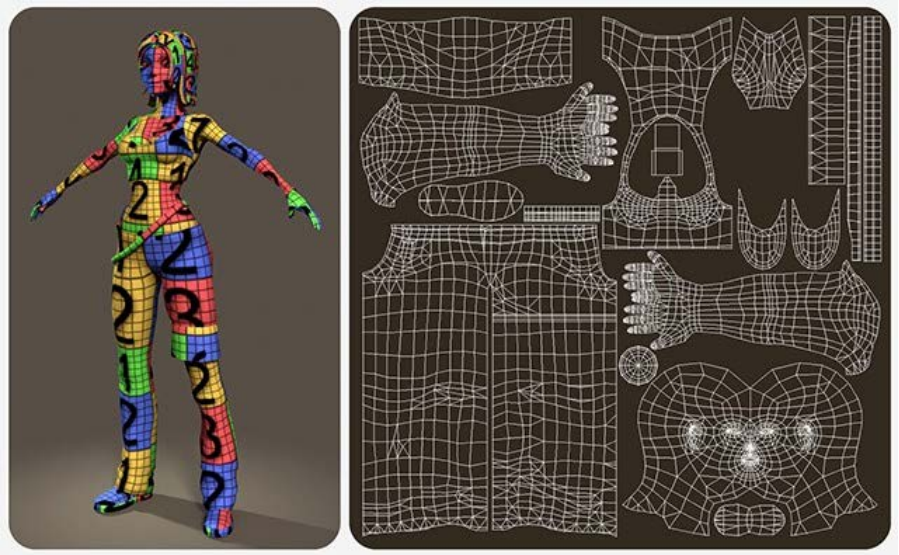
Computer Graphics: Texturing

Dept. of Game Software
Yejin Kim

Overview

- Texture Mapping
- Texture Filtering
- Tutorials



Texture Mapping

- Motivations
  - 3D scenes created with diffuse lighting look flat and cartoonish
  - Adding more details(polygons) on the object surface → Expensive
  - Texture mapping pastes images onto the object surfaces in the scene, adding realistic fine detail without exploding the geometry

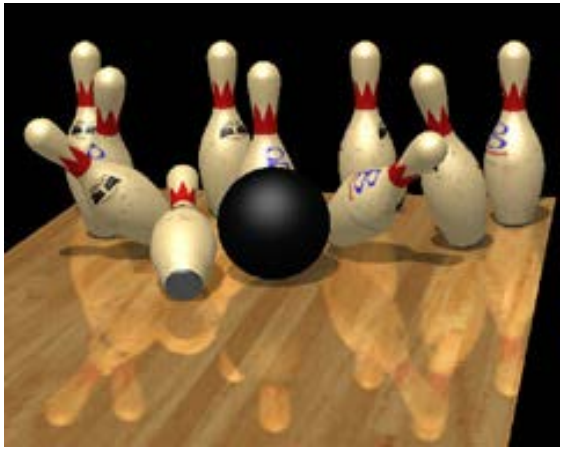
Virtual Fighter (1993)

Virtual Fighter 2 (1994)

Texture Mapping

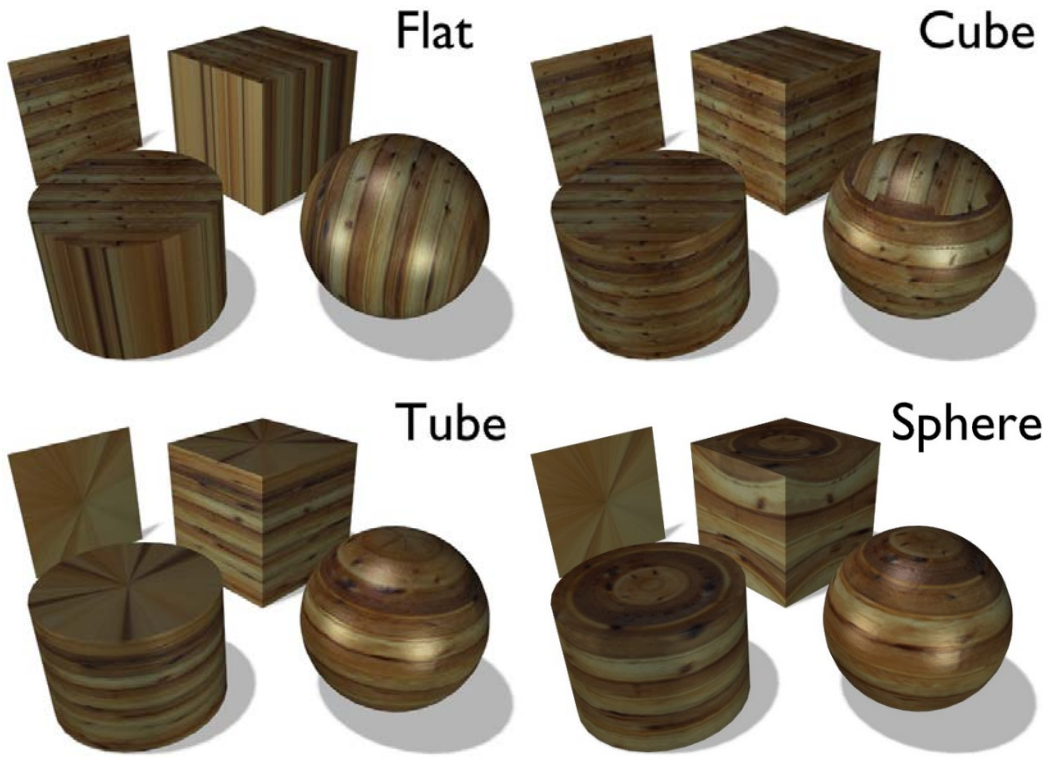
- Texture (diffuse) mapping *[Edwin Catmul, 74]*
  - A method to wrap a 2D image on a 3D object surface
  - Use images to capture the texture of surfaces

- Texture map
  - An image(a 2D array of color values) applied to the polygon surface
  - Define high frequency details on the object surface
    - Color or reflectance (diffuse, ambient, specular)
    - Transparency (smoke effects)
  - Many techniques
    - Multitexturing, mipmaps, normal/bump/light mapping, etc.



