Texture Mapping, Blending

김준호

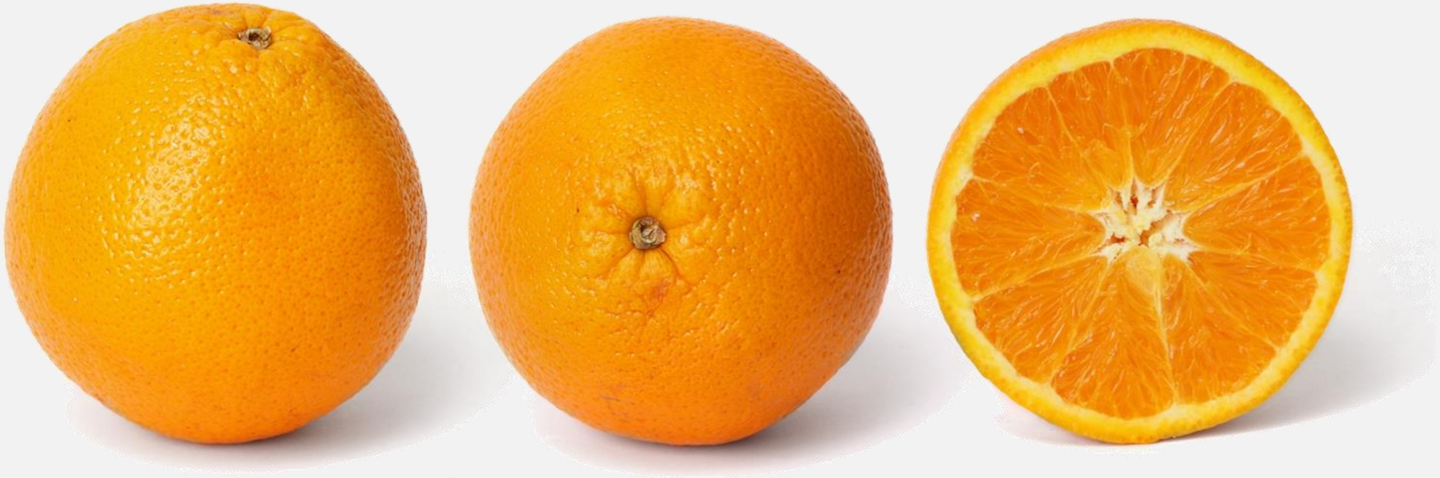
Visual Computing Lab.

국민대학교 소프트웨어학부

Texture Mapping Basics

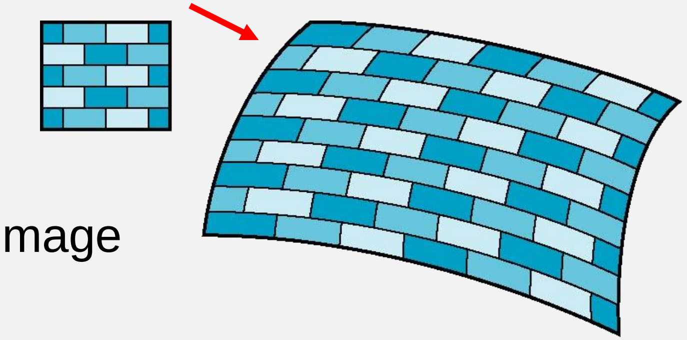
Modeling Object w/ Details

- How can you represent the details of an object?



