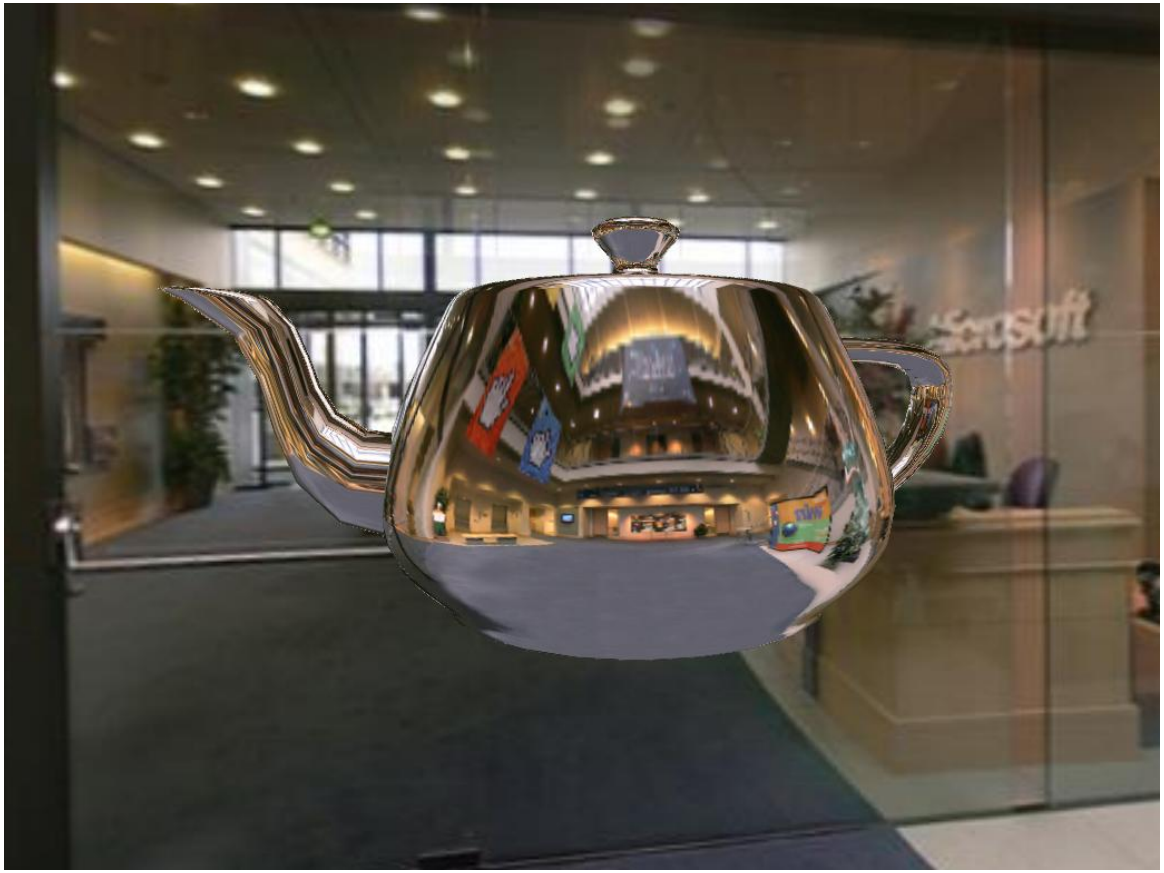
  - Normalized the screen space from [-1,1] to [0,1]

$$s = \frac{r_x}{2m} + \frac{1}{2} \qquad t = \frac{r_y}{2m} + \frac{1}{2}$$

  - (s, t) is the target texture coordinate



Sphere Mapping



Samples from DirectX 9.0 SDK

Bump and Normal Mapping

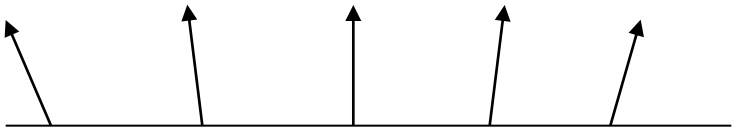
- Represent surface details and avoid heavy geometric computation.