Texture Mapping

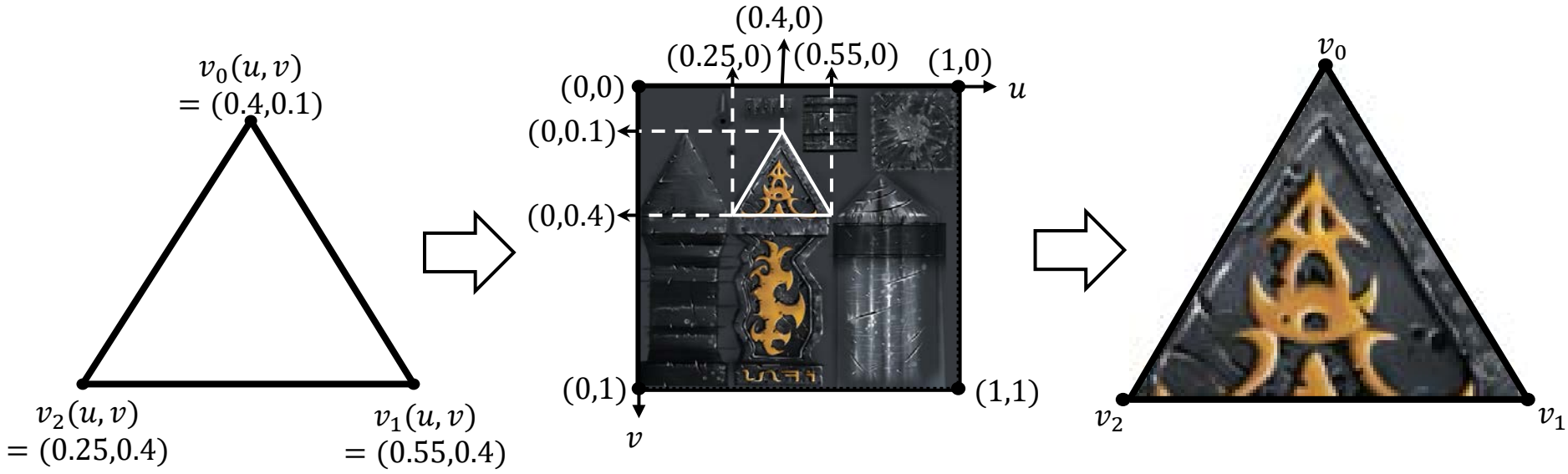
- Different mapping methods
  - **Flat**, cube, cylinder, sphere, etc.
  - Using triangulated meshes reduces the problem to mapping a portion of the image to each **triangle**



Texture Mapping

- Texel (texture pixel)
  - Fundamental unit of a texture map
  - Value stored at a texel affects surface appearance
    - Diffuse reflectance, shininess, transparency, etc.

- Mapping the texture to an triangular(flat) shape
  - Each texel is specified by texture coordinate (*u, v*) = [0, 1]
  - Map an *uv*-space to polygon's space



Texture Mapping

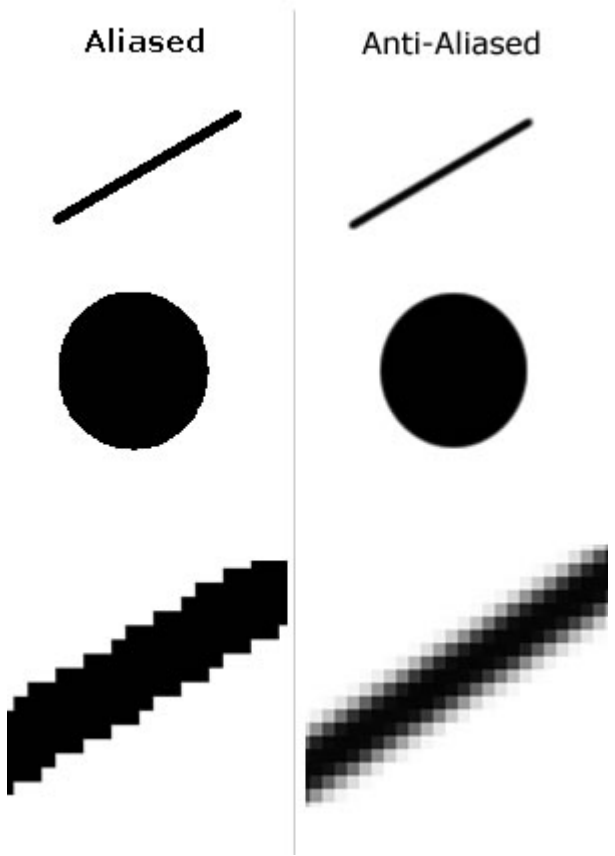
- User-generated mappings
  - For complex 3D objects, mapping textures is a field of artistic design

Lara Croft: Tomb Raider, 2001

Texture Mapping

- Aliasing effect
  - The appearance of jagged edges(jaggies) in a rasterized image
  - Occurs when real-world objects which comprise smooth, continuous curves are rasterized using pixels
  - Due to rasterization with sampling at a low frequency(undersampling)

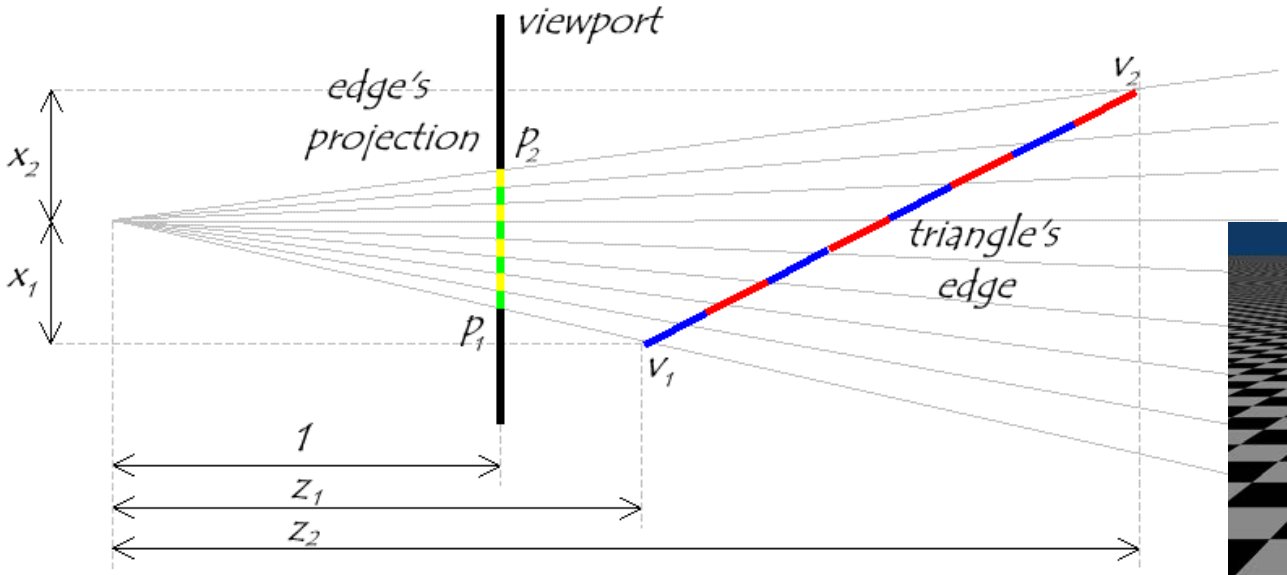
- Antialiasing
  - A technique used in computer graphics to remove the aliasing effect