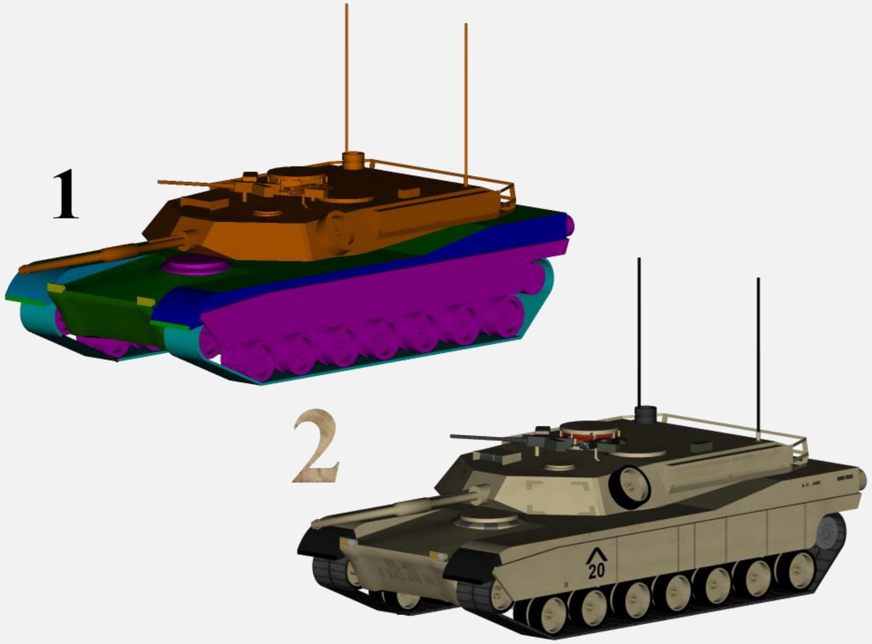
Basic Idea of Texture Mapping

- Uses images to fill inside of polygons



2D image

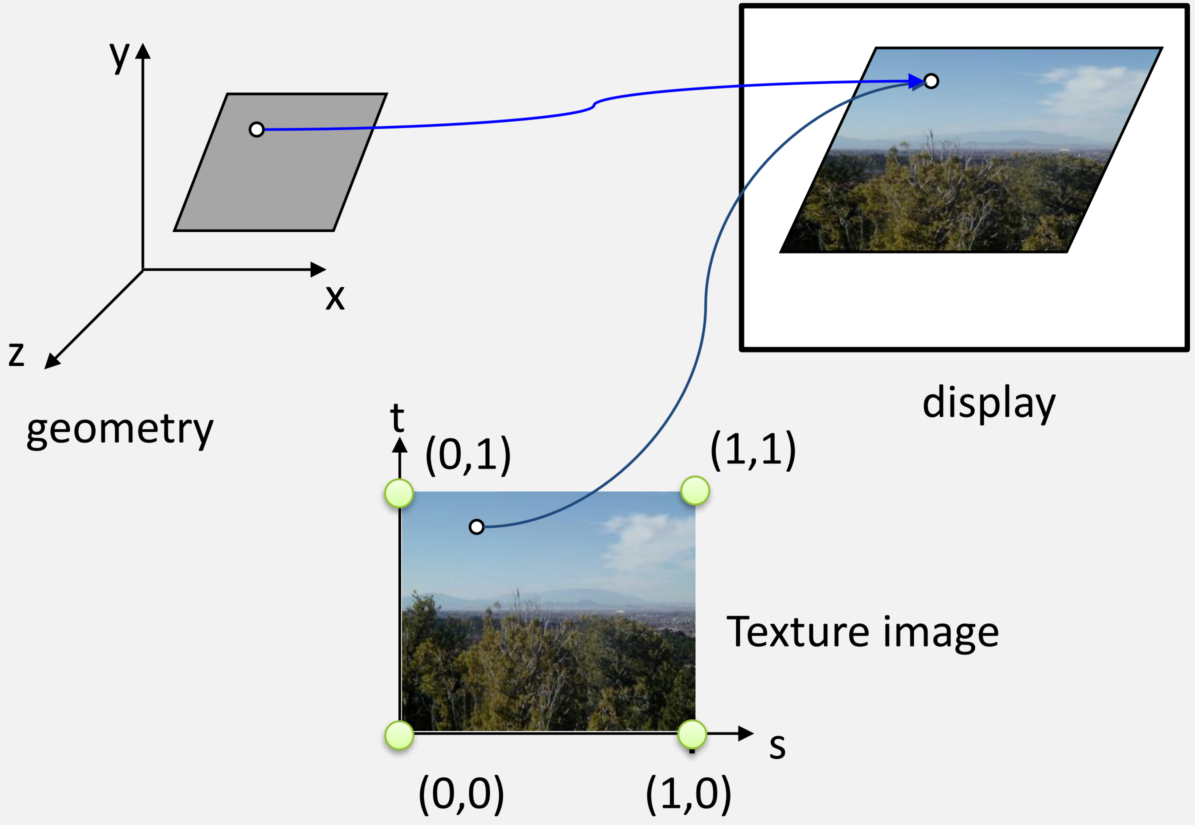
3D surface



Texture

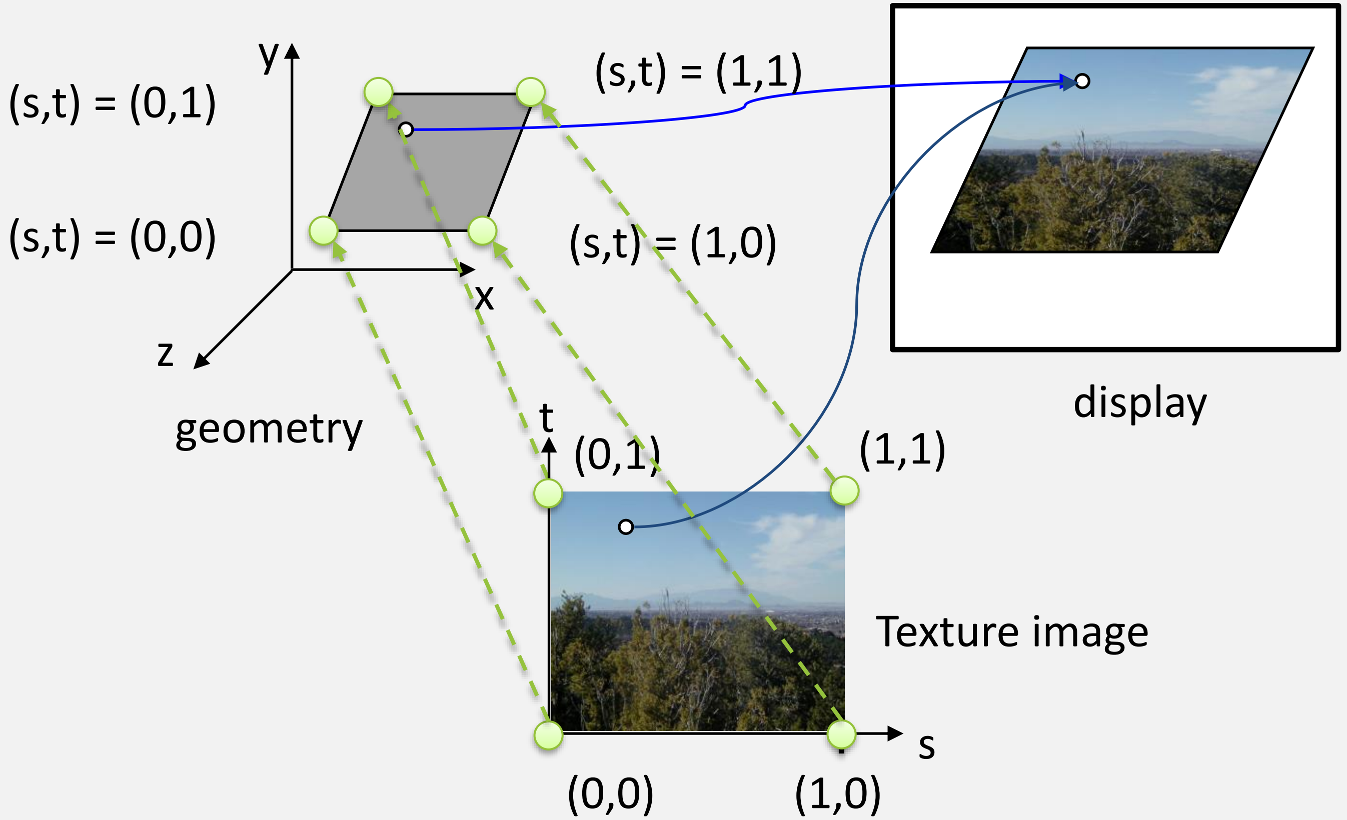
- Definition from Oxford dictionaries
  - The character of appearnce of a textile fabric as determined by the arrangement and thickness of its threads
- Definition used in computer graphics
  - Large chuncks of ***image data*** that can be used to paint the surfaces of objects