http://www.ozone3d.net/tutorials/bump_mapping.php

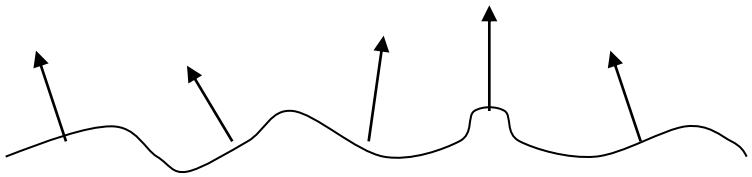
Bump and Normal Mapping

- Calculate reflection (Phong Shading) with a normal map.
- Or with a height map.

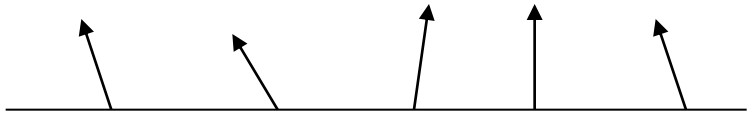
Smooth surface



Bumpy surface



Bump-mapped surface



Bump Mapping

- Let $P = P(u,v)$ be a smooth parametric surface, with normals $N = N(u,v)$.
- Apply a bump map $b = b(u,v)$:

$$P' = P + bN$$

$$N' = P'_u \times P'_v$$

$$P'_u = \frac{\partial}{\partial u}(P + bN) = P_u + b_u N + bN_u \approx P_u + b_u N$$

$$P'_v = \frac{\partial}{\partial v}(P + bN) = P_v + b_v N + bN_v \approx P_v + b_v N$$

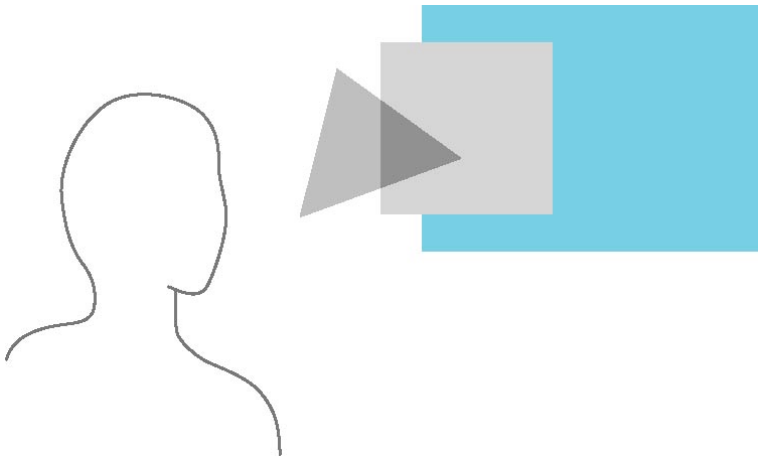
Bump Mapping (cont.)

$$N' \approx (P_u + b_u N) \times (P_v + b_v N)$$
$$= P_u \times P_v + b_u (N \times P_v) + b_v (P_u \times N) + b_u b_v (N \times N)$$
$$= N + b_u (N \times P_v) + b_v (P_u \times N)$$