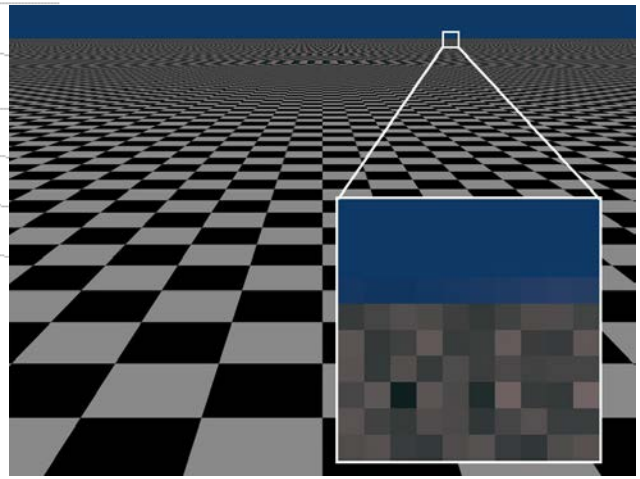


Texture Mapping

- Aliasing effect in texture mapping
  - Warping at the edges of triangles making up the mesh
  - Uniform steps in screen space ≠ uniform steps in world coordinates

  → Use texture filters

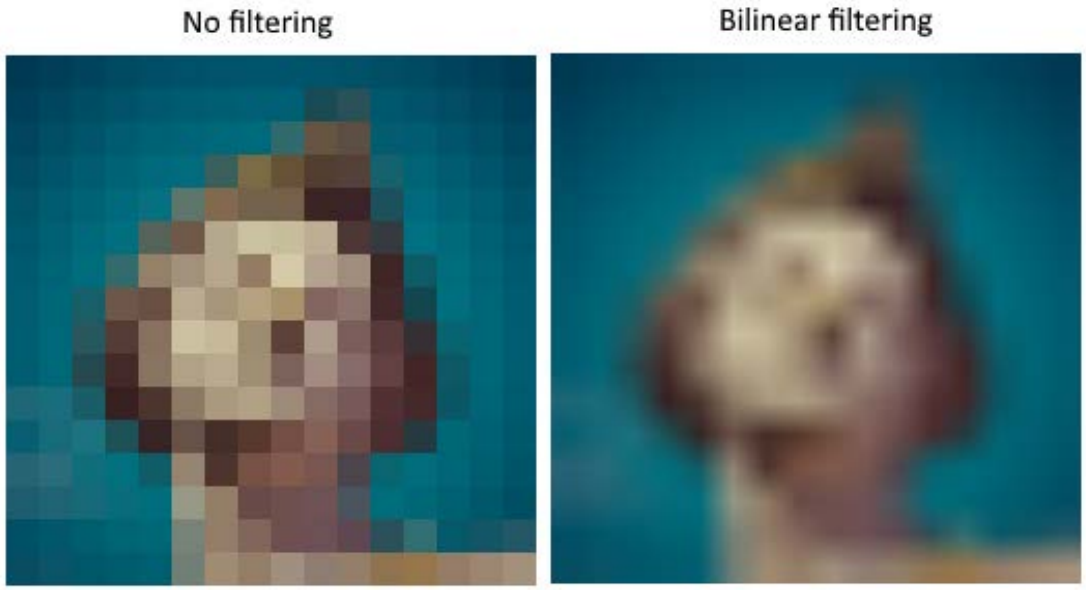




Moire(모아레) pattern

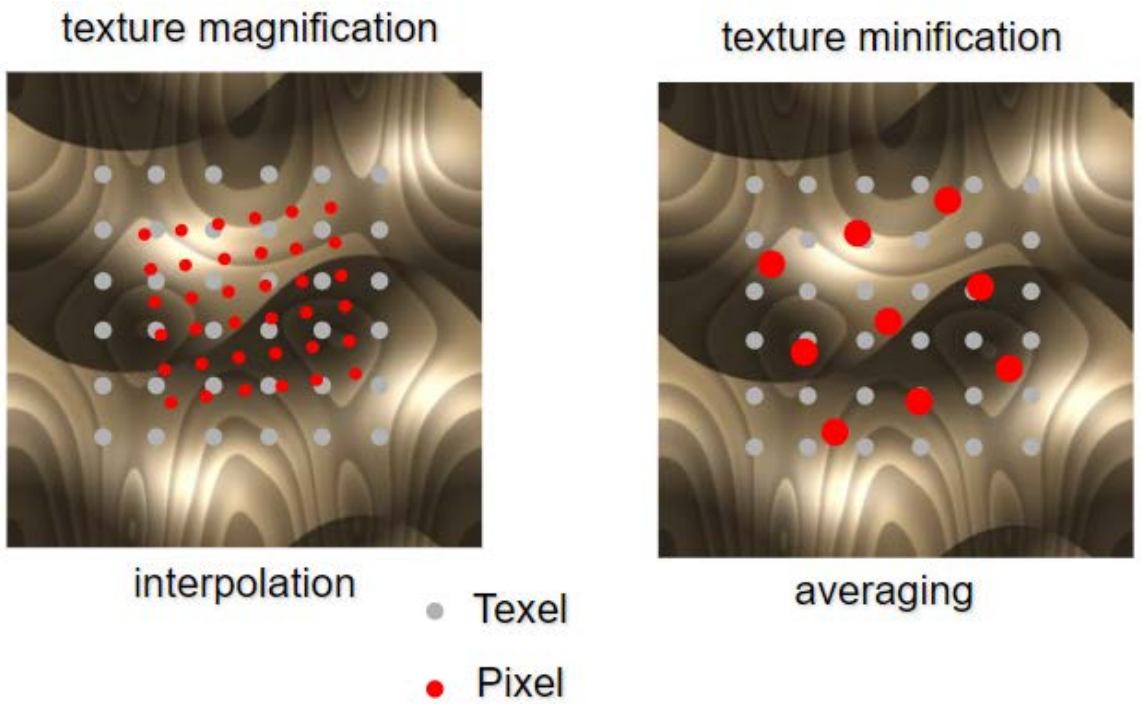
Texture Filtering

- Texture filtering (smoothing)
  - A method used to determine the texture color for a texture mapped pixel using the colors of nearby(neighbor) texels
  - Problem: size of one pixel ≠ size of one texel
  - Even if pixel size ≈ texel size, a pixel will in general fall between nearby texels