Texture Mapping

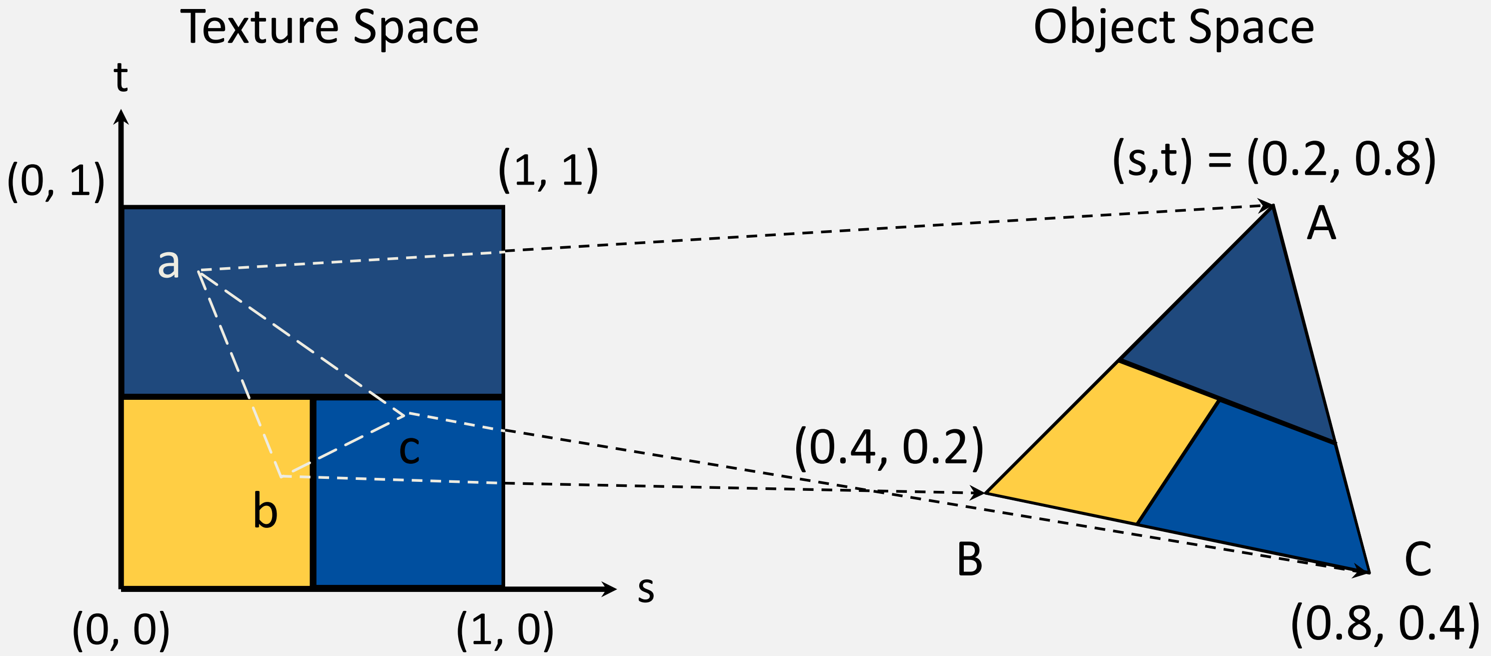


Texture Mapping

- We need to specify per-vertex texture coordinates

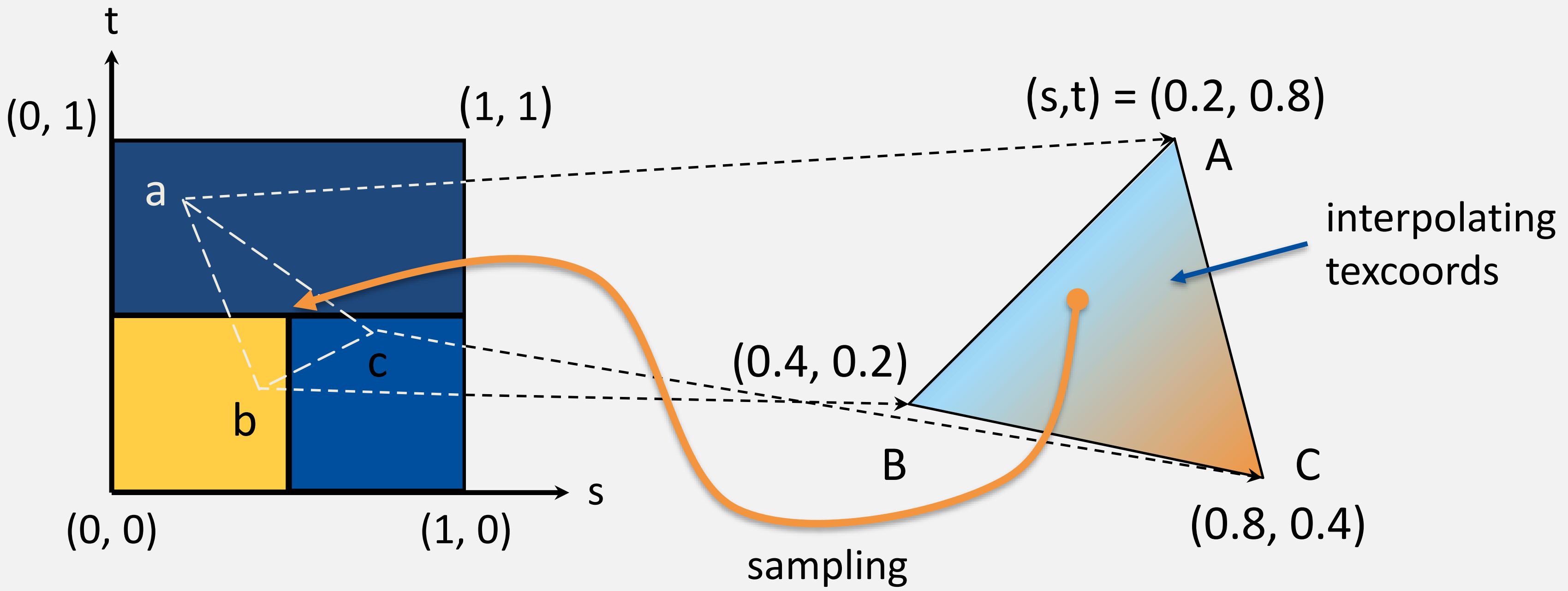


Texture Coordinates Matter



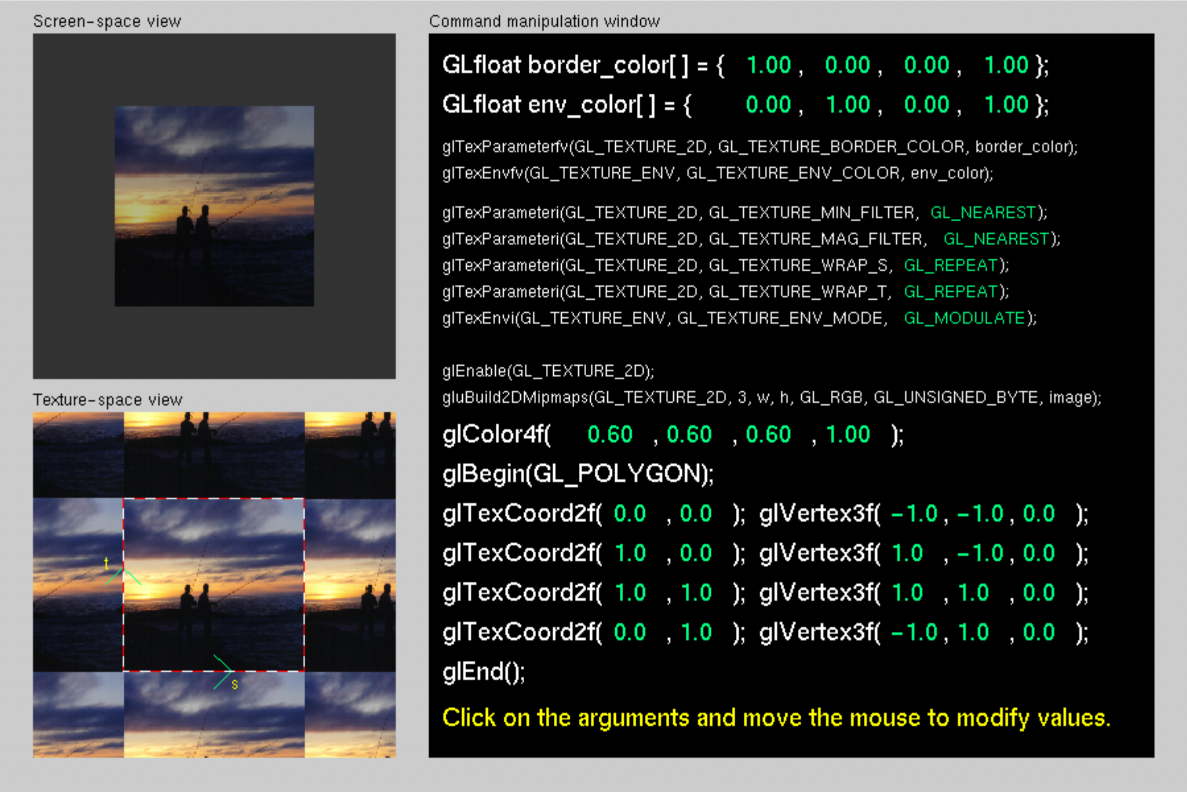
How can OpenGL Patch Colors from Texture?

- Bilinear interpolation on texture coordinates
- Sample colors from a texture image
  - Sampling problem!



Demo – Texture Mapping

- Tutorial from Nate Robins
  - http://user.xmission.com/~nate/tutors.html



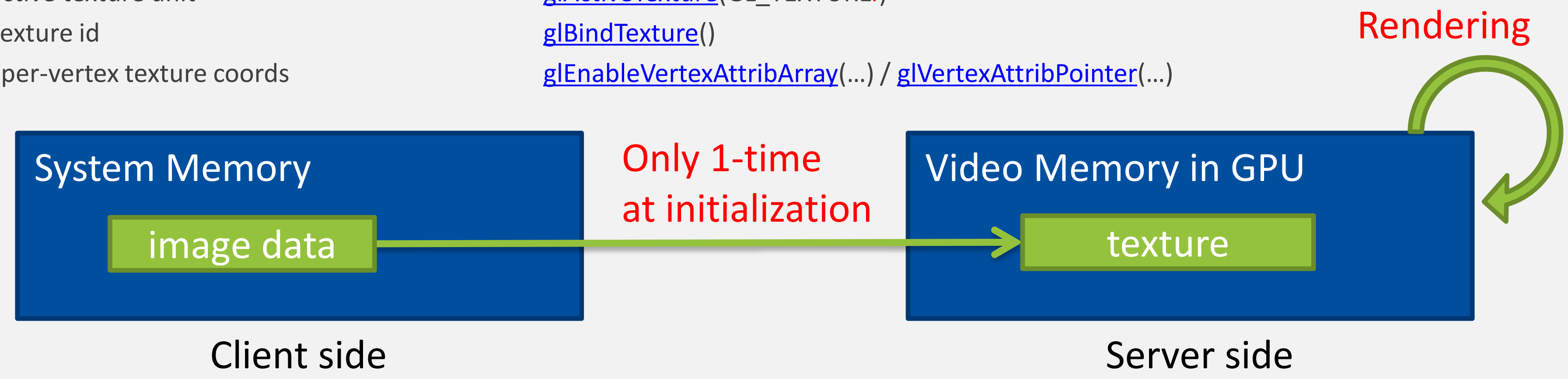
```
GLfloat border_color[ ] = {  1.00 ,  0.00 ,  0.00 ,  1.00 };
GLfloat env_color[ ] = {     0.00 ,  1.00 ,  0.00 ,  1.00 };

glTexParameterfv(GL_TEXTURE_2D, GL_TEXTURE_BORDER_COLOR, border_color);
glTexEnvfv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, env_color);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,  GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,  GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,  GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,  GL_REPEAT);
glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,  GL_MODULATE);

glEnable(GL_TEXTURE_2D);
gluBuild2DMipmaps(GL_TEXTURE_2D, 3, w, h, GL_RGB, GL_UNSIGNED_BYTE, image);
glColor4f(   0.60  , 0.60  , 0.60  , 1.00  );
glBegin(GL_POLYGON);
glTexCoord2f( 0.0  , 0.0  );  glVertex3f( -1.0 , -1.0 , 0.0  );
glTexCoord2f( 1.0  , 0.0  );  glVertex3f( 1.0  , -1.0 , 0.0  );
glTexCoord2f( 1.0  , 1.0  );  glVertex3f( 1.0  , 1.0  , 0.0  );
glTexCoord2f( 0.0  , 1.0  );  glVertex3f( -1.0 , 1.0  , 0.0  );
glEnd();
```

Click on the arguments and move the mouse to modify values.

Using Texture Mapping in Modern OpenGL

- Steps to use texture mapping

1. Generate texture identifiers	glGenTextures()
2. Binding a texture id	glBindTexture()
3. Specify texture data	
• Load image from a file (or generate image)	
• Select active texture unit	glActiveTexture(GL_TEXTURE*i*)
• Specify texture parameters	glTexParameter()
– Wrapping mode, Filtering methods	
• Specify texture data	glTexImage2D()
• Specify texture sampler in shader	glUniform1i(glGetUniformLocation(…), *i*)
4. Rendering with texture mapping	
• Select active texture unit	glActiveTexture(GL_TEXTURE*i*)
• Bind a texture id	glBindTexture()
• Specify per-vertex texture coords	glEnableVertexAttribArray(…) / glVertexAttribPointer(…)



Modern OpenGL – Texture Mapping

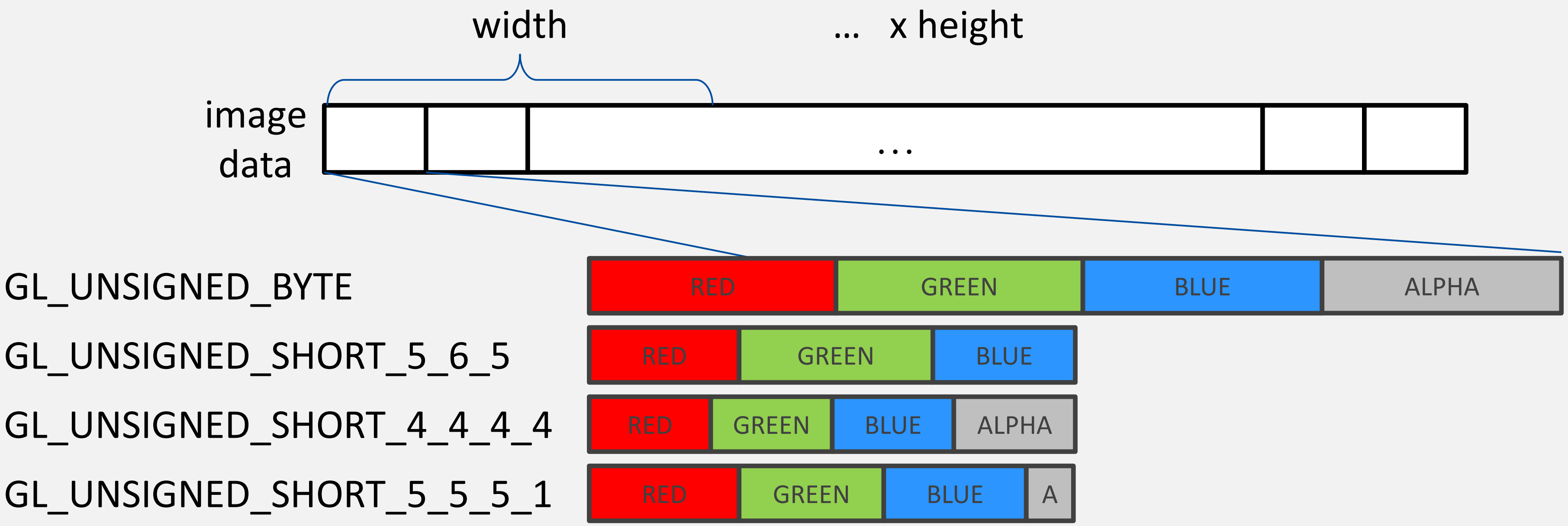
- Initialization
  1. Generate texture ids
  2. Binding a texture id
  3. Specify texture data
     - Load image from a file (or generate image)
     - Select active texture unit
     - Wrapping mode, Filtering methods
     - Specify texture data
     - Specify texture sampler in shader
  4. Enable texture mapping
  5. Binding a texture id
  6. Rendering w/ texcoords