Compositing, Blending and Accumulation Buffer

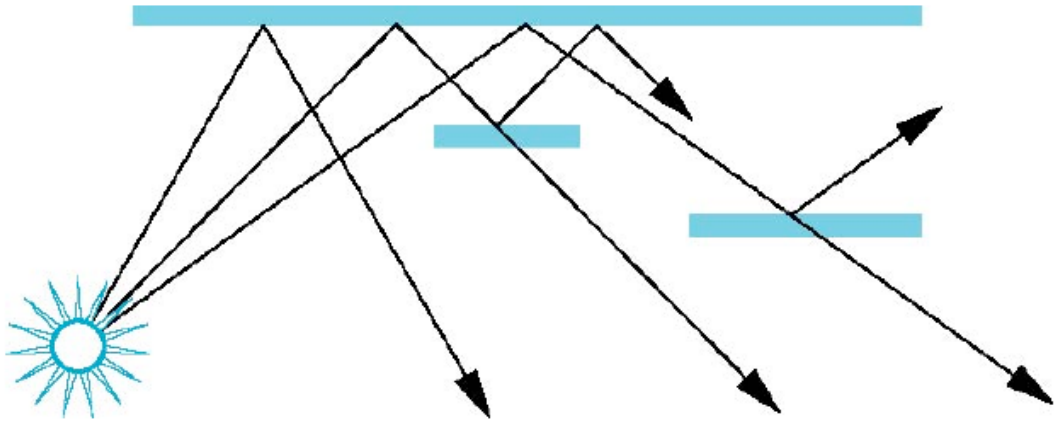
Opacity and Transparency

- Opaque surfaces permit no light to pass through
- Transparent surfaces permit all light to pass
- Translucent surfaces pass some light
  - translucency = 1 – opacity (α)



Physical Models

- Dealing with translucency in a physically correct manner is difficult due to
  - the complexity of the internal interactions of light and matter
  - Using a pipeline renderer

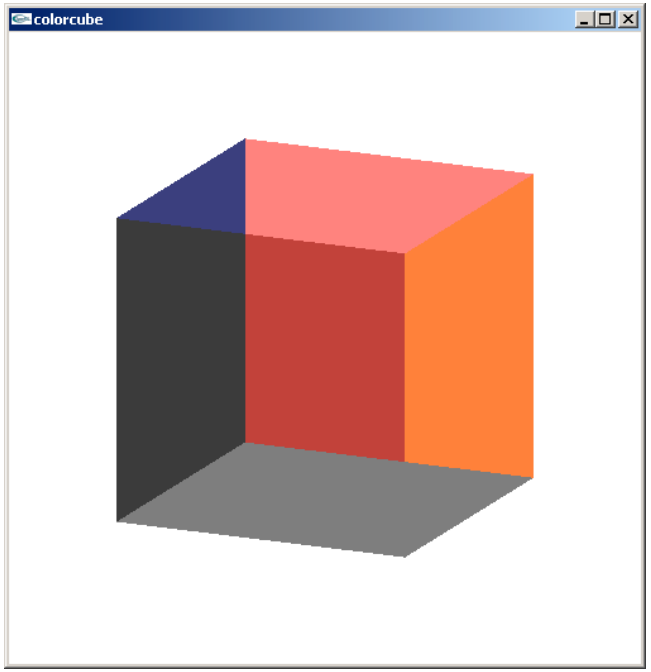


Blending Equation

- We can define source and destination blending factors for each RGBA component
  - $s = [s_r, s_g, s_b, s_\alpha]$
  - $d = [d_r, d_g, d_b, d_\alpha]$
- Suppose that the source and destination colors are
  - $b = [b_r, b_g, b_b, b_\alpha]$
  - $c = [c_r, c_g, c_b, c_\alpha]$
- Blend as
  - $c' = [b_r s_r + c_r d_r, b_g s_g + c_g d_g , b_b s_b + c_b d_b , b_\alpha s_\alpha + c_\alpha d_\alpha]$