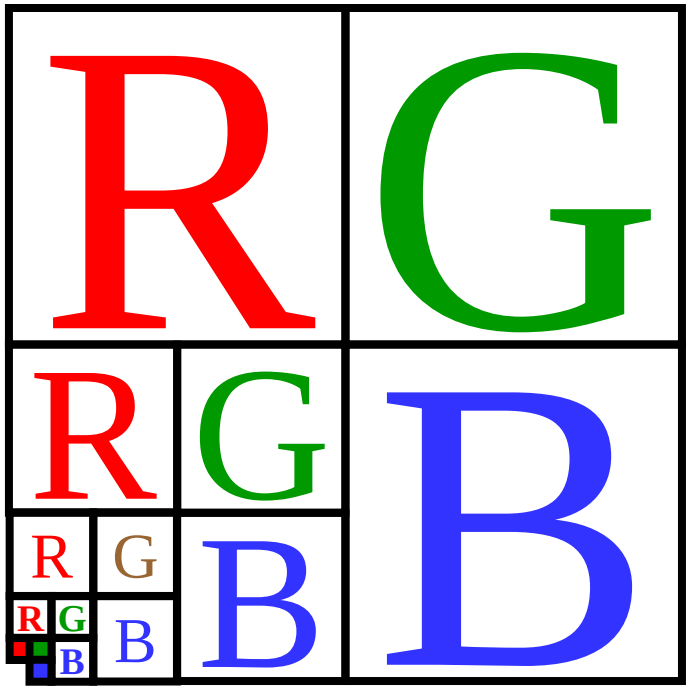
Texture Filtering

- Magnification and minification filters
  - If a pixel < a texel, interpolate between texel values → a texture appears **magnified**
  - If a pixel > a texel, average the contribution from multiple texels → a texture appears **minified**



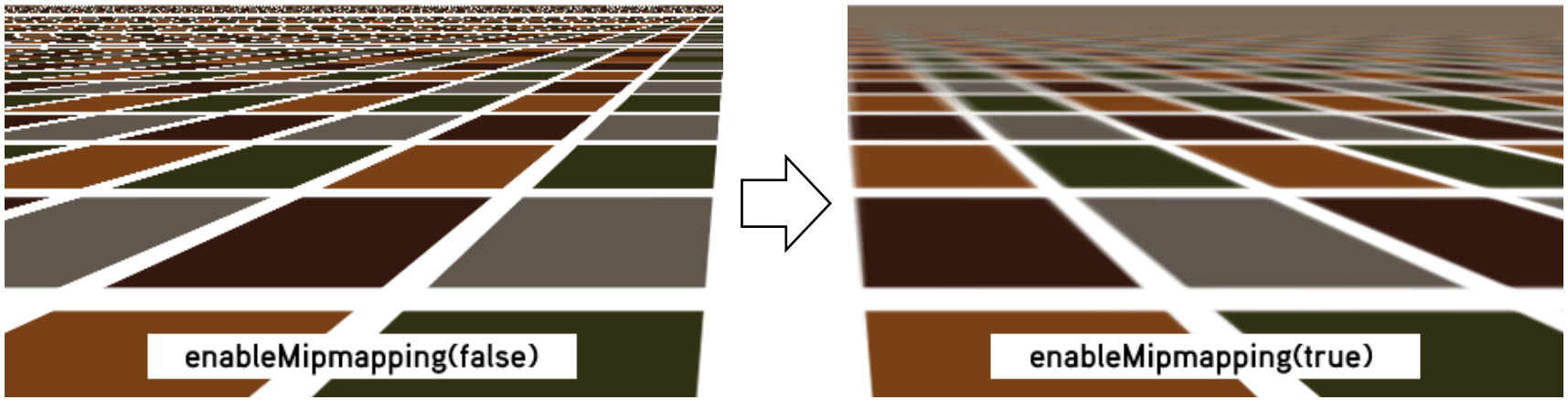
Texture Filtering

- MIP mapping
  - MIP: Multum In Parvo (many things in a small place)
  - Use a collection of multi-resolution images
  - Apply different sizes (levels) of the MIP-map images depending on the distance between the camera and object
  - Texture minification filter to avoid aliasing(Moire, flickering, texture crawling, etc.) effects



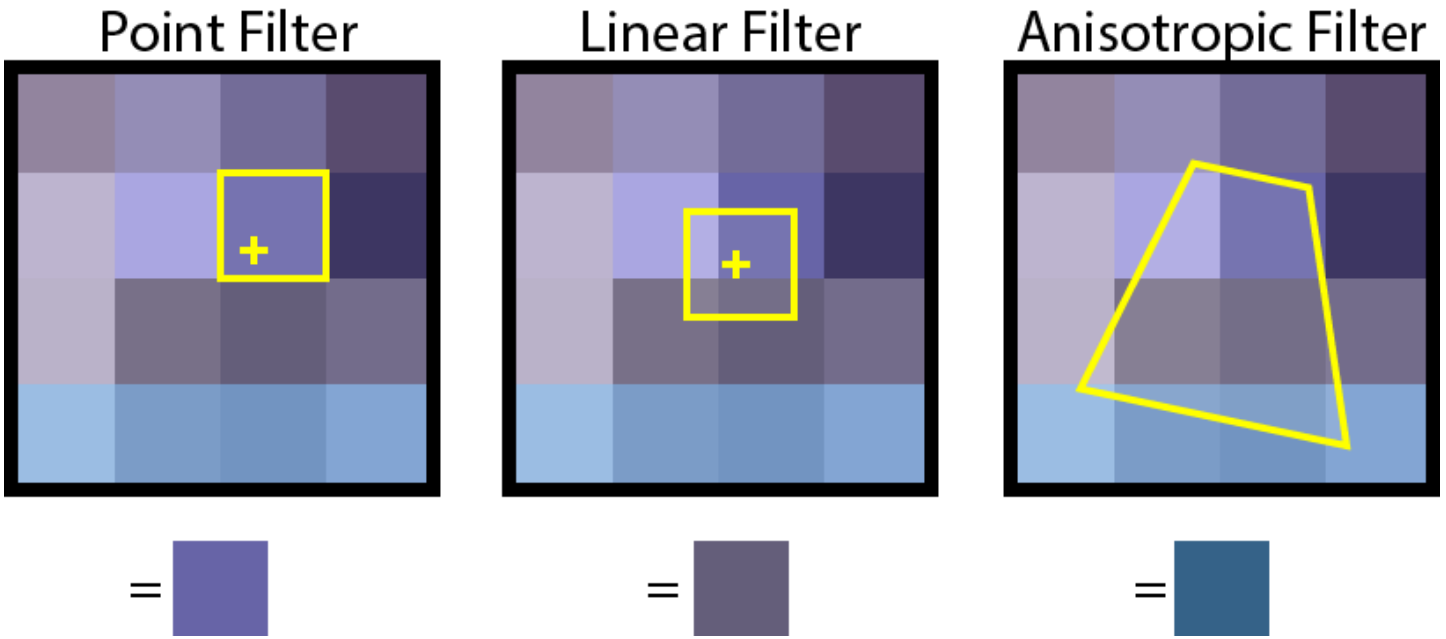
Texture Filtering

- Using MIP mapping
  - Each level of the MIP-map represents a pre-blurred version of multiple texels
    - A texel at level n represents 2^n original texels
  - When rendering,
    - Figure out the texture coverage of the pixel
      - The size of the pixel in texels of the original map
    - Find the level of the MIP map in which texels average approximately that many original texels
    - Interpolate the value of the four nearest texels