- Modern OpenGL codes (C/C++)

```
// variables for texture mapping
GLuint          tex_id;
GLsizei         width, height;
GLbyte          *img_pixels;


// load an image from system
// img_pixels must locate the client-side memory of the image
// width/height should be update
// pixel format is important
// …

// Generate a texture
glGenTextures(1, &tex_id);
// Bind a texture w/ the following OpenGL texture functions
glBindTexture(GL_TEXTURE_2D, tex_id);
// Select active texture unit 0
glActiveTexture(GL_TEXTURE0);

// Set texture parameters (wrapping modes, sampling methods)
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

// Transfer an image data in the client side to the server side
glTexImage2D(tex_id, 0, GL_RGBA, width, height, 0,
             GL_RGBA, GL_UNSIGNED_BYTE, pixels);
// Specify texture sampler in shader
glUniform1i(glGetUniformLocation(program, "my_sampler"), 0);
```

Specifying a Texture Image

- Load an image from a file
  - There is no OpenGL function about it
    - You should use a platform-specific way
  - The image data should be stored in bitmap-like data structure
    - Data must be admitted by glTexImage2D()



Modern OpenGL codes – Texture Mapping

- Rendering
  1. Generate texture ids
  2. Binding a texture id
  3. Specify texture data
     - Load image from a file (or generate image)
     - Select active texture unit
     - Wrapping mode, Filtering methods
     - Specify texture data
     - Specify texture sampler in shader
  4. Select active texture unit
  5. Binding a texture id
  6. Specify per-vertex texture coords

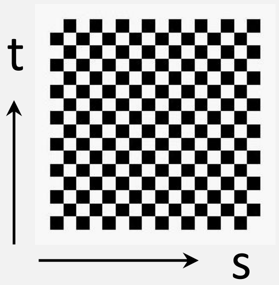
- Modern OpenGL codes (C/C++)

```
// Select active texture unit
glActiveTexture(GL_TEXTURE0);

// Bind a texture w/ the following OpenGL texture functions
glBindTexture(GL_TEXTURE_2D, tex_id);


// Rendering w/ texcoords
glBindBuffer(…)
glEnableVertexAttribArray(loc_a_texcoord)

glVertexAttribPointer(loc_a_texcoord, …)

glDrawArrays(…);

glDisableClientState(loc_a_texcoord);
```

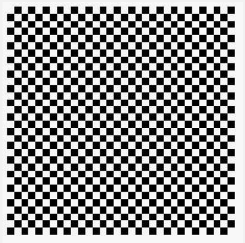
Texture Address Mode

Texture Address Mode

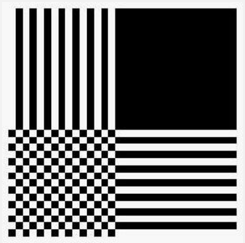
- How to repeat a given texture patterns when each texcoords *s* or *t* is not in [0,1]
- Why do we need it?



texture



GL_REPEAT
wrapping



GL_CLAMP_TO_EDGE
clamping

Texture Address Mode in Modern OpenGL

- Steps to use texture mapping
  1. Generate texture identifiers
  2. Binding a texture id
  3. Specify texture data
     - Load image from a file (or generate image)
     - Select active texture unit
     - **Specify texture parameters**
       - **Wrapping mode**, Filtering methods
     - Specify texture data
     - Specify texture sampler in shader
  4. Rendering with texture mapping
     - Select active texture unit
     - Bind a texture id
     - Specify per-vertex texture coords

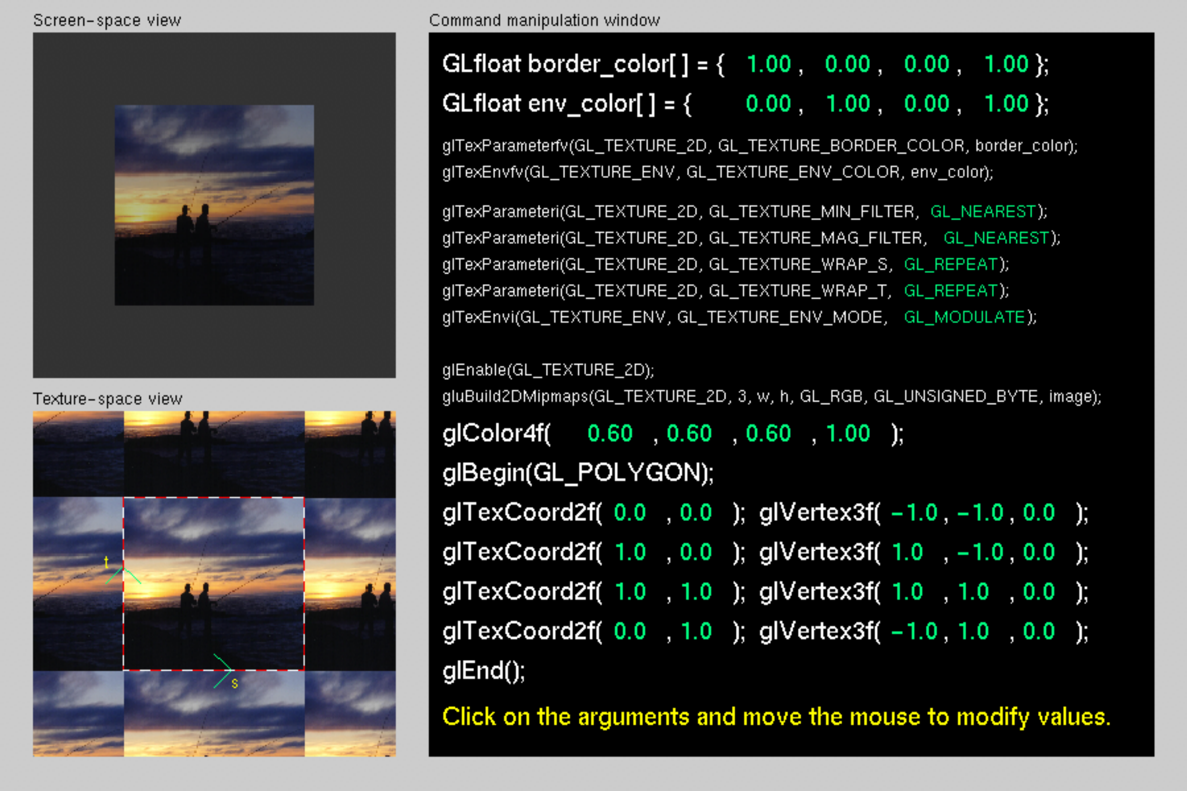
Texture Address Mode in Modern OpenGL

- Steps to use
  1. Generate te
  2. Binding a te
  3. Specify text
     - Load ima
     - Select active texture unit
     - **Specify texture parameters**
       - **Wrapping mode**, Filtering methods
     - Specify texture data
     - Specify texture sampler in shader
  4. Rendering with texture mapping
     - Select active texture unit
     - Bind a texture id
     - Specify per-vertex texture coords

```
// texture address mode as repeat
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);


// texture address mode as clamp
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
```