Order Dependency

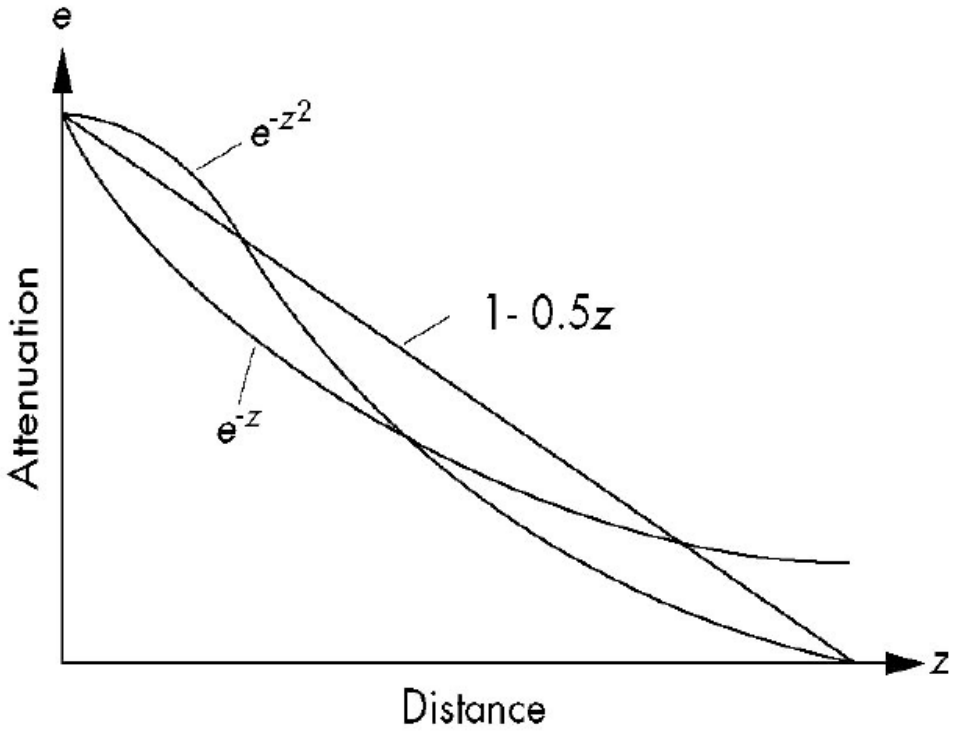
- Is this image correct?
  - Probably not
  - Polygons are rendered in the order they pass down the pipeline
  - Blending functions are order dependent



Fog

- We can composite with a fixed color and have the blending factors depend on depth
  - Simulates a fog effect
  - Blend source color Cs and fog color Cf by
  - Cs'=f Cs + (1-f) Cf
- f is the *fog factor*
  - Exponential
  - Gaussian
  - Linear

Fog Functions



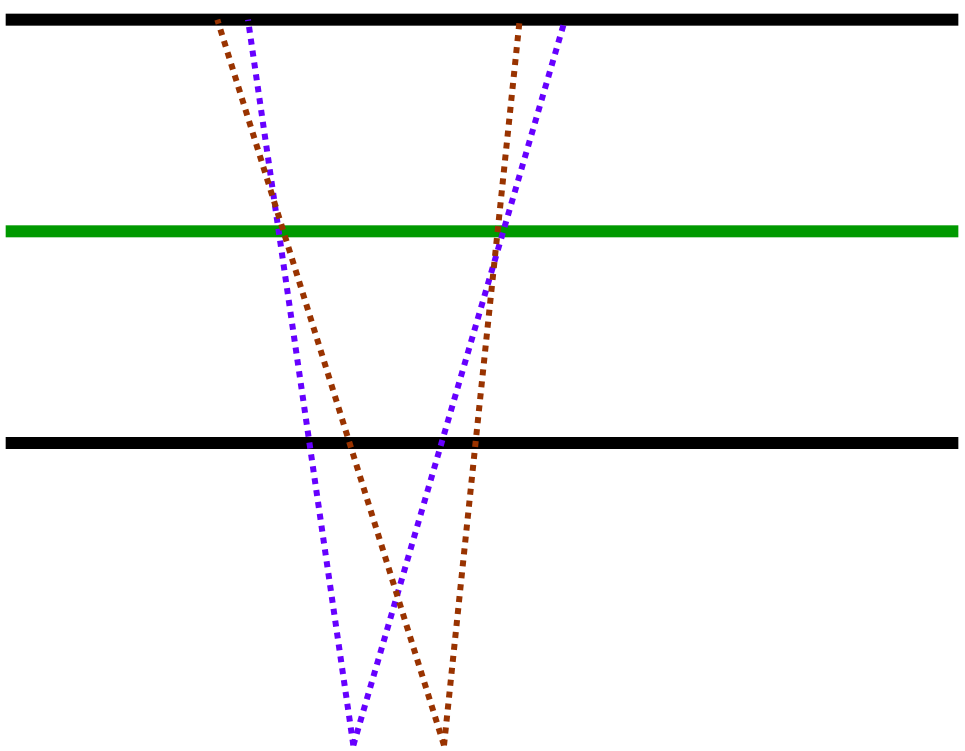
Accumulation Buffer

- Compositing and blending are limited by resolution of the frame buffer
  - Typically 8 bits per color component
- The accumulation buffer is a high resolution buffer
  - 16 or more bits per component
  - Write into it or read from it with a scale factor
- Slower than direct compositing into the frame buffer

