Web Based Graphics & Virtual Reality Systems

Transparency and Blending, Sampling and Antialiasing

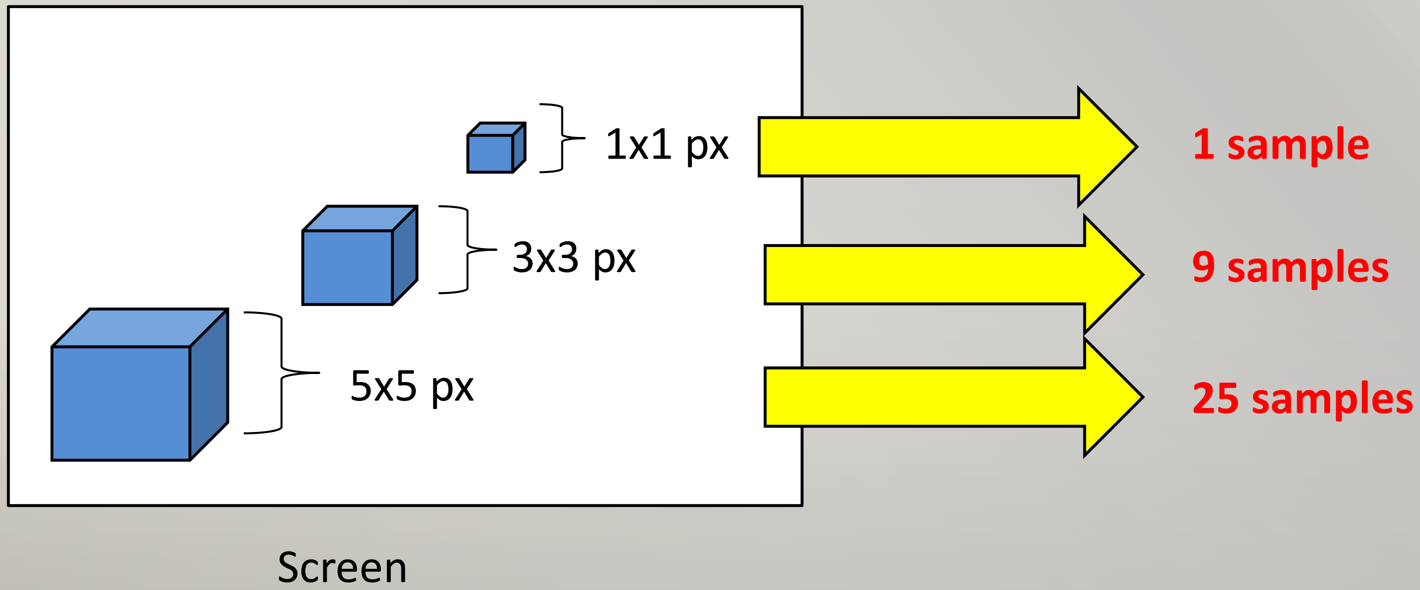
Recap

- In the last lectures, we discussed concepts related to texture mapping object surfaces.
- So that objects can be rendered with richer color and realistic appearance
- This lecture, we will study how transparent objects are defined and how the colors are blended together

Antialiasing and Mipmapping

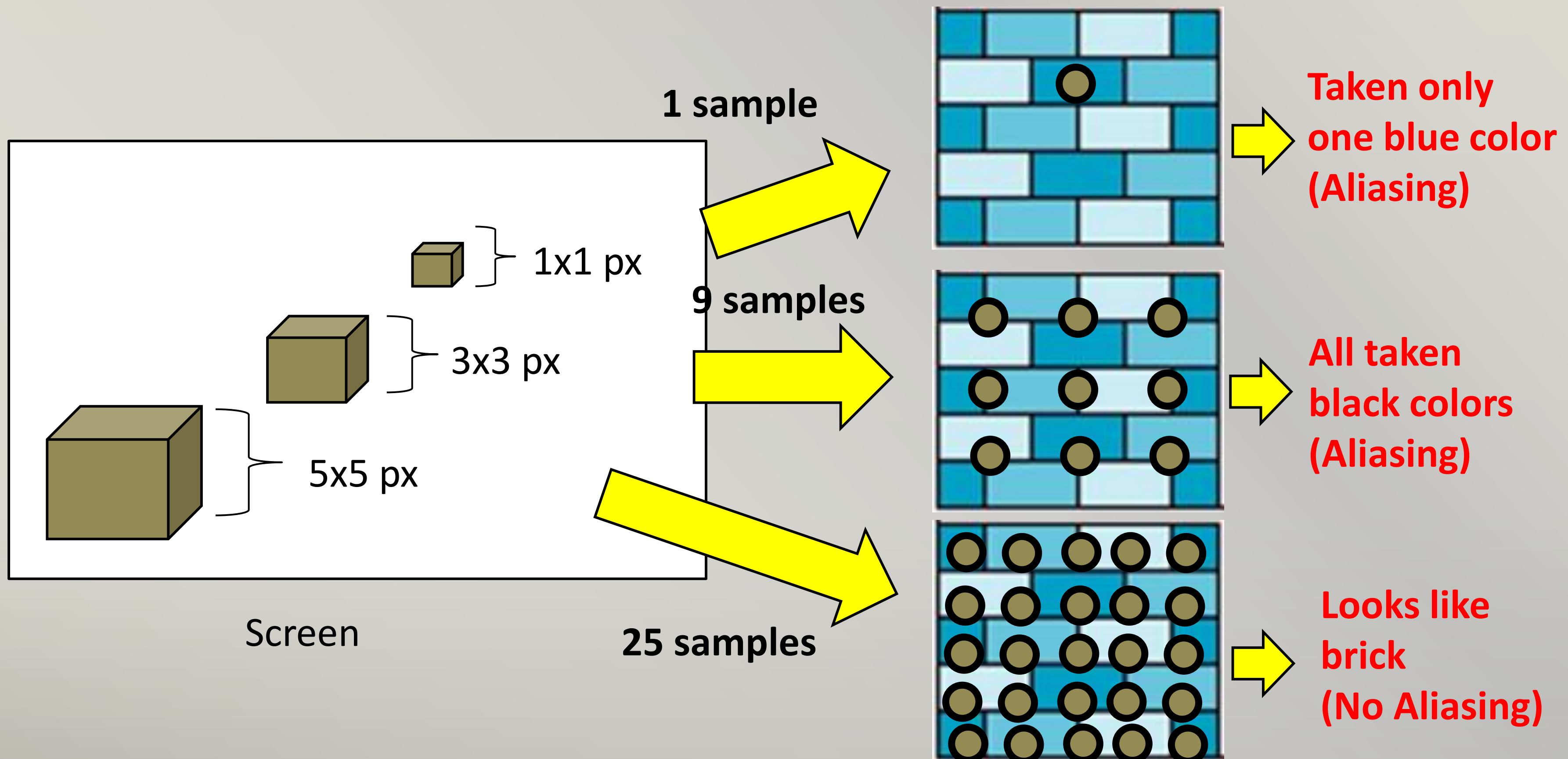
Aliasing Problem

- Imagine when an object is moving away from the camera, it will become smaller on screen
- So it occupies lesser pixels on the screen
- If the object is texture mapped, it will also requires less samples from the texture



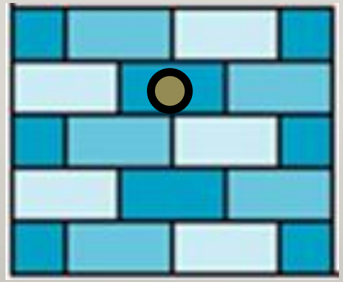
Aliasing Problem

- If point sampling on the texture is used, this may lead to aliasing errors

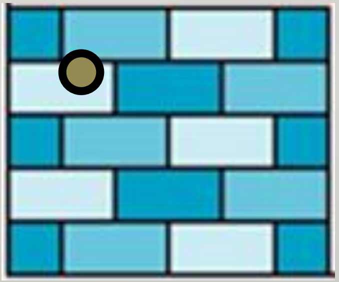


Aliasing Problem

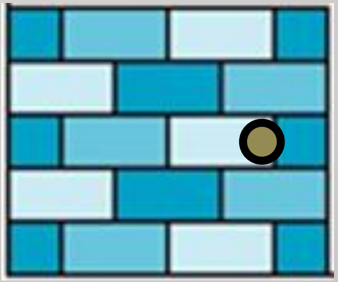
- Aliasing is a common problem in sampling when there is no enough samples taken for a signal (our case is the image)
- To solve it, consider the most extreme case in last example in which only 1 sample is going to be taken
- The question now is where we should take this sample inside the texture ???



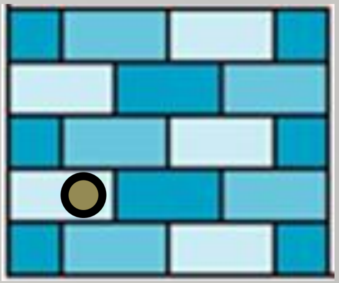
Here?