Demo – Texture Mapping

- Tutorial from Nate Robins
  - http://user.xmission.com/~nate/tutors.html



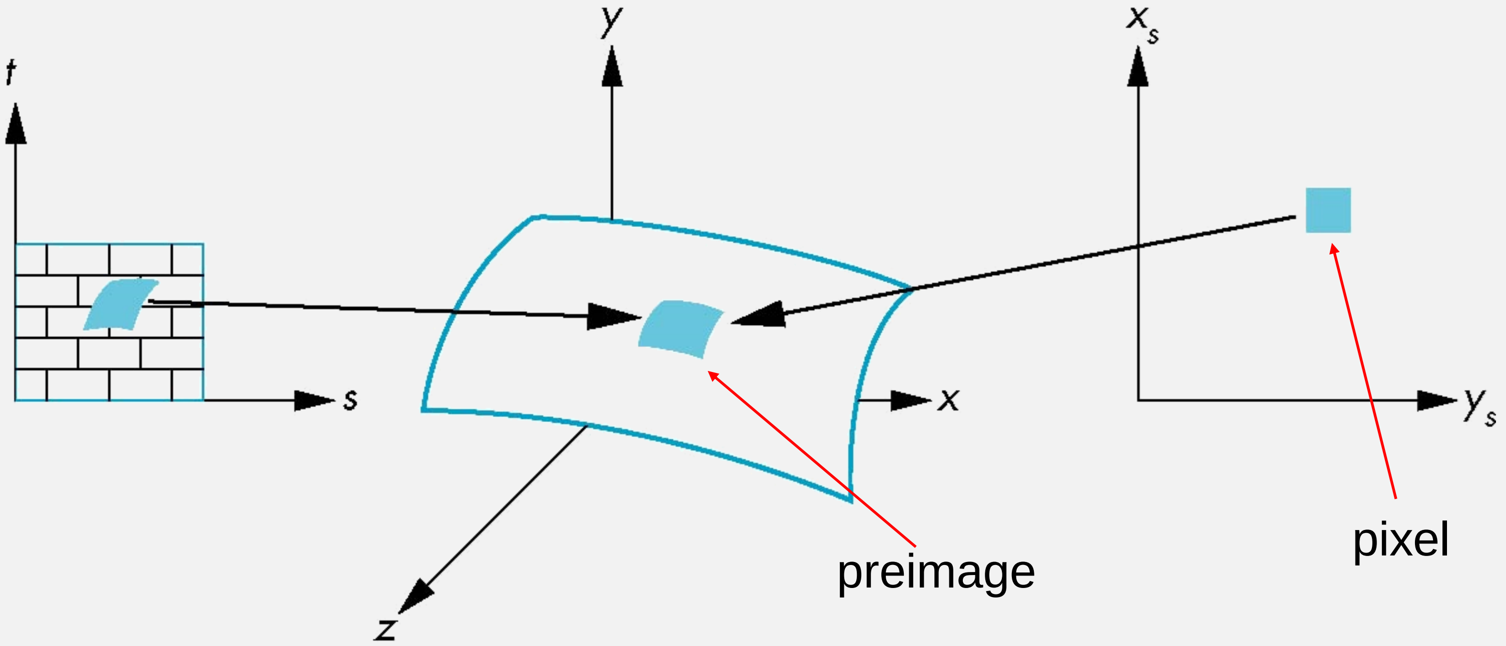
```
GLfloat border_color[ ] = {   1.00 ,   0.00 ,   0.00 ,   1.00 };
GLfloat env_color[ ] = {       0.00 ,   1.00 ,   0.00 ,   1.00 };
glTexParameterfv(GL_TEXTURE_2D, GL_TEXTURE_BORDER_COLOR, border_color);
glTexEnvfv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, env_color);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,  GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,  GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,  GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,  GL_REPEAT);
glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,  GL_MODULATE);

glEnable(GL_TEXTURE_2D);
gluBuild2DMipmaps(GL_TEXTURE_2D, 3, w, h, GL_RGB, GL_UNSIGNED_BYTE, image);
glColor4f(    0.60   , 0.60   , 0.60   , 1.00   );
glBegin(GL_POLYGON);
glTexCoord2f( 0.0   , 0.0   );  glVertex3f( -1.0 , -1.0 , 0.0   );
glTexCoord2f( 1.0   , 0.0   );  glVertex3f( 1.0   , -1.0 , 0.0   );
glTexCoord2f( 1.0   , 1.0   );  glVertex3f( 1.0   , 1.0   , 0.0   );
glTexCoord2f( 0.0   , 1.0   );  glVertex3f( -1.0 , 1.0   , 0.0   );
glEnd();
```

Click on the arguments and move the mouse to modify values.

Sampling Problem in Texture Mapping

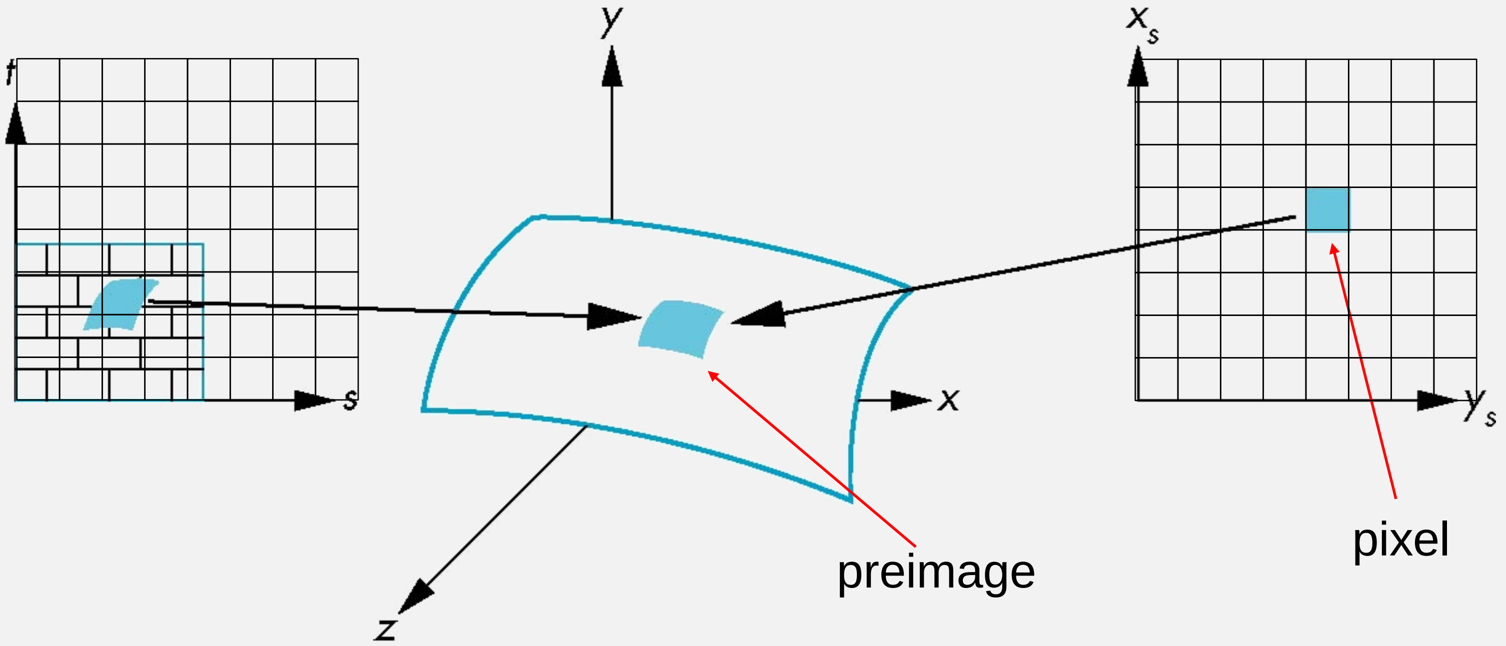
Sampling Problem

- A pixel must have one color value!!!

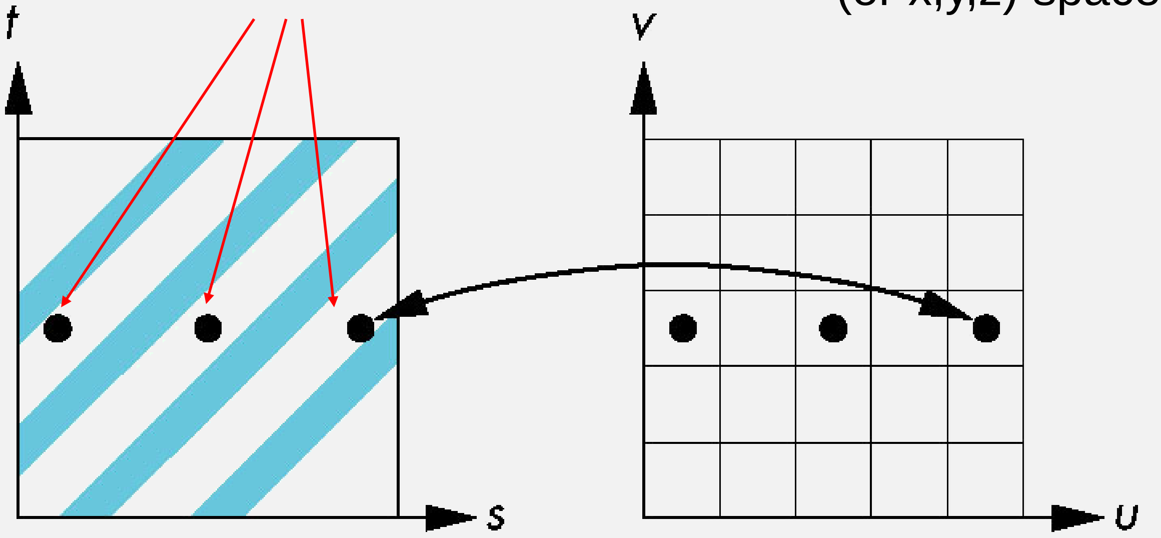


Sampling Problem

- A pixel must have one color value!!!
  - A pixel may correspond to several texels
  - A pixel may correspond to a small portion of a texel



Aliasing



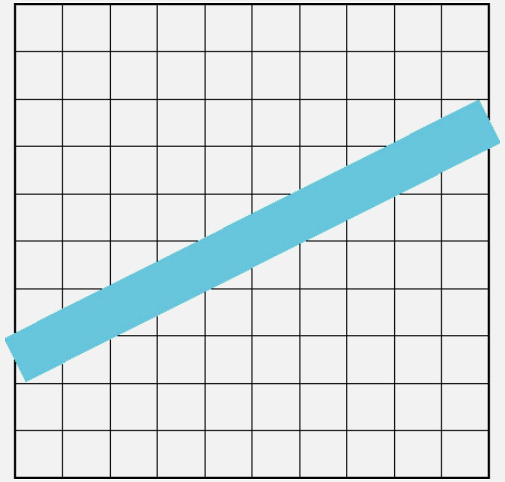
miss blue stripes

point samples in u,v
(or x,y,z) space

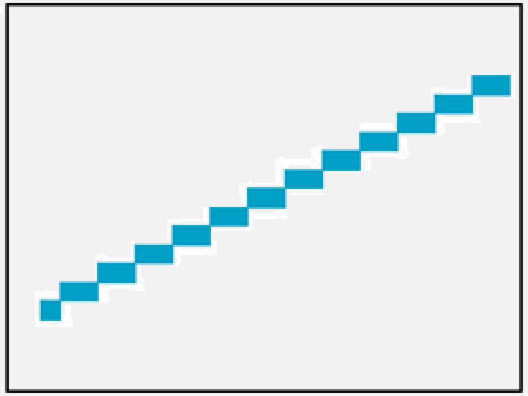
point samples in texture space

Aliasing

- What is aliasing?
  - Artifact from the limited sampling rates
  - What are antialiasing examples in the real-world? (video)