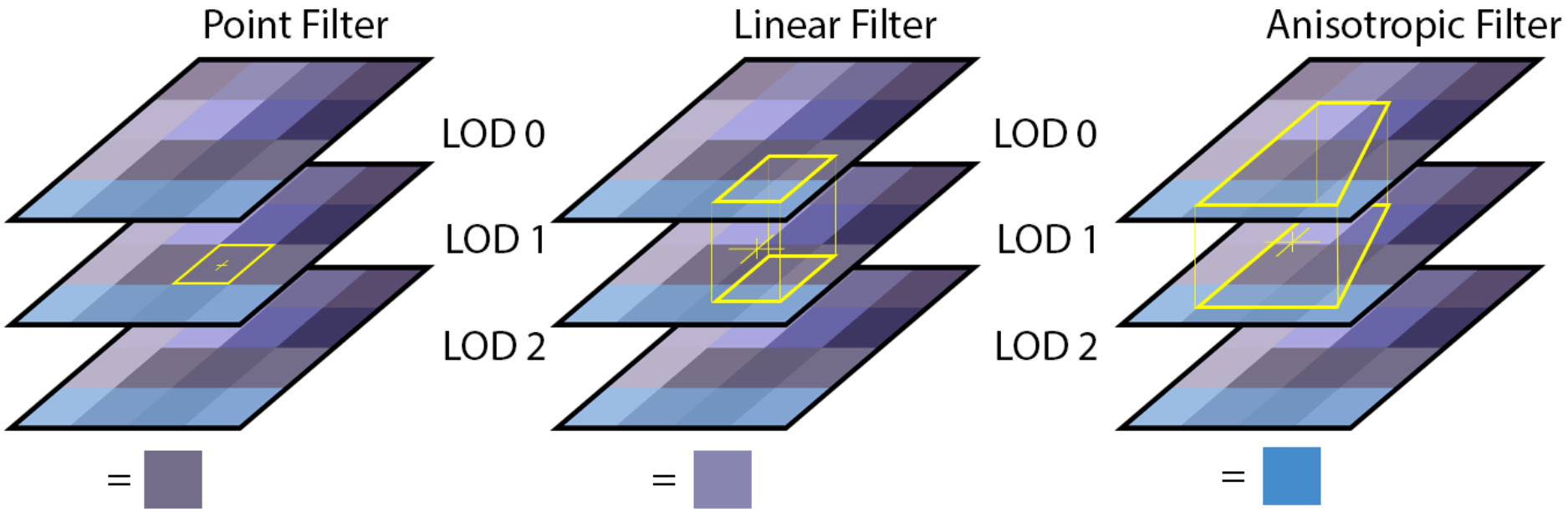
Texture Filtering

- Filtering methods
  - Point: uses the color of the texel closest to the pixel center
  - (Bi)linear: uses the weighted average color of the four nearest texels to the pixel center
  - Anisotropic: sampling the texture as a non-square shape
    - Anisotropy(비등방성): the property of being directionally dependent



Texture Filtering

- Filters with MIP mapping
  - (Tri)linear: uses a color value between two adjacent levels of the MIP map, which is the third interpolated value from two bilinear interpolations

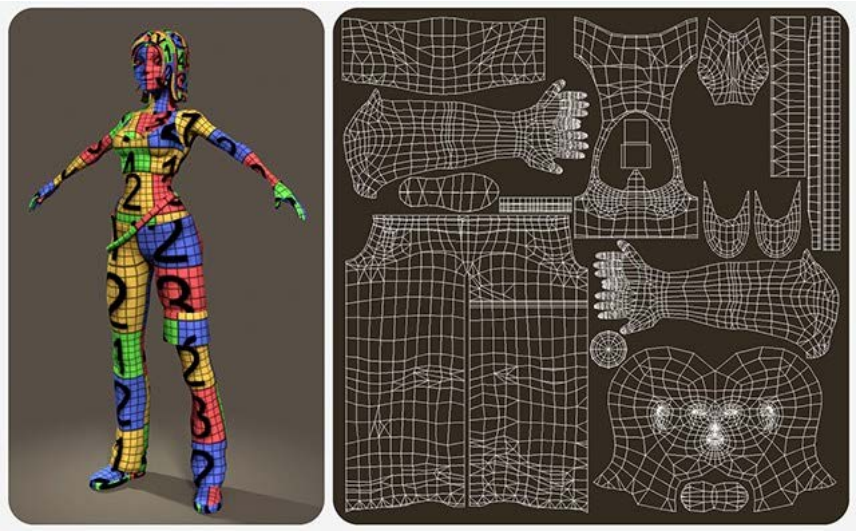


Texture Filtering

- Anisotropic filtering effects
  - Mapped texel is NOT perfectly square in the rendered space
  - Reducing blur and preserving detail at extreme viewing angles