Here?

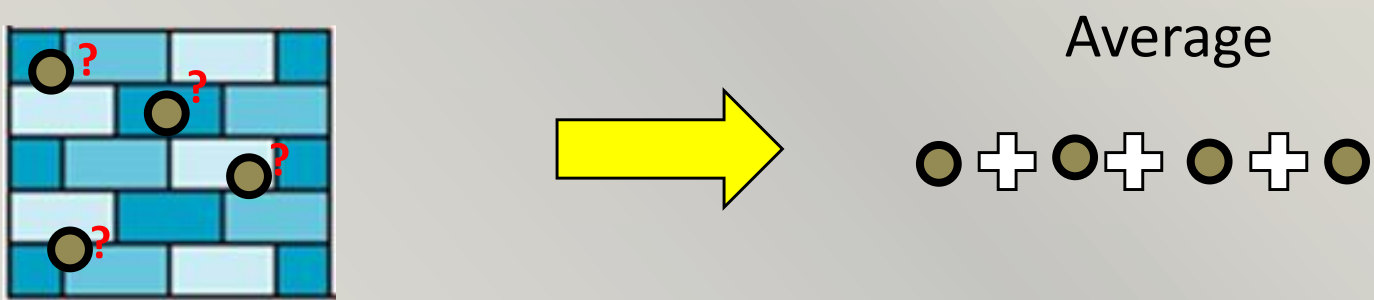


Here?



Here?

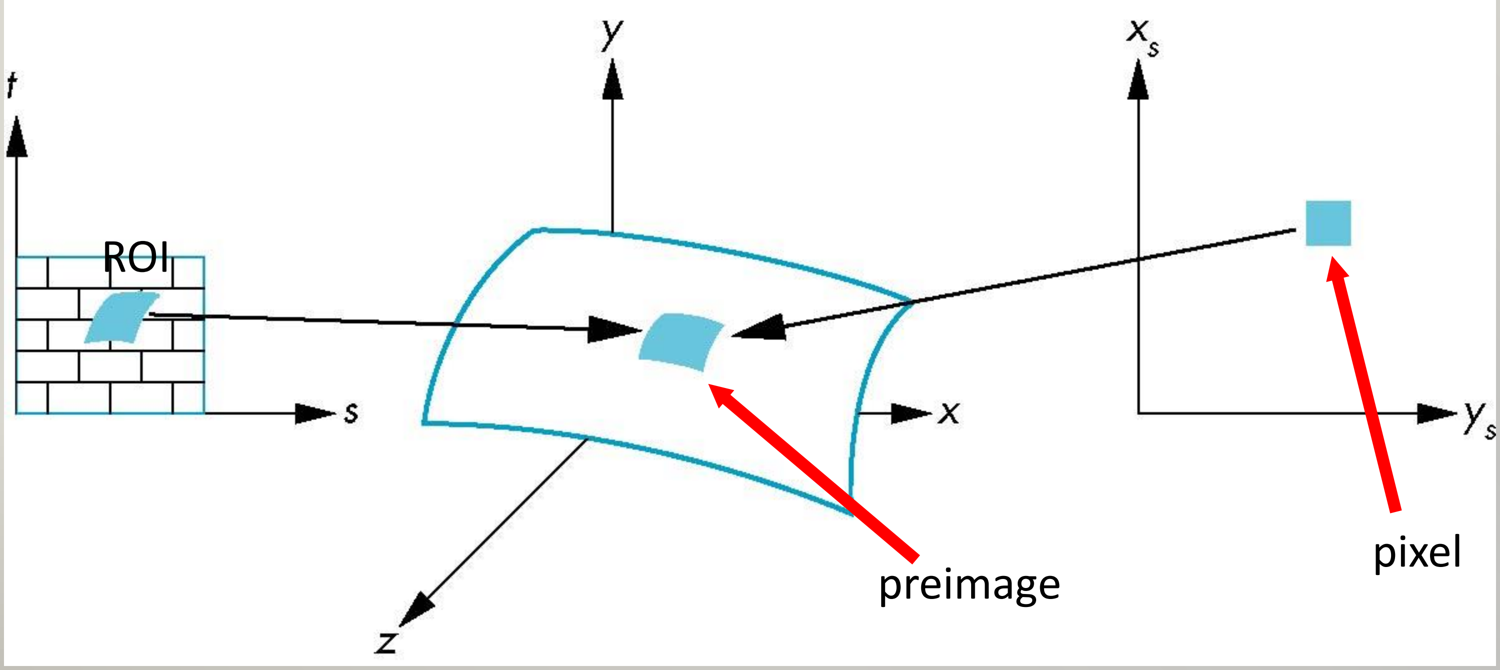
Anti-Aliasing



- It seems that none of the above sampling is reasonable
- Then, how about taking all of them into account, and then <u>**average**</u> the value of them?
- This is one of the commonly used ***anti-aliasing*** approach

Area Averaging

- Usually, we will do a regular sampling within the region of interest (ROI)
  - *Referred as area averaging*
- Although it is slower, the quality will be better

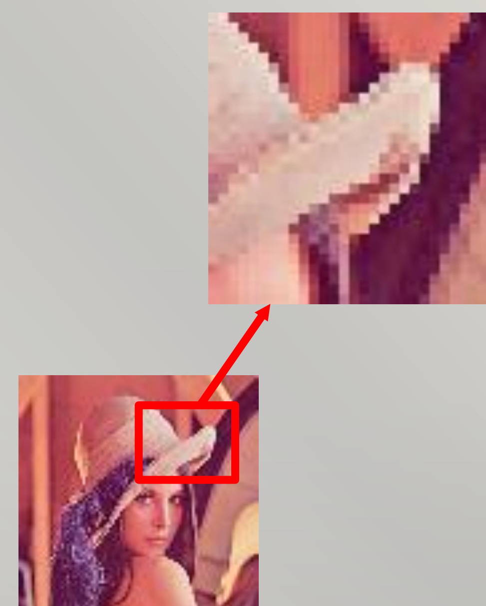


Area Averaging

- Weighted average
  - Apart from normal average, weighted average is commonly used in image processing
  - Usually, the central region has heavier weight
- If you are familiar with image processing, it is the same problem as downsampling
  - Or the problem encountered when the image is enlarged

$$1/256 \times \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Weightings of a Gaussian filter



Area Averaging

- As mentioned before, the averaging/ filtering on texture is slow
- It becomes especially heavy if high resolution texture is used
- Too time consuming for real-time rendering engine
- The method of MIPMapping is therefore proposed to solve the slow filtering

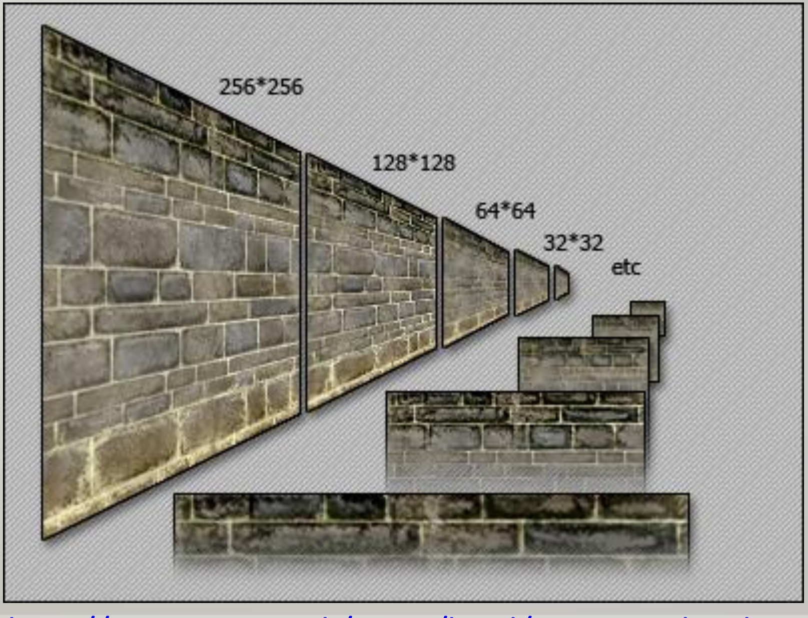
Mipmapped Textures

- *Mipmapping* is proposed to **pre-process** the averaging of texture maps before it really needs
  - Reduce processing time
  - Lessens interpolation errors
  - Higher quality texture

Pre-generate smaller textures

Mipmapped Textures

- For the easy of hardware design, textures are downsampled by a factor of 2
  - E.g. 256 x 256 -> 128 x 128 -> 64 x 64
- Many engines (e.g. OpenGL) require the resolution of texture can only be numbers of power of 2