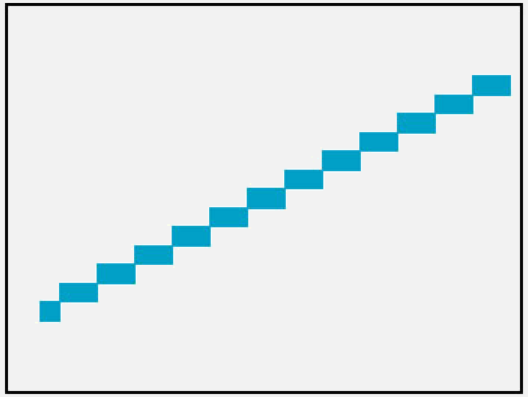
Ideal line



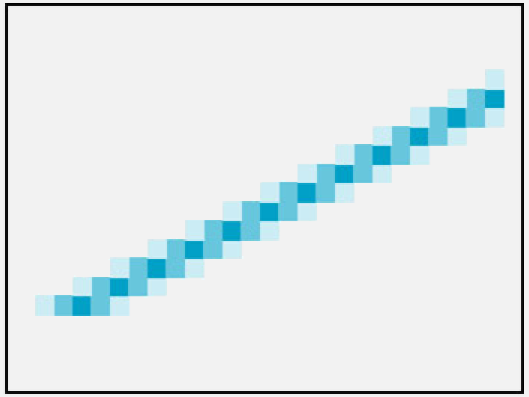
Rasterized line

Antialiasing

- Then, what is **anti**aliasing?
  - Ideally, it implies to cut-off the high-frequency terms.
  - In practice, it implies blurring



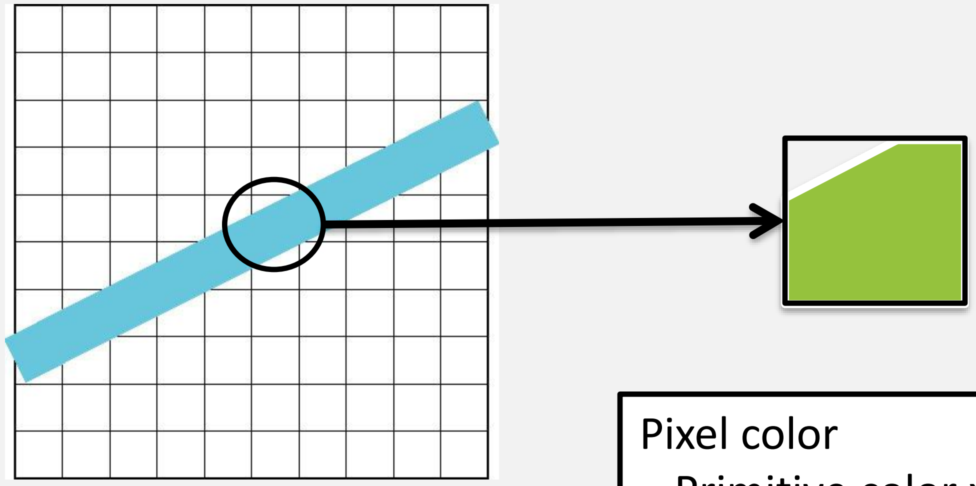
original



antialising

Antialising

- Basic idea?
  - Consider the contribution of the primitive about each pixel

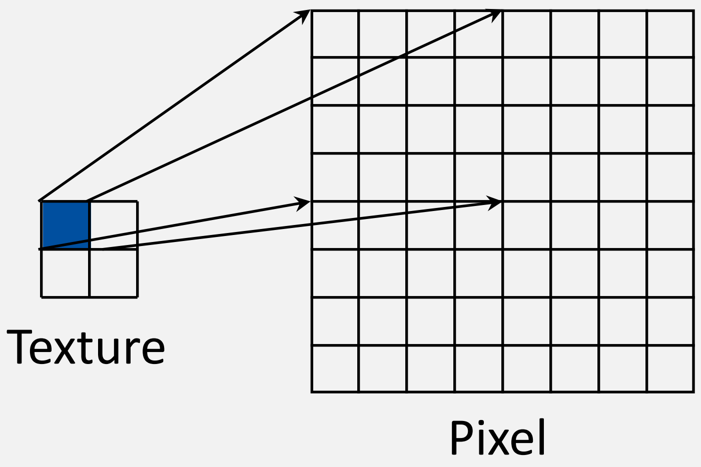


Pixel color
= Primitive color x (occupancy rate)

Magnification/Minification

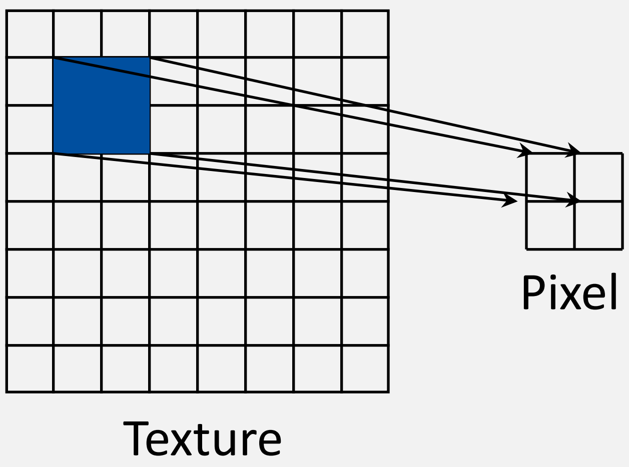
Magnification

- A texel is larger than 1 pixel
  - In general, zoom-in case



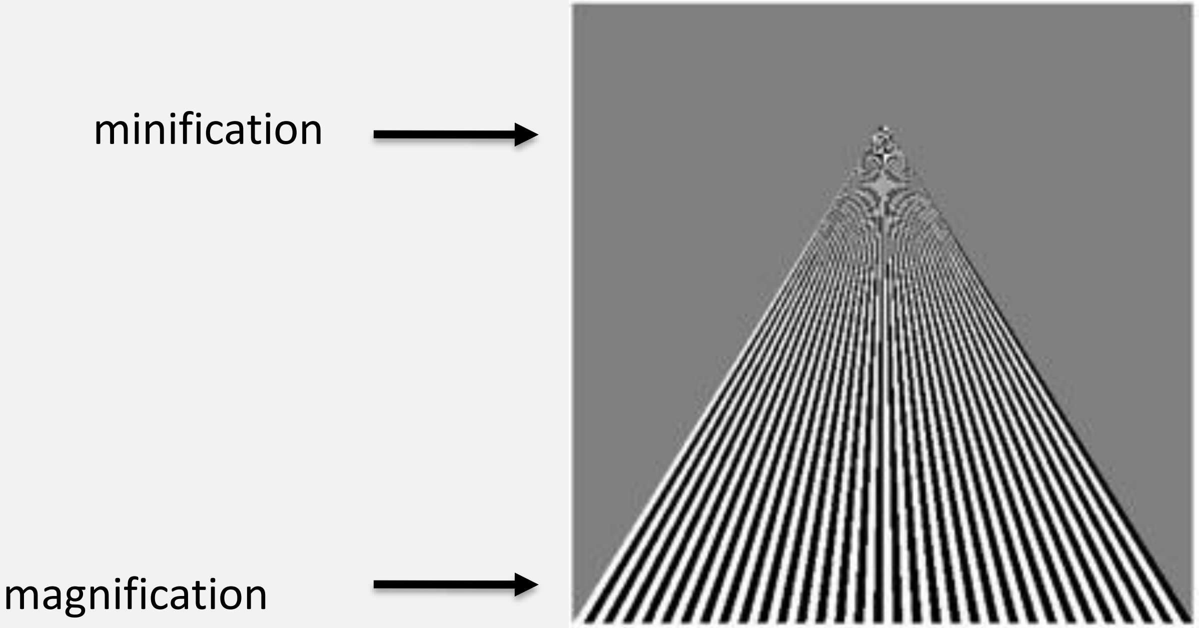
Minification

- A texel is smaller than 1 pixel
  - In general, zoom-out case



Example

- Aliasing happens both of the minification part and the magnification part

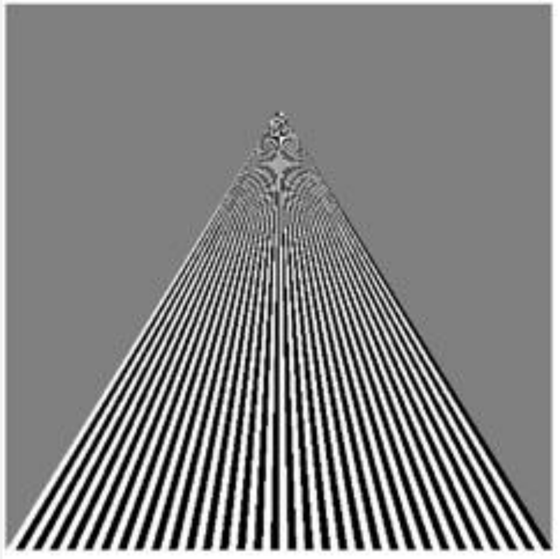


minification

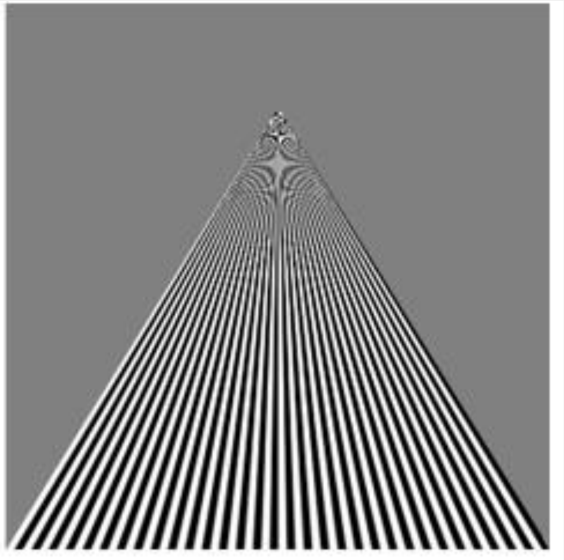
magnification

Texture Filtering

- Sampling patterns in textures
  - Nearest sampling (default)
  - Linear sampling (better quality)