Tutorials

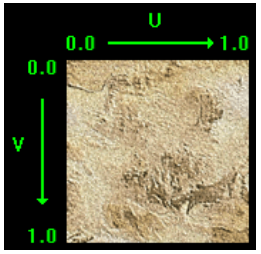
- Texture Mapping
- Texturing 3D Model

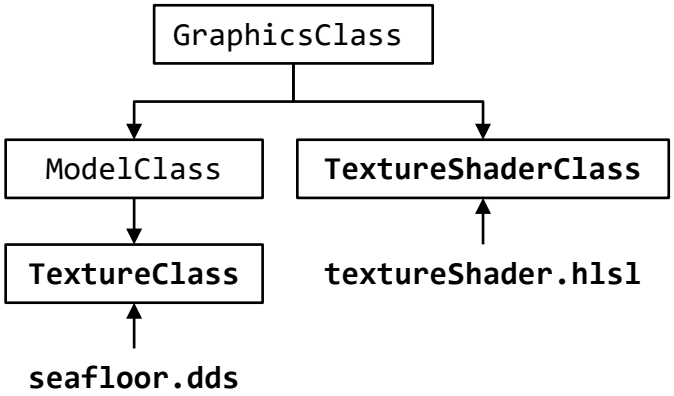


- 2D Image
- Text with Font Image
- Text with DirectWrite
- Billboarding

3-1 Texturing Mapping

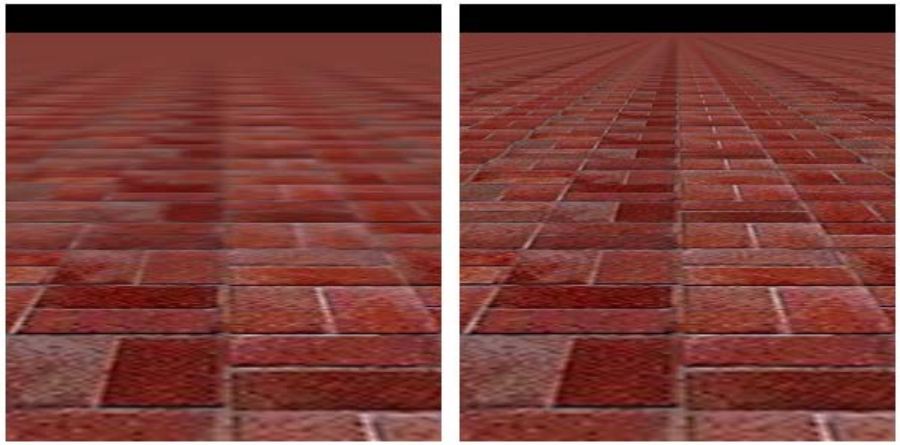
- Adding texture classes to the Framework
  - TextureClass: handle a single texture resource
  - TextureShaderClass: render a texture using HLSL
  - *.hlsl: a vertex and pixel shader file for texturing polygon
  - *.dds: image file format for DirectX
    - Use Visual Studio to convert from other image formats, jpeg, png, etc.





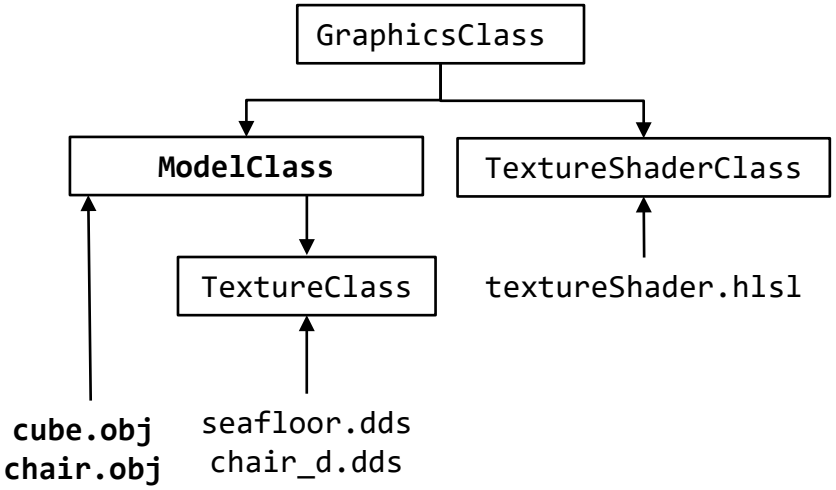
3-1 Texturing Mapping

- Texture filters with MIP mapping
  - Point filter: D3D11_FILTER_MIN_MAG_MIP_POINT
  - Bilinear filter: D3D11_FILTER_MIN_MAG_LINEAR_MIP_POINT
  - Trilinear filter: D3D11_FILTER_MIN_MAG_MIP_LINEAR
  - Anisotropic: D3D11_FILTER_ANISOTROPIC
    - MaxAnisotropy = 2, 4, 8, 16



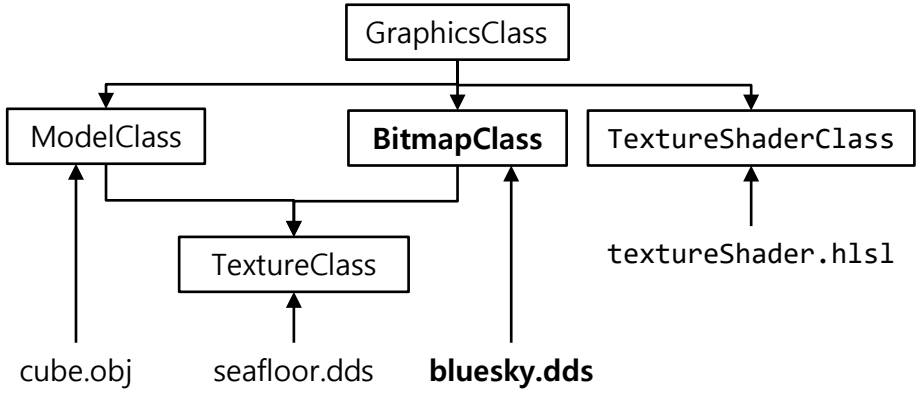
3-2 Texturing Mapping

- Applying a texture image to a 3D model
  - ModelClass: loads 3D model data from a model file
  - *.obj: a 3D model file



3-3 Image

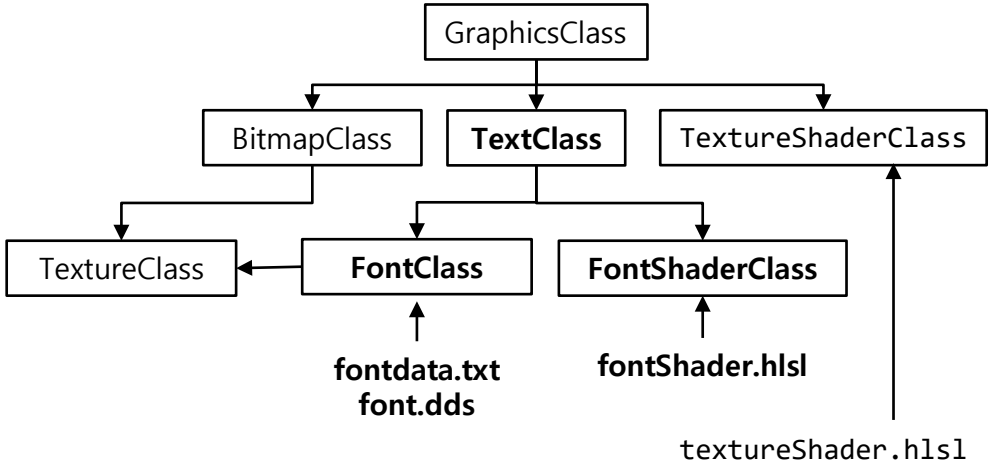
- Draw a 2D image from an external file
  - **BitmapClass**: draw a 2D image on a quad polygon
  - Use a orthogonal (parallel) matrix for projecting an image
  - Turn on/off a Z (depth) buffer for rendering an image