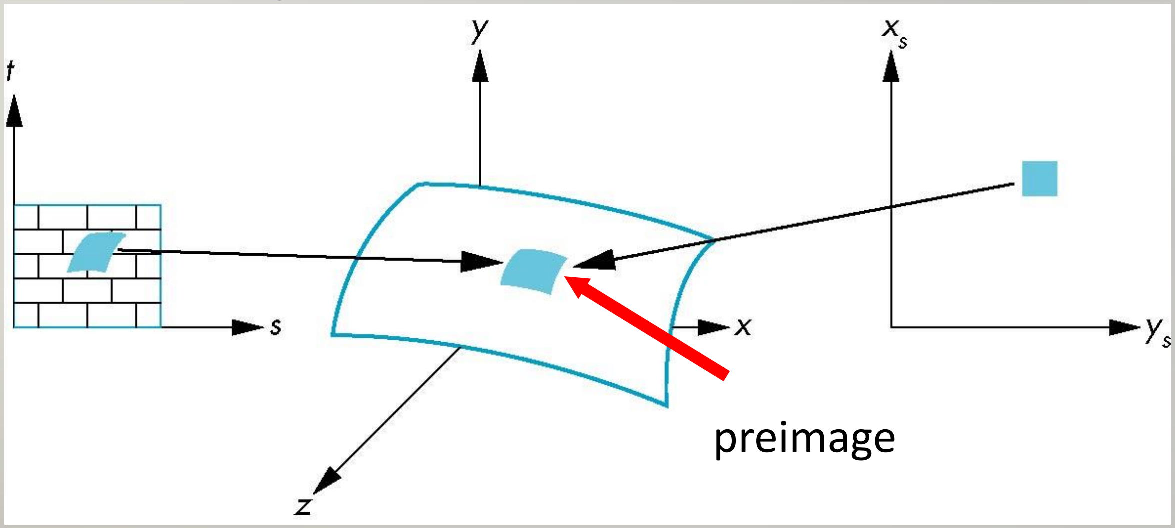


http://game-art.co.uk/agtec/html/mipmaps.html

Mipmapping

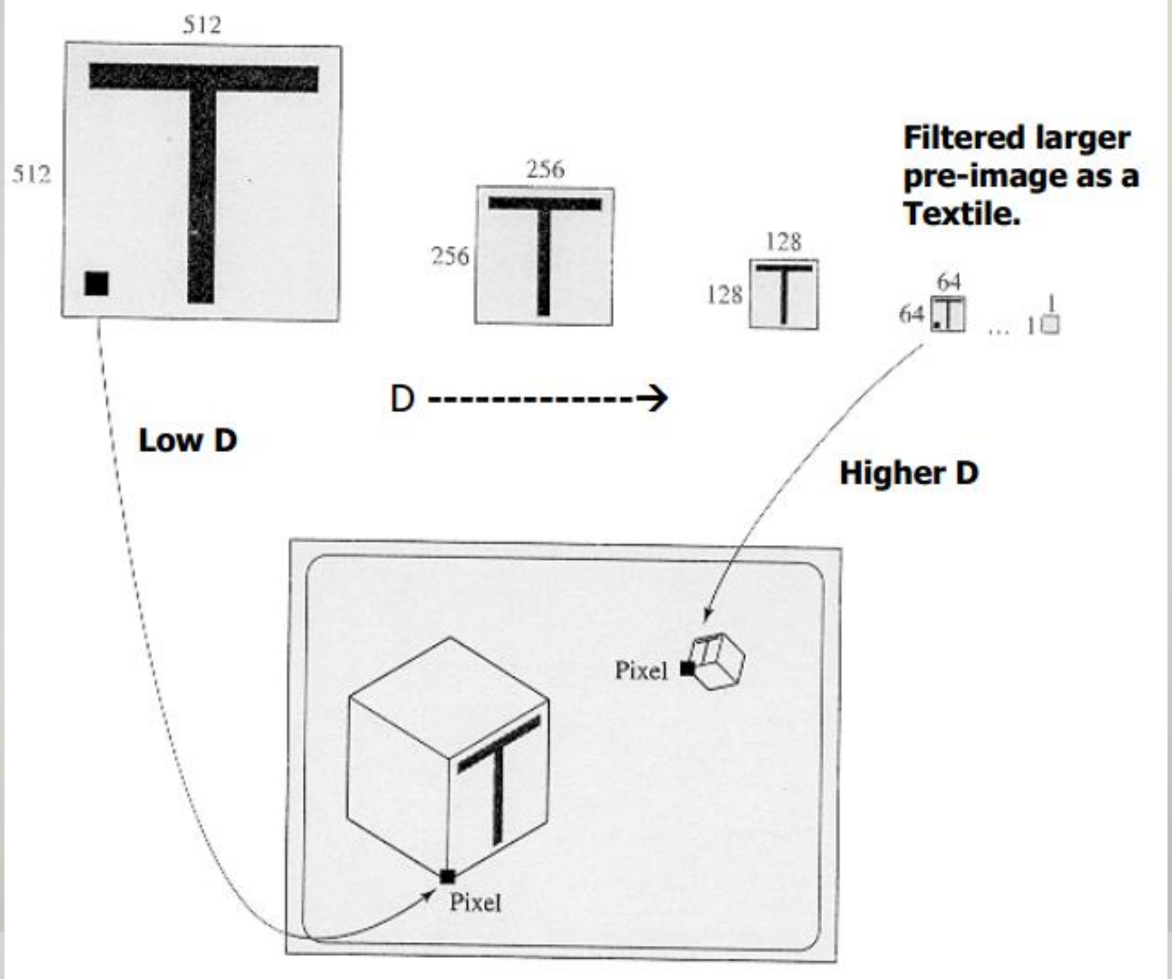
How to select which scaled texture?

- An object near to the viewer, and larger in screen (with smaller pre-image), selects a single texel from a high resolution map
- An object further away from the viewer and smaller in screen (with larger pre-image), selects a single texel from a low resolution map



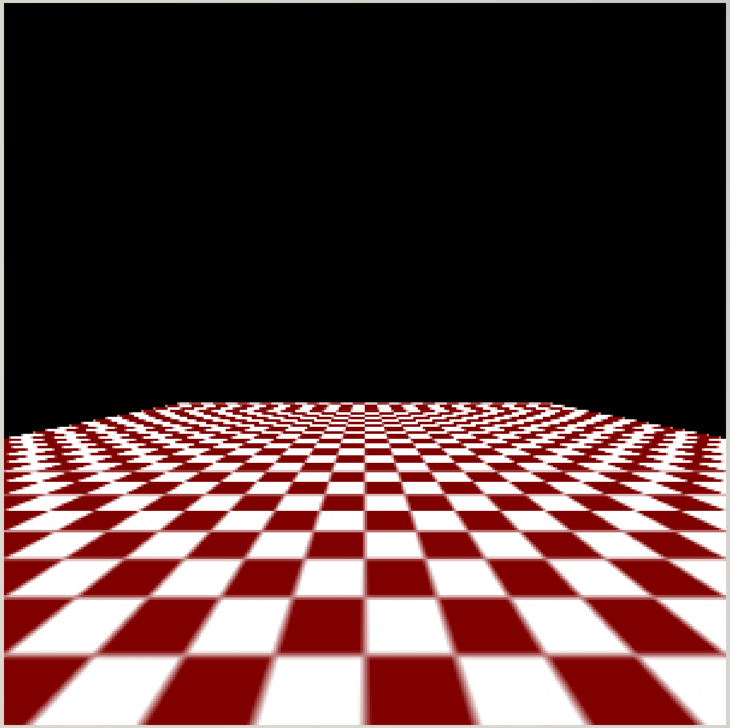
MIpmapping

- By a suitable choice of D (Depth), a texture at appropriate resolution is selected
- The pixel's center is mapped into that map determined by D and this single value is used