Nearest sampling



Linear sampling

Texture Filtering in Modern OpenGL

- Steps to use texture mapping
  1. Generate texture identifiers
  2. Binding a texture id
  3. Specify texture data
     - Load image from a file (or generate image)
     - Select active texture unit
     - **Specify texture parameters**
       - **Wrapping mode**, **Filtering methods** glTexParameter()
     - Specify texture data
     - Specify texture sampler in shader
  4. Rendering with texture mapping
     - Select active texture unit
     - Bind a texture id
     - Specify per-vertex texture coords

Texture Filtering in Modern OpenGL

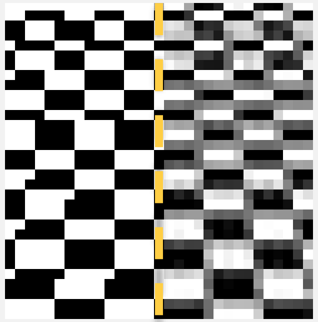
```
// setting for nearest samplings
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);


// setting for linear samplings
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```

- Steps to use
  1. Generate te
  2. Binding a te
  3. Specify text
     - Load ima
     - Select active texture unit
     - **Specify texture parameters**
       - **Wrapping mode**, **Filtering methods** glTexParameter()
     - Specify texture data
     - Specify texture sampler in shader
  4. Rendering with texture mapping
     - Select active texture unit
     - Bind a texture id
     - Specify per-vertex texture coords

Advanced Topics of Texture Mapping

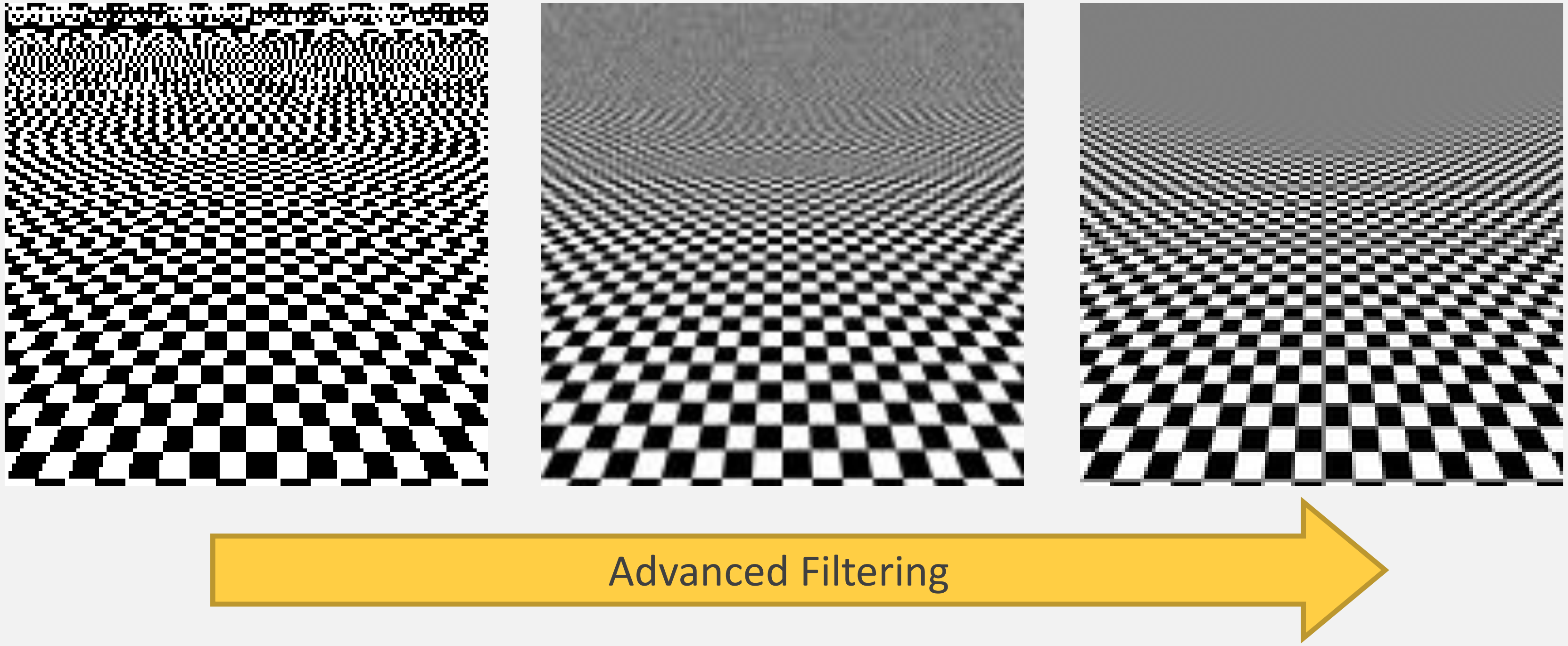
Advanced texture Sampling

- Aliasing of textures
  - Antialiasing is a kind of using blur filters



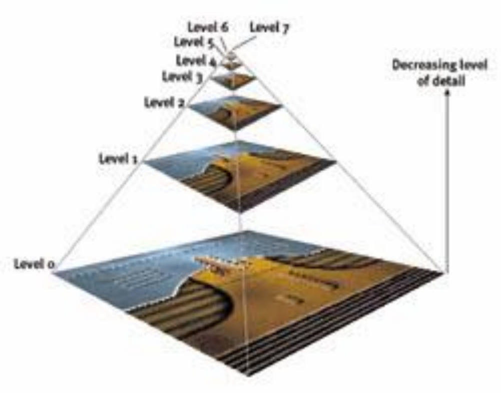
Nearest-point Filtering