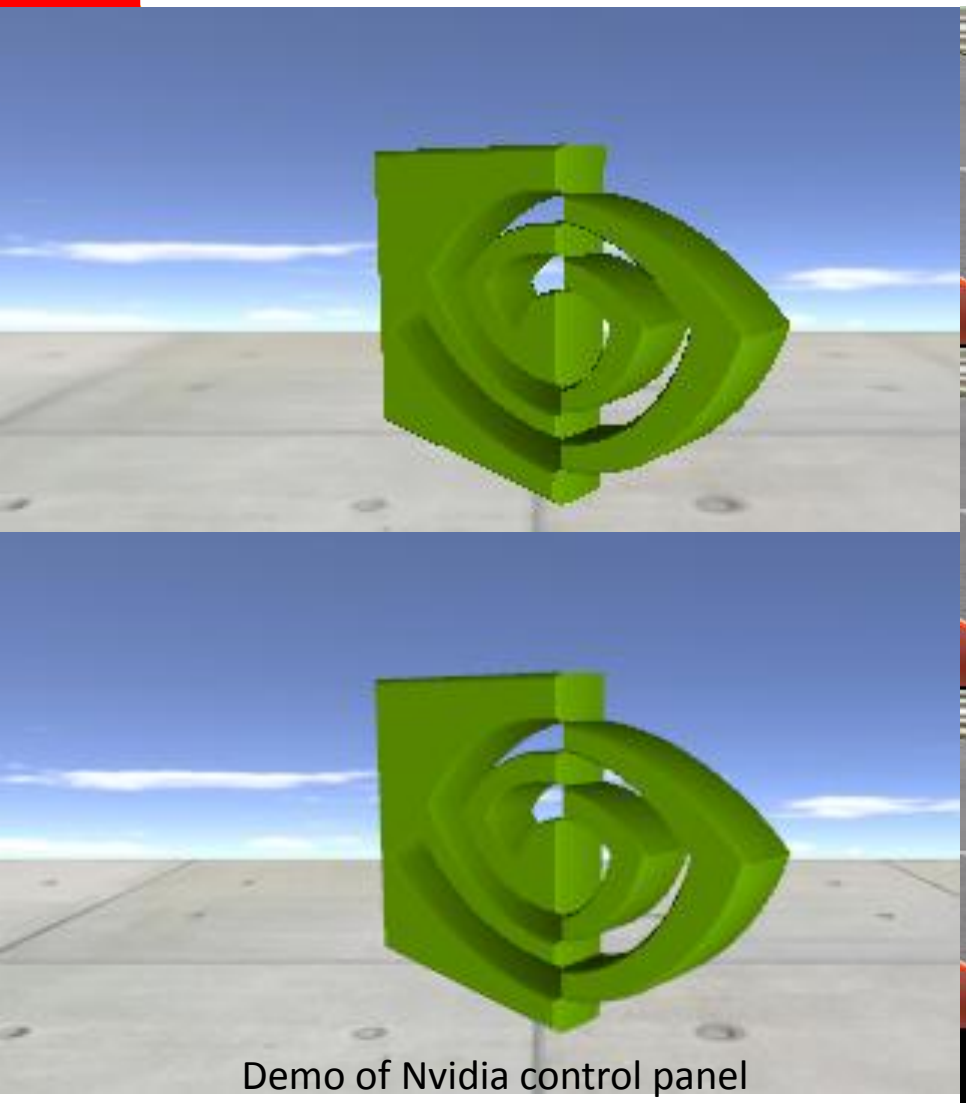
Applications

- Compositing
- Image Filtering
- Whole scene antialiasing
- Motion effects
- …..