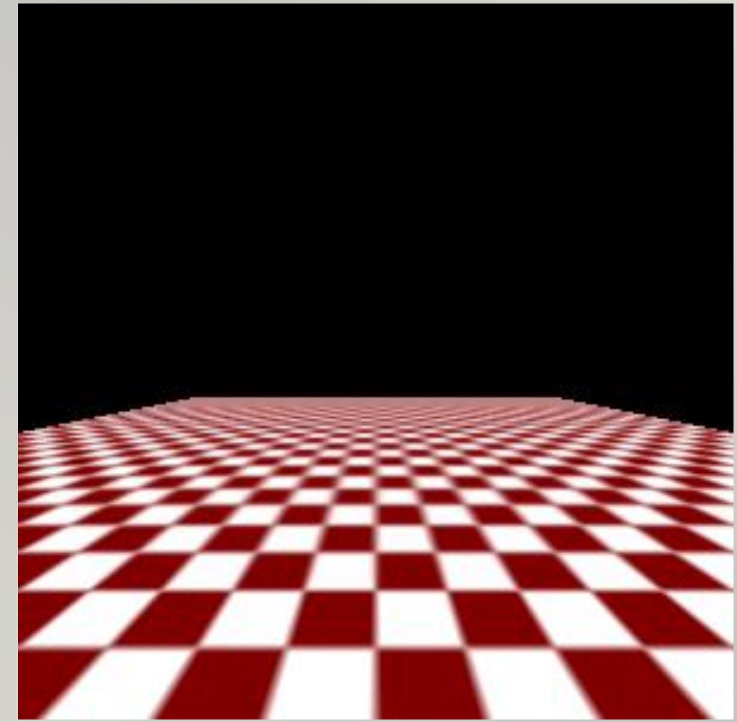


Example

- A checker pattern is textured on a floor



No mipmapping



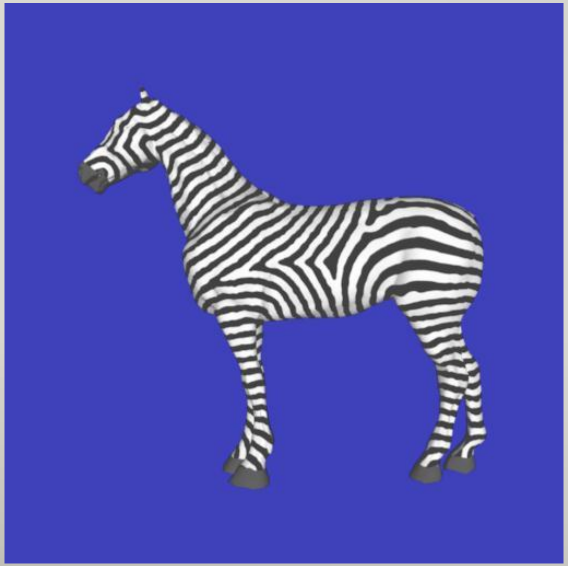
Use mipmapping

Procedural Textures

- Apart from preparing textures by professional artists, some textures can be generated by mathematical equations
- They are referred as **Procedural Textures**
- Usually, these textures are some natural elements
  - E.g. wood, marble, metal, fur and stone

Procedural Textures

- Some methods for procedural textures
  - Fractal noise
  - Turbulence function
  - Reaction Diffusion equation



Reaction Diffusion equation

Fractal noise

Turbulence function

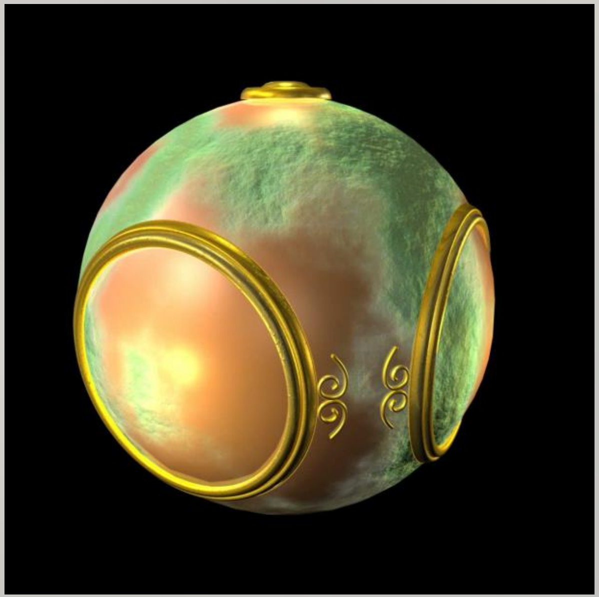
Procedural Textures

- Advantage
  - Prepare new texture pattern easily
  - Can create infinitely large texture by giving proper parameters
- Disadvantage
  - The type of patterns created are limited

    usually natural elements similar to noise pattern
  - The generated pattern is difficult to be controlled by parameters

Advanced Texture Mapping

- Many special effects can be mimicked using texture related techniques
  - Environmental mapping
  - Bump mapping



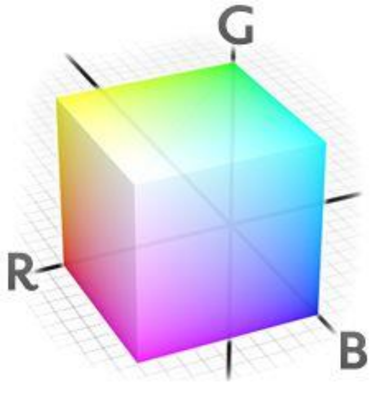
Summary

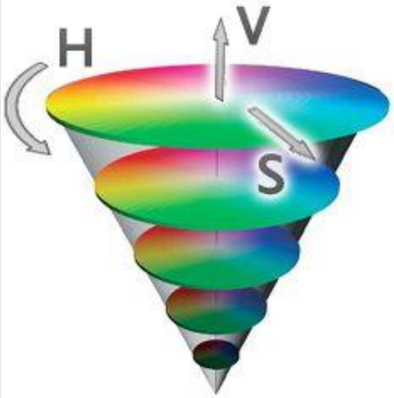
- Texture mapping is commonly used way to add details to object
- We have studied the methods to assign texture coordinates to objects
  - Direct UV mapping
  - Two-stage mapping
- Aliasing may occur when the texture mapped object are far from camera
  - Mipmapping is a standard method in real-time rendering for anti-aliasing

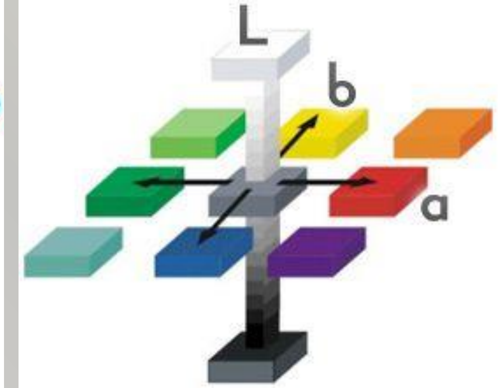
Transparency and Blending

Color Spaces

- Common color models have 3 channels to define colors
- Various color spaces are available
  - RGB
  - YIQ
  - YUV
  - LAB
  - HSV







http://learn.colorotate.org

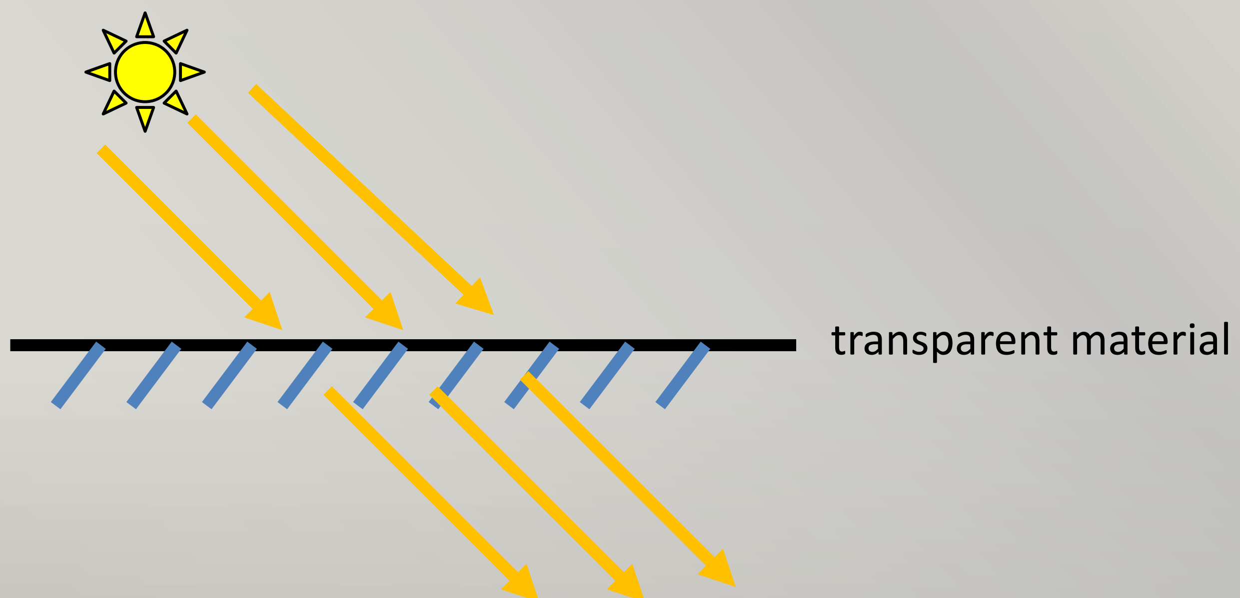
- There are formulas to convert between one another

Alpha Channel

- However, a special extra channel may be included in many graphics application
- It is the alpha channel
  - Defining the opacity of the color
- Opacity vs Transparency
  - They simply mean the opposite