3-4 Text with Font Image

- Draw texts using a font image file
  - TextClass: handles the 2D text drawing process
  - FontClass: handles the texture for the font data
  - FontShaderClass: renders fonts using HLSL

- Uses two different shaders



3-4 Text with Font Image

- Font image: font.dds (1024x16)

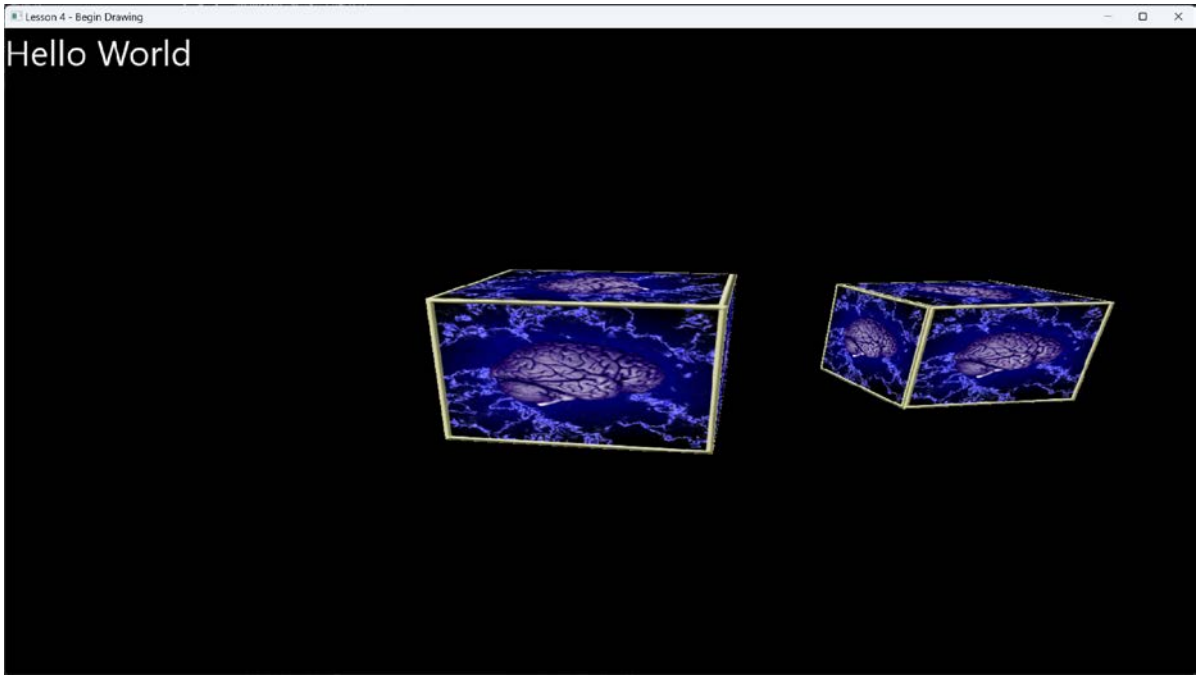
- Font data: fontdata.TXT
  - Location & size (pixels) of each character
  - Format: [Ascii value of character] [The character] [Left Texture U coordinate] [Right Texture U Coordinate] [Pixel Width of Character]

- Rendering notes
  - Turn on/off a depth buffer
  - Turn on/off alpha blending for transparency

```
32   0.0        0.0         0
33 ! 0.0        0.000976563 1
34 " 0.00195313 0.00488281  3
35 # 0.00585938 0.0136719   8
…

48 0 0.0761719  0.0820313   6
49 1 0.0830078  0.0859375   3
50 2 0.0869141  0.0927734   6
…

65 A 0.185547   0.194336    9
66 B 0.195313   0.202148    7
67 C 0.203125   0.209961    7
…

97 a 0.421875   0.426758    5
98 b 0.427734   0.432617    5
99 c 0.433594   0.438477    5
…

125 } 0.573242  0.576172    3
126 ~ 0.577148  0.583984    7
```

3-5 Text with DirectWrite

- Surface sharing technique (*BraynzarSoft)
  - D3D 11, D3D 10.1, and D2D all use the DXGI 1.1 so we can use DXGI to create a render target which can be shared between the three API's
  1. Use D2D with a D3D 10.1 device to render to a surface
  2. Use a D3D 11 device to render that shared surface onto a square in screen space which overlays the entire scene



3-5 Text with DirectWrite

- D3D10.1, D2D, and DirectWrite setup
  - Use the same adapter as the D3D 11 device
  - Create a square and a shader resource view from the shared texture
    - ID3D11Texture2D: a shared texture between API's
  - Render a text onto the square

```
bool InitD2D_D3D101_DWrite(IDXGIAdapter1 *Adapter);
void InitD2DScreenTexture();
void RenderText(std::wstring text);
```

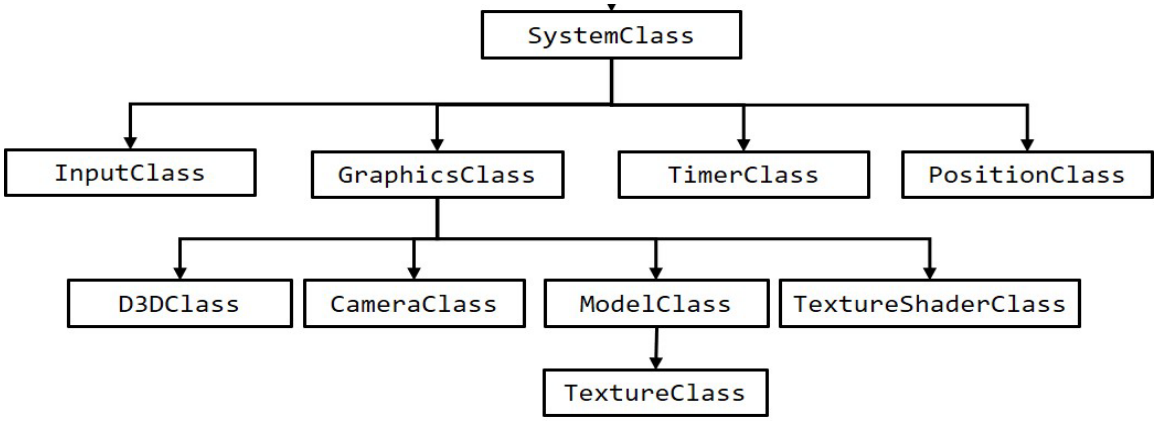
- Creating a font format using DirectWrite
  - IDWriteFactor::CreateTextFormat()
    - Set font weight, style, stretch, size, language