Depth of Focus

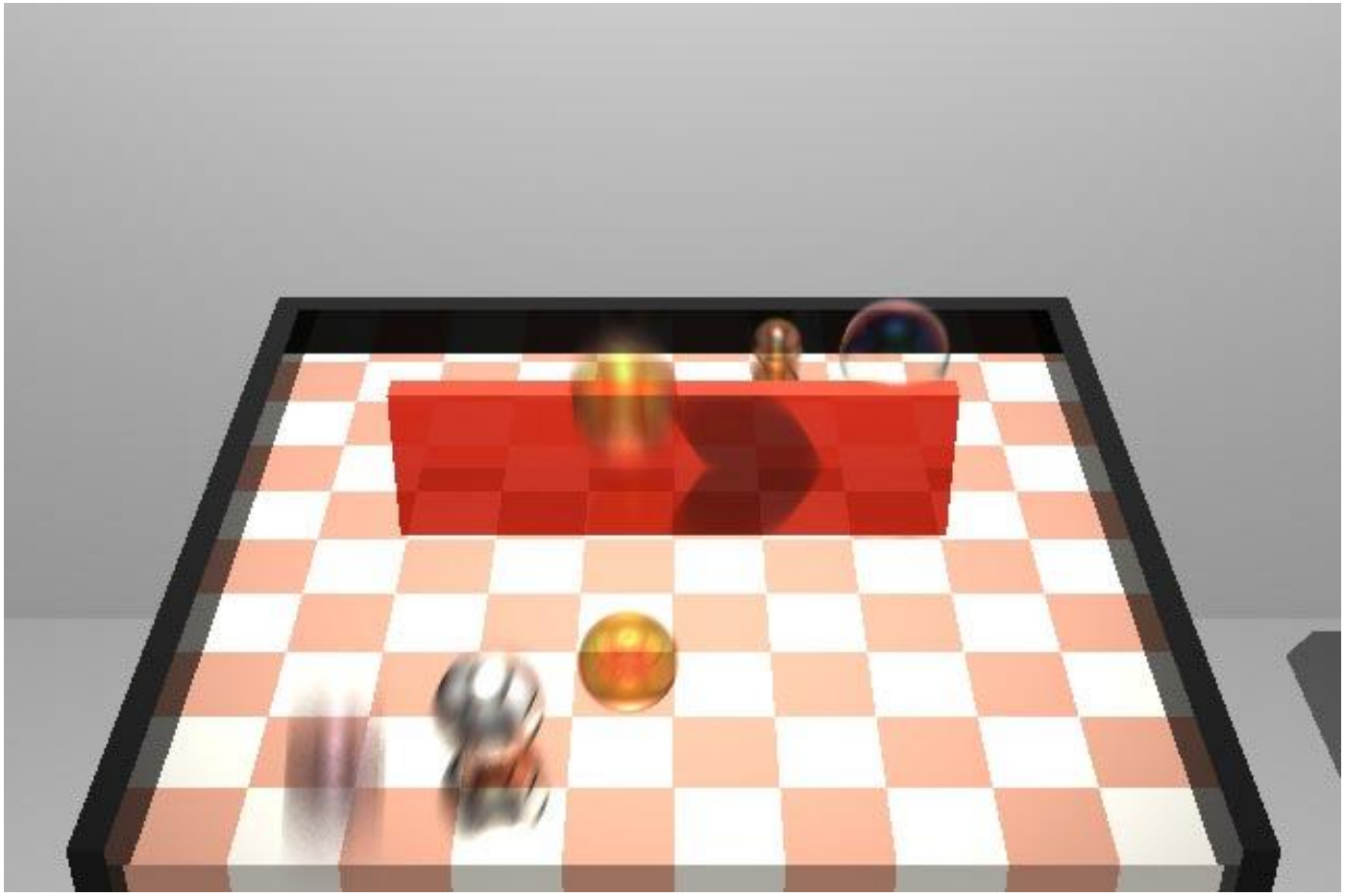


Anti-aliasing (Full screen and Multiple samples)



Demo of Nvidia control panel

Motion blur



http://www.eml.hiroshima-u.ac.jp/gallery/ComputerGraphics/motion_blur/