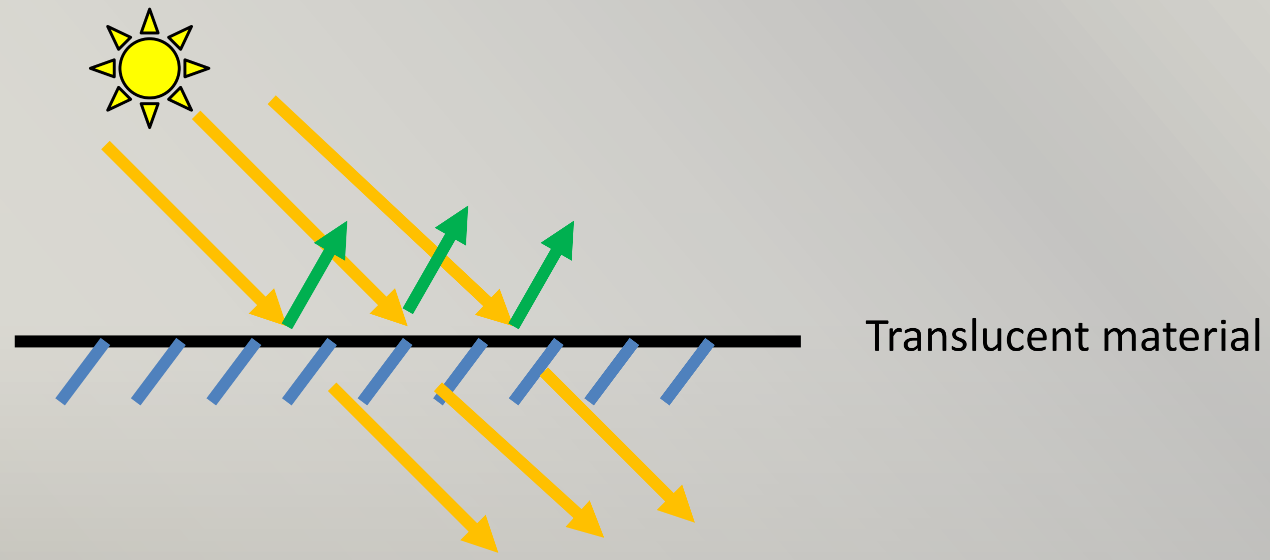
Transparency, Translucency and opacity

- By definition, a transparent material does not reflect light off its surface
  - Clear glass is a nearly transparent material
  - So, perfect transparent material is invisible



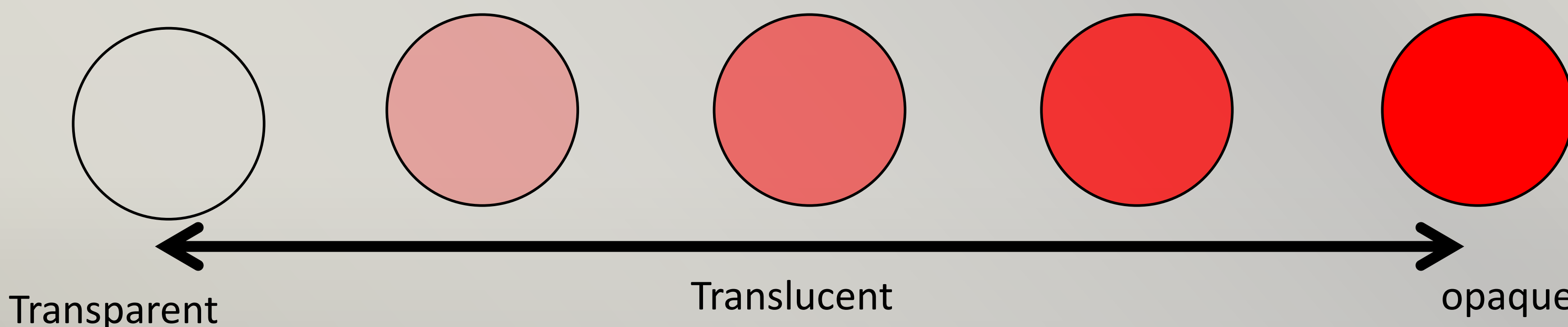
Transparency, Translucency and opacity

- An opaque material will reflect all light from its surface
- Translucent material will partly reflect light off and partly go through its surface



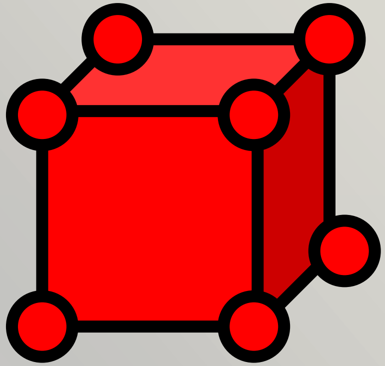
Transparent, Translucent and opaque

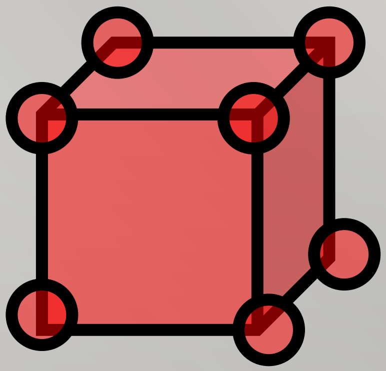
- Alpha value = opacity
  - The higher the value, the more opaque it is
  - The lesser the value, the more transparent it is
- Commonly between 0.0 - 1.0 and come with RGB to form RGBA



Assigning Opacity

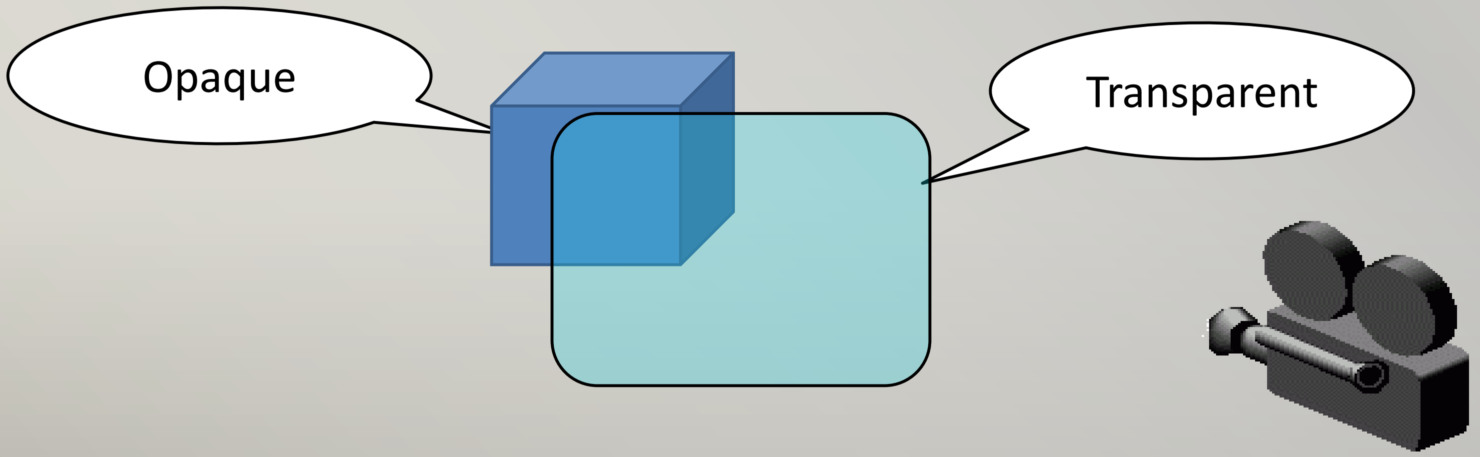
- The same as assigning material or color to object, vertices stored the opacity value
- E.g. for a normal opaque red color
  - (1.0, 0.0, 0.0, 1.0)
  - (R,G,B,A)
- for a semi-transparent red color
  - (1.0, 0.0, 0.0, 0.5)





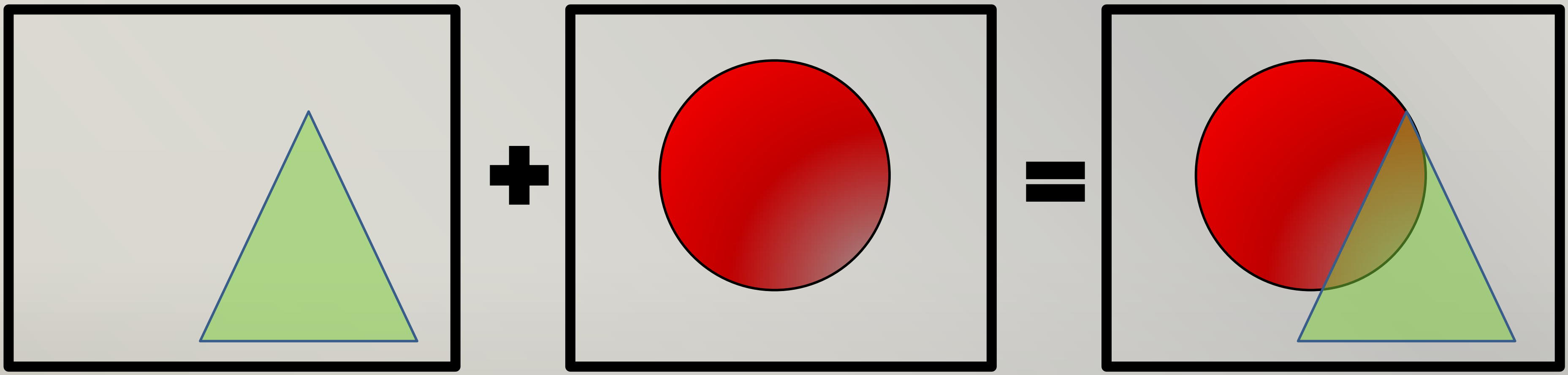
Layers of Objects surfaces

- The effect of transparency will only be seen when multiple objects come together
  - E.g. a transparent object is in front of another opaque object
- We can then see both of them the same time
- It is something like layers of object surfaces