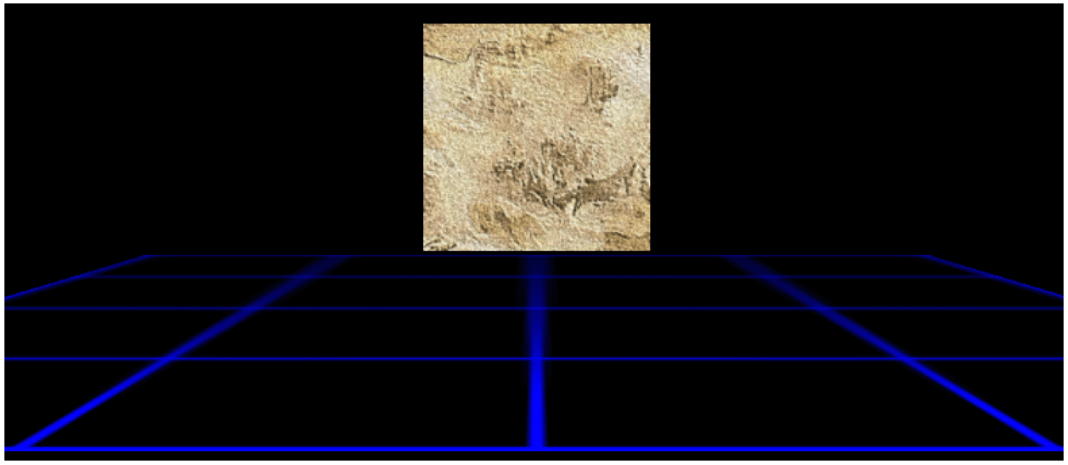
3-6 Billboarding

- Billboard
  - A technique in 3D graphics in which a sprite (i.e. a textured quad mesh) is rendered perpendicular to the camera without respect to camera movement
  - (+) Can reduce rendering computation

3-6 Billboarding

- Rotate (cylindrically) the billboard toward camera's position
  - PositionClass: Use left/right arrow keys to move with speed





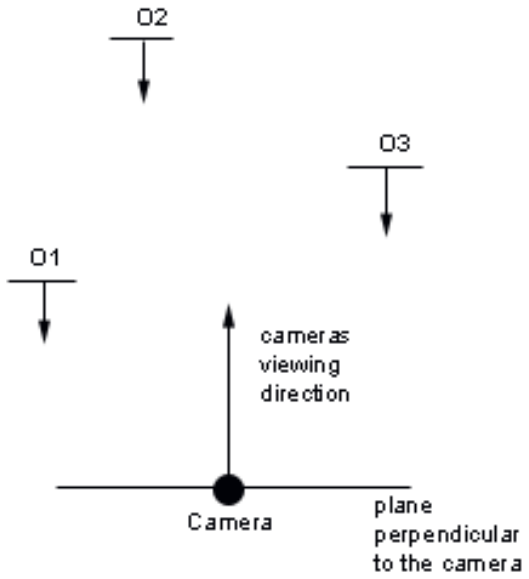
3-6 Billboarding

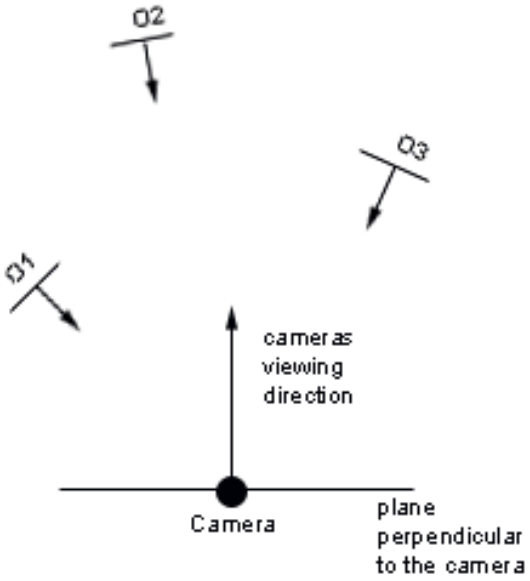
- Billboarding calculation
  - Get a position of the camera and set a position of billboard
  - Determine the rotation for the billboard so it faces the camera based on the camera's position
  - Create a world matrix for each billboard by combining the rotation and translation matrices
    - Spherical rotation:

$$M1 = \begin{bmatrix} a0 & a4 & a8 & a12 \\ a1 & a5 & a9 & a13 \\ a2 & a6 & a10 & a14 \\ a3 & a7 & a11 & a15 \end{bmatrix}$$

    - Cylindrical rotation:

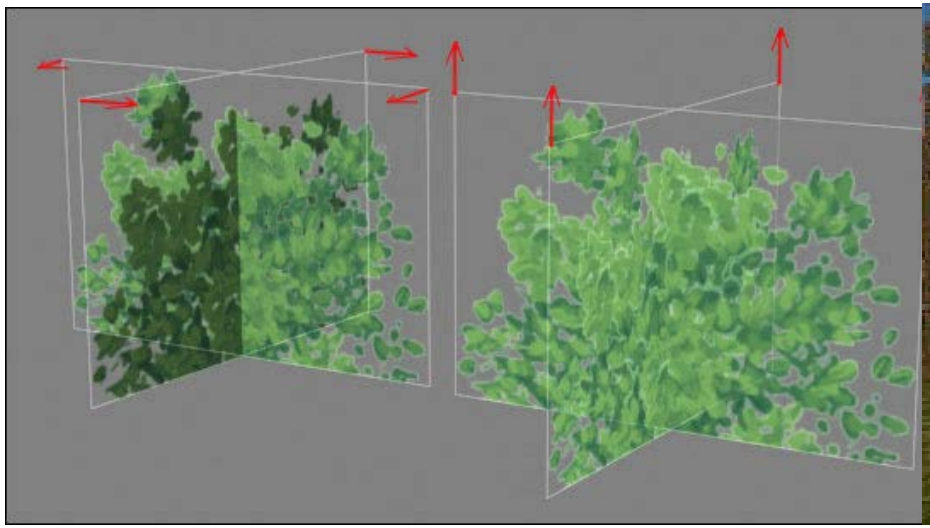
$$M1 = \begin{bmatrix} 1 & a4 & 0 & a12 \\ 0 & a5 & 0 & a13 \\ 0 & a6 & 1 & a14 \\ a3 & a7 & a11 & a15 \end{bmatrix}$$





3-6 Billboarding

- Cross-billboards with multiple images



References

- Wikipedia
  - www.wikipedia.org
- Introduction to DirectX 11
  - www.3dgep.com/introduction-to-directx-11
- Raster Tek
  - www.ratertek.com
- Braynzar Soft
  - www.braynzarsoft.net)
- CS 445: Introduction to Computer Graphics *[Aaron Bloomield]*
  - www.cs.virginia.edu/~asb/teaching/cs445-fall06

Q & A