Blending

- The process of combining the colors from different layers of surfaces
  - Source (Incoming)
  - Destination (Current Pixel)
  - Blend (become Destination in next round of blending)



Source (Incoming)

Destination (Current pixel)

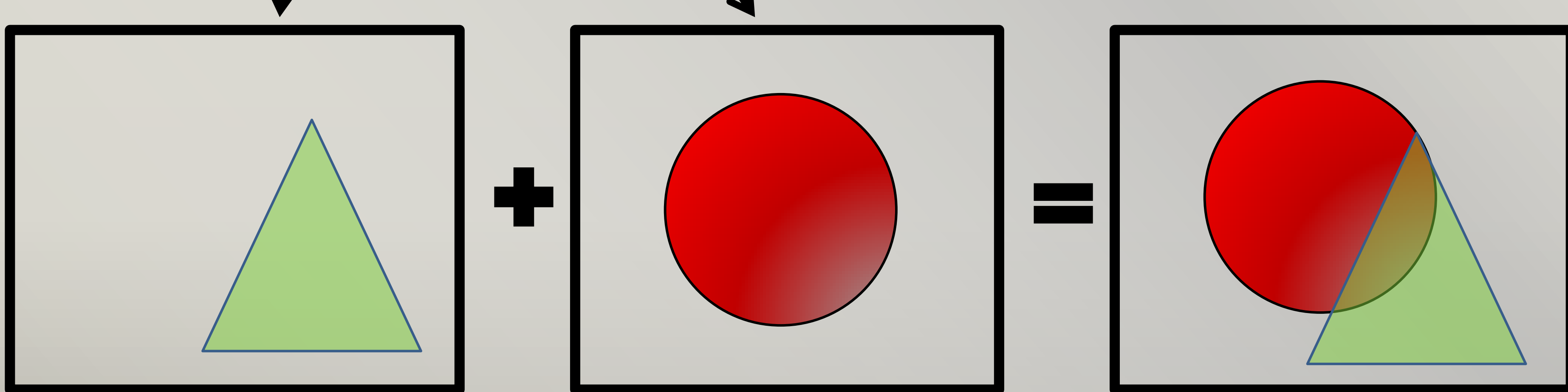
Blend (New current pixel)

Blending Computation

- Consider a particular source and destination pixel:

$$C_{\text{blend}} = t_{source} C_{\text{source}} + t_{destination}\, C_{\text{destination}}$$

Here t_{source} and $t_{destination}$ are blending factors from source and destination respectively



Blending Computation

- One commonly used blending factors
  - $t_{source} = \alpha$ and $t_{destination} = (1-\alpha)$
- Here α is the alpha value of the source
- Therefore, we have **alpha blending equation:**

$$C_{blend} = \alpha\, C_{source} + (1-\alpha)\, C_{destination}$$

- What is does it the same as **linearly interpolate** between the two colors with the alpha value
  - The higher the alpha value is, the more the source color contributes

Blending Computation

- A numerical example:

$C_{\text{source}} = (0.7, 0.5, 0.3, 0.4), \;\; C_{\text{destination}} = (1.0, 0.4, 0.3, 1.0)$

- So, $\alpha = 0.4$
- As $C_{\text{blend}} = \alpha\, C_{\text{source}} + (1 - \alpha)\, C_{\text{destination}}$

- $C_{\text{blend}} = 0.4 \; * (0.7, 0.5, 0.3) + 0.6 * (1.0, 0.4, 0.3)$

  $C_{\text{blend}} = (0.28 + 0.6, 0.2 + 0.24, 0.12 + 0.18)$

  $= (0.88, 0.44, 0.3)$

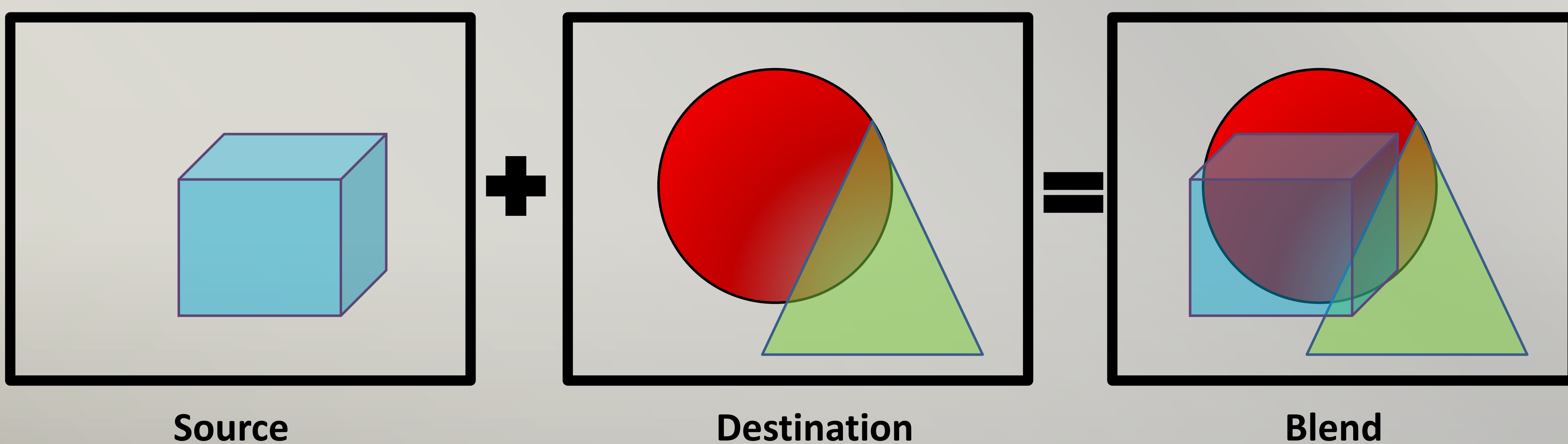
Different Types of Blending

- Apart from alpha blending, there are other types of blending by having different values of t_{source} and $t_{destination}$
  - Additive Blending : $t_{source} = 1$, $t_{destination} = 1$
  - Multiplicative Blending : $t_{source} = 0$, $t_{destination} = C_{source}$
  - 2X Multiplicative Blending : $t_{source} = C_{destination}$, $t_{destination} = C_{source}$

Msdn.microsoft.com

Blending Of Multiple Objects

- For multiple objects, the blending is simply repeating the source and destination blending for more times
  - Consider the blended result in last example, and now becomes destination

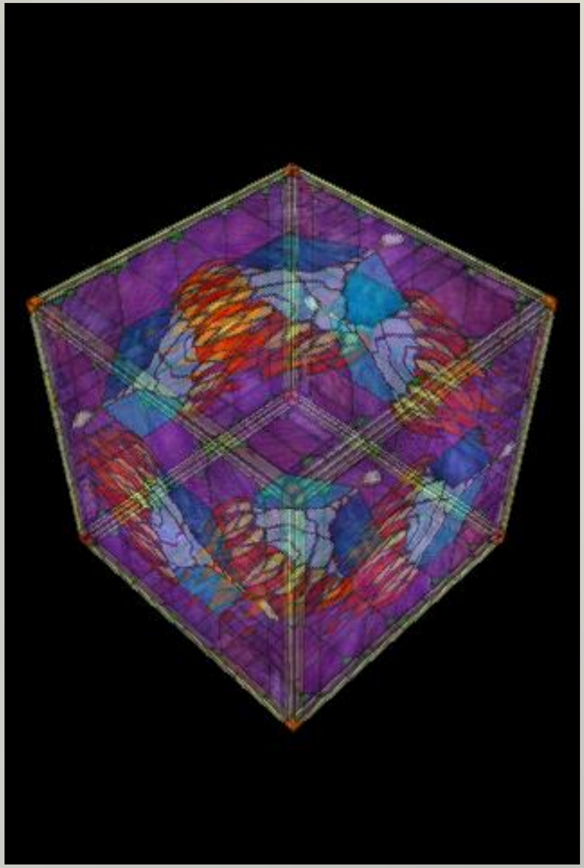


Blending In OpenGL

- In OpenGL, choices of blending factors:
  - GL_ZERO, GL_ONE,
  - GL_SRC_COLOR, GL_ONE_MINUS_SRC_COLOR,
  - GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA
- To set the blending factors, we invoke :
  - glBlendFunc (GLenum source, GLenum destination);
- For the case of alpha blending, we have
  - glBlendFunc (GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA)

Blending In OpenGL

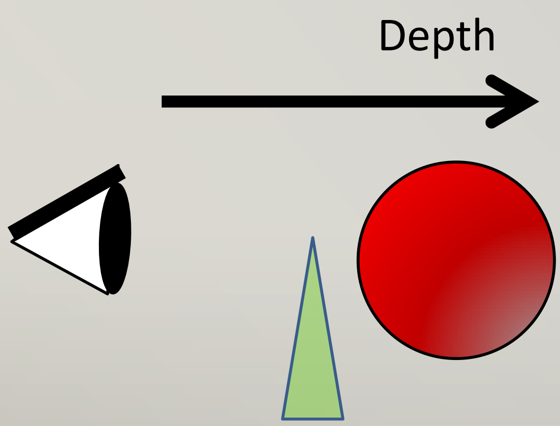
- The alpha blending effect in OpenGL can be used for making glass like materials
  - E.g. a glass box (right figure)
  - You are strongly recommended to take a look of Nehe’s blending examples for more details



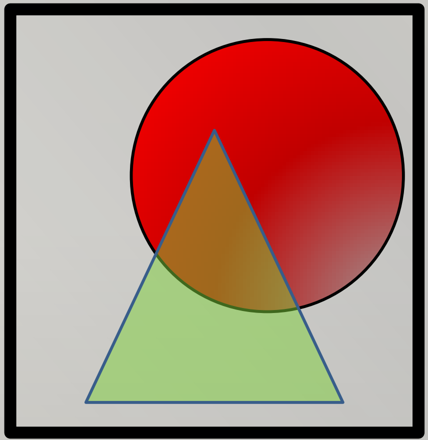
http://nehe.gamedev.net/tutorial/blending/16001/

Drawing Order

- To ensure blending takes effect as expected, the drawing order of object surface is important
- Consider a case that a transparent object is before an opaque object
  - Expected output is as shown on right



Expected Output

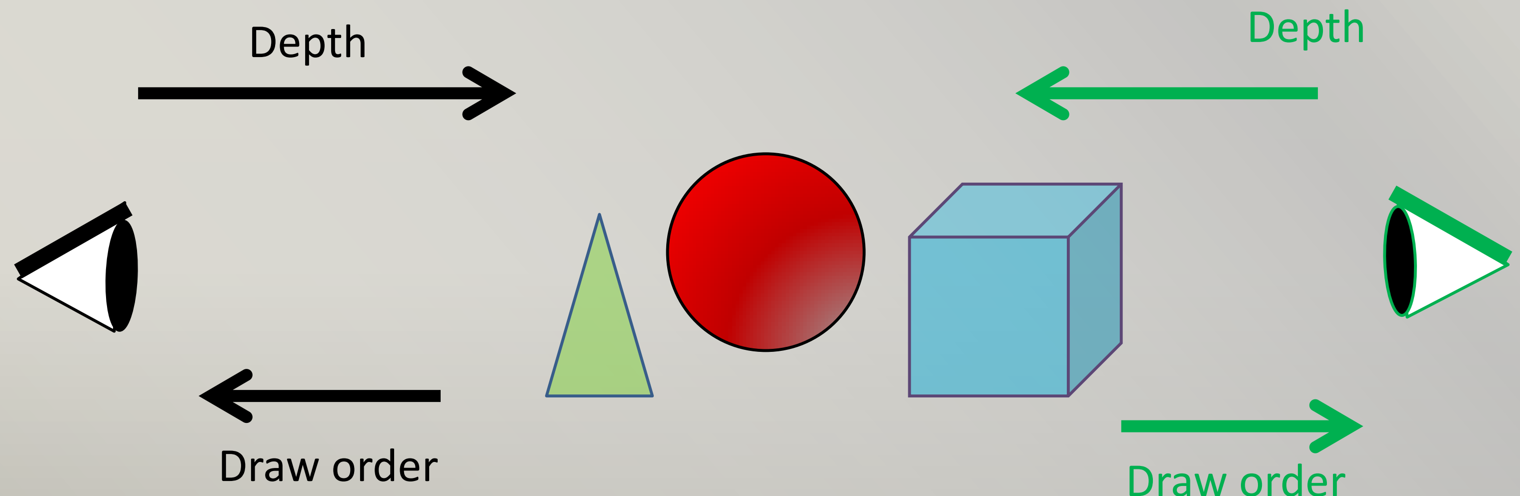


Drawing Order

- However, if we first draw a nearer transparent object before a farther opaque object
  - The transparent object becomes **“Destination”**
  - Opaque object becomes **“Source”**
- Result is **NOT** what we expected !!!

Drawing Order

- Therefore, the drawing order is important
- Drawing order depends on **depth**
  - Draw from farther objects/surfaces to nearer ones
- And the depth is **view dependent**

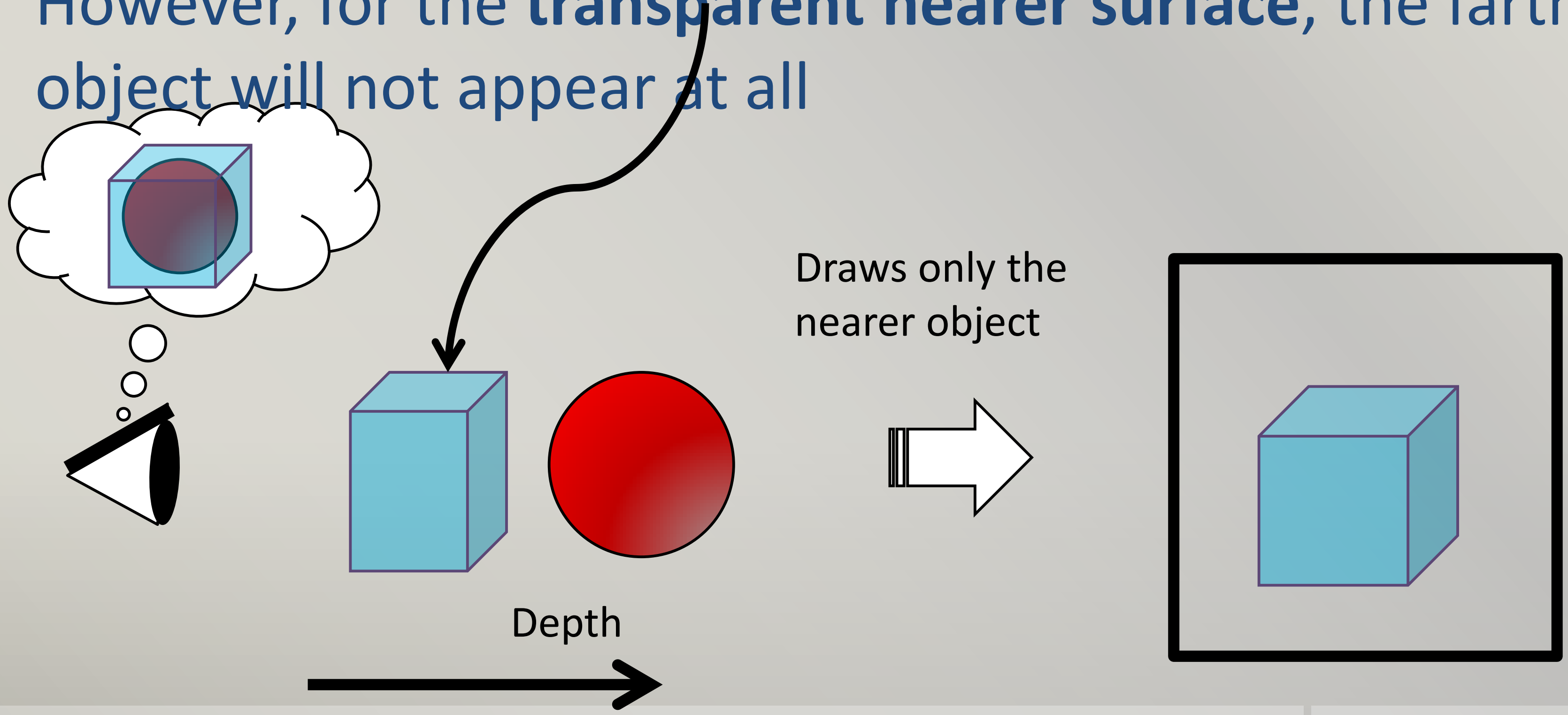


Hidden Surface Removal

- Most real-time rendering engine will perform hidden surface removal to render faster
  - Usually hidden surface removal is performed with the use of depth buffer (also called Z-buffer)
  - But we will leave the details on depth buffer in coming lessons
- However, we can not perform hidden surface removal when blending of objects has to be done

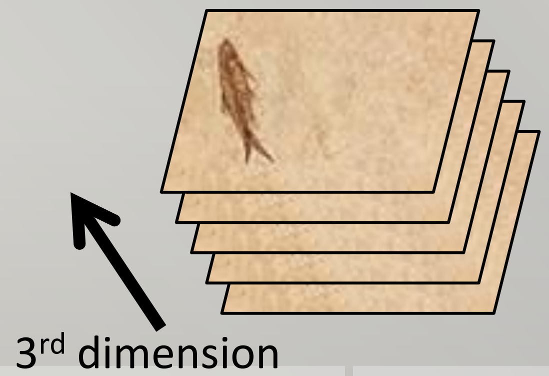
Hidden Surface Removal

- The basic idea of hidden surface removal is to **skip** surfaces that is covered by other nearer surfaces
  - This works if the nearer surface is opaque
- However, for the **transparent nearer surface**, the farther object will not appear at all



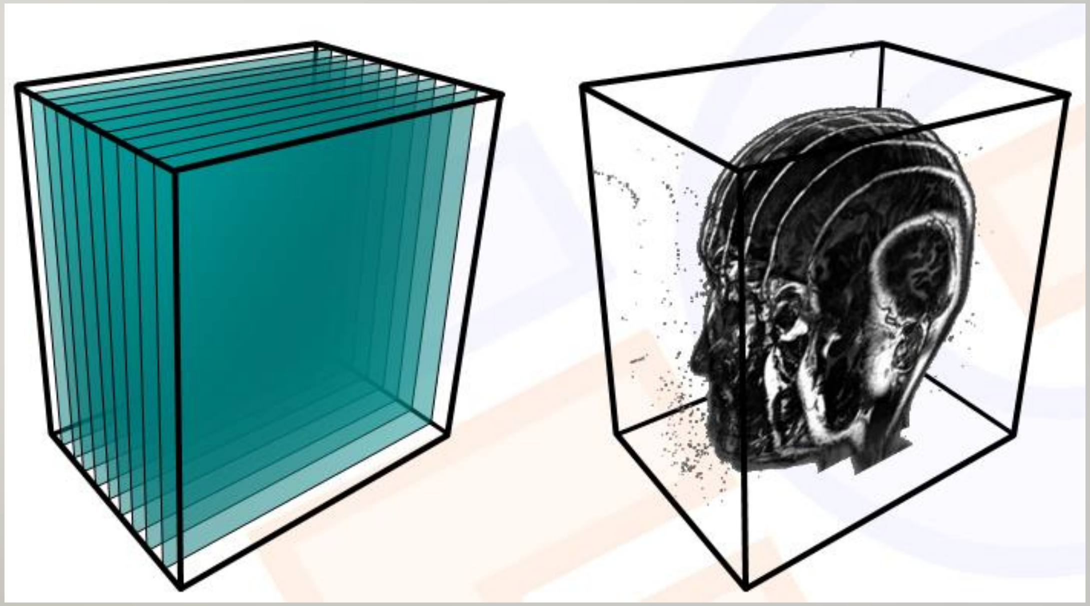
Volume Rendering

- The blending of surfaces is important for rendering of volume data (or 3D Texture)
- Volume data is just like an image with one more dimension
  - But don’t mix it with 3D/Stereo image
- You can imagine it is formed by tightly stacking many images in the 3rd dimension
  - So a volume is formed
  - One element in volume data is called **voxel**



Volume Rendering

- But rendering of volume is not as trivial as 2D image
  - Since there is no volumetric primitives for rendering
- We create a stack of 2D quads, and each of them texture with one slice of the volume
  - Finally, quads are blended together to render the result



A stack of 2D quads Blending of textured quads

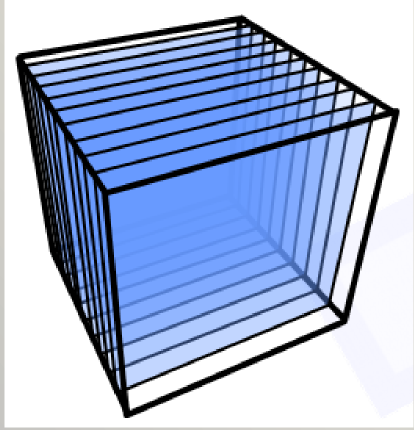
Slicing

- We have to make clear that the slices is not the same as the stack of images
- Two slicing schemes are available
  - Axis-aligned slicing

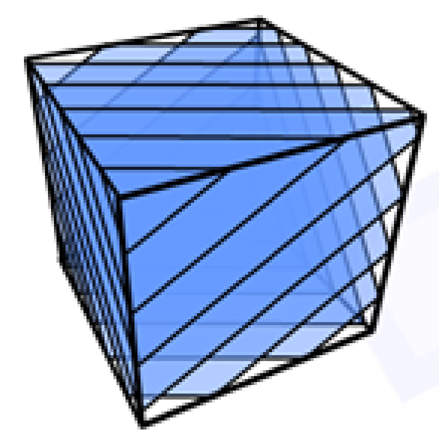
    Not view dependent
  - View-aligned slicing

    Slicing parallel to the image plane

    View dependent



Axis-aligned slicing



View-aligned slicing

Applications of Volume Rendering

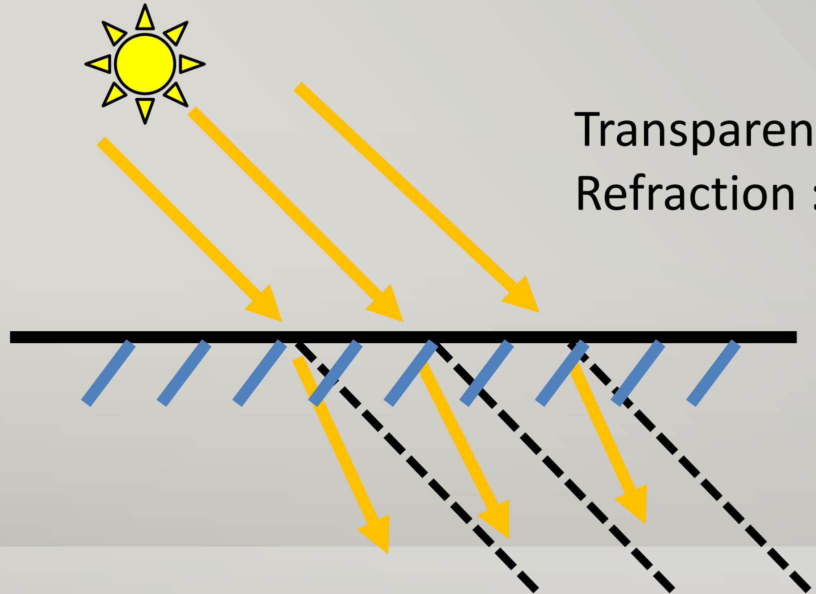
- Medical visualization involves many volume data
  - CT, MRI and 3D Ultrasound Images
- Volume rendering is used in rendering of **smoke, cloud** or other similar effects

GPU GEMS

GPU GEMS

Transparency vs Refraction

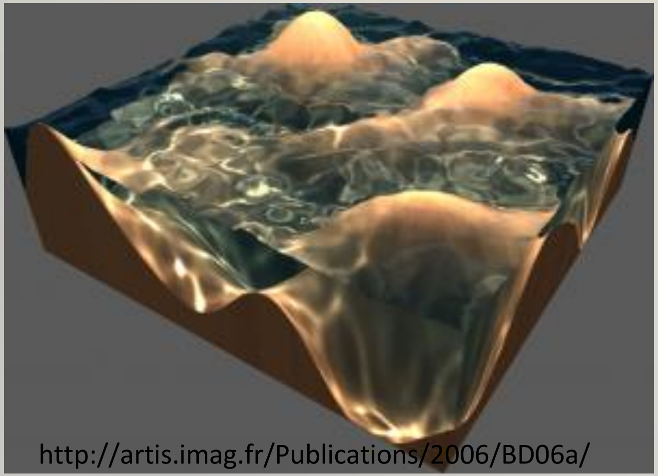
- In real world, usually transparency and refraction will happen at the same time
- Since, real materials like glass or water surface will let light pass through and change the path of light at the same time



Transparency : Light pass through
Refraction : Path of light changed

Transparency vs Refraction

- Alpha blending alone can only achieve transparency
- To simulate refraction is more complicated
  - As computation involve the rate of light bended on a surface and also the surface orientation
  - GPU can assist to perform this in real-time now



http://artis.imag.fr/Publications/2006/BD06a/

GPU GEMS

Summary

- In addition to RGB, alpha channel is added to the end to represent the opacity
- Alpha blending is one kind of blending commonly used in graphics
  - It can be used for transparent material
- Blending equation with different blending factors forms different kinds of blending
- Volume rendering heavily depends on alpha blending slices of textured quads