Linear Filtering

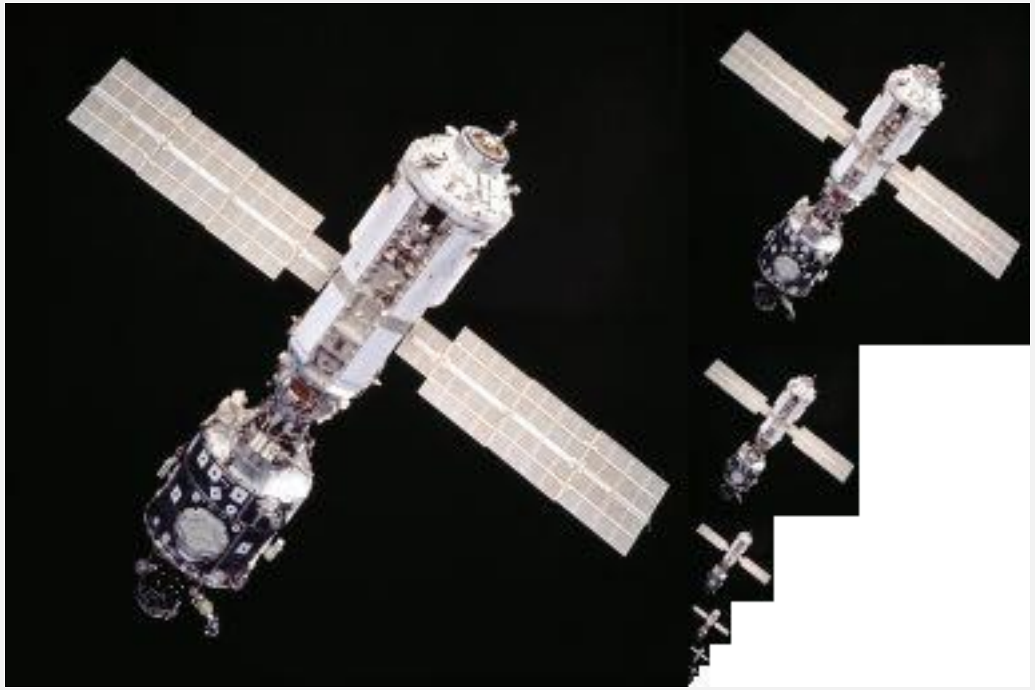


Mipmap

- Efficient handling minification problem
- Building an image pyramid

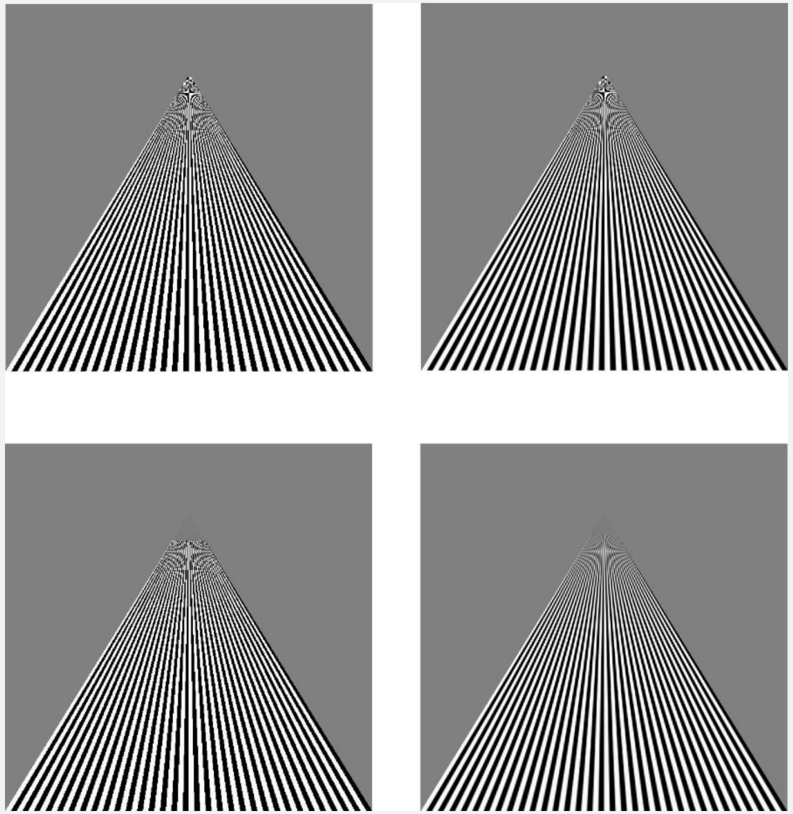


Concept of
image pyramid



Mipmap

- With mipmap, OpenGL use pre-filtered image (i.e., mipmap) for texture sampling



Point sampling

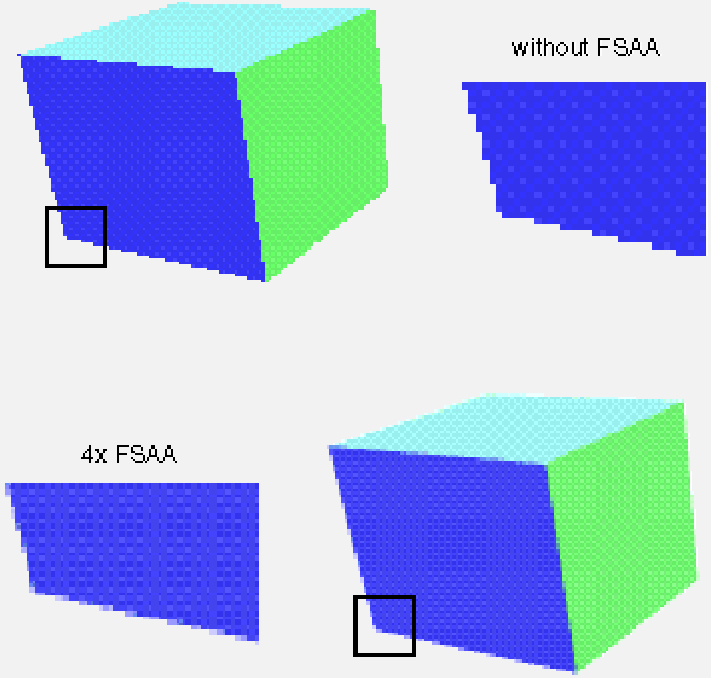
Linear filtering

Mipmapped
Point sampling

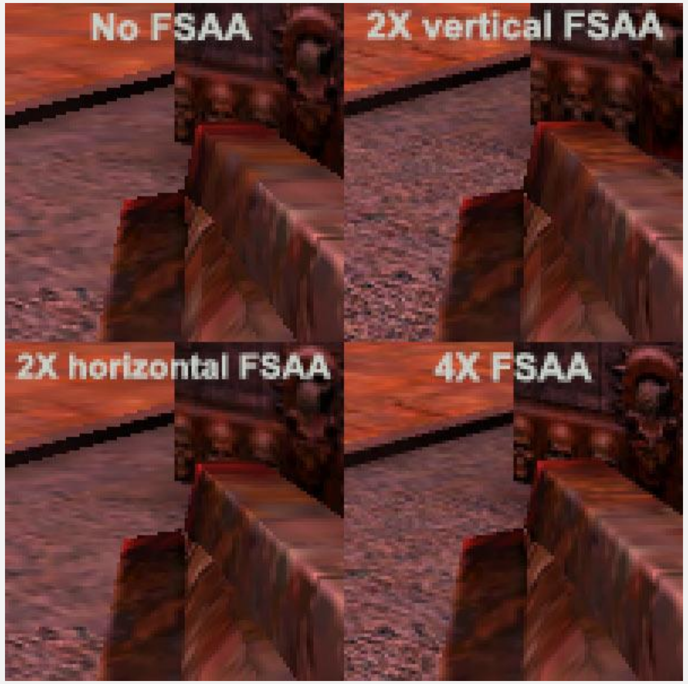
Mipmapped
Linear filtering

Full-Scene Anti-Aliasing (FSAA) or Multisample Anti-Aliasing (MSAA)

- Super sampling anti-aliasing for avoiding aliasing (or jaggies) on full-screen images



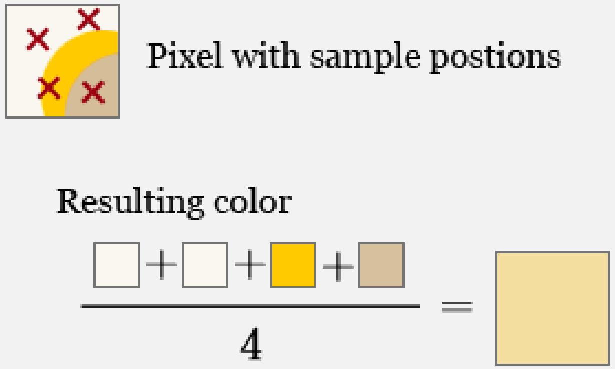
http://techpubs.sgi.com/library/dynaweb_docs/0650/SGI_EndUser/books/MPK_UG/sgi_html/ch04.html]

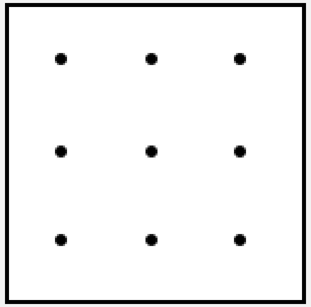


http://www.dansdata.com/prophet4500.htm

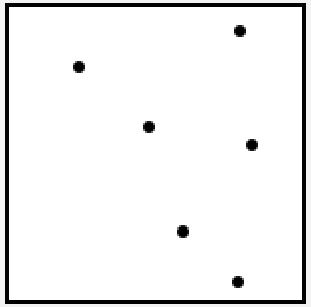
Full-Scene Anti-Aliasing (FSAA) or Multisample Anti-Aliasing (MSAA)

- Super sampling anti-aliasing for avoiding aliasing (or jaggies) on full-screen images

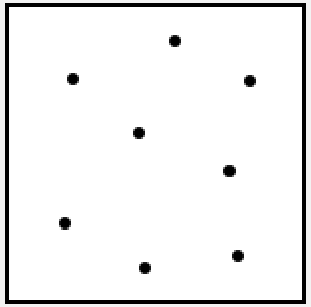




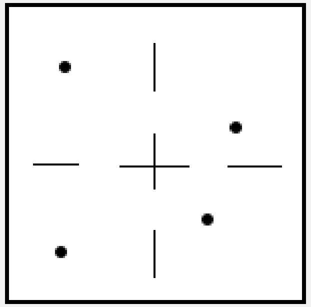
Grid



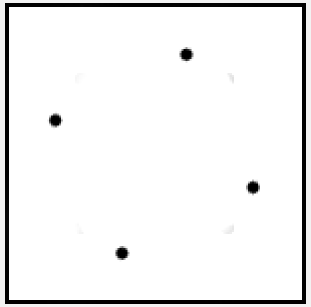
Random



Poisson



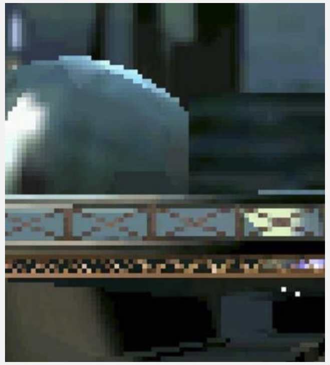
Jitter



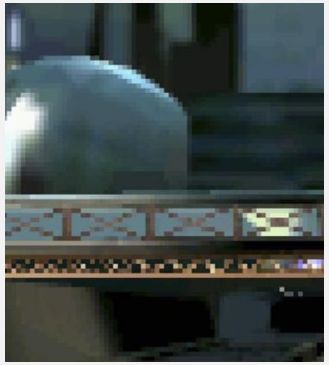
Rotated grid

Full-Scene Anti-Aliasing (FSAA) or Multisample Anti-Aliasing (MSAA)

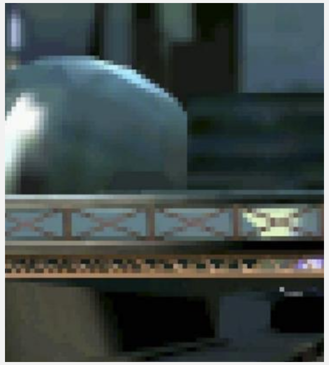
- FSAA invokes the issues about performance and power efficiency
  - GPU bendors (e.g., ARM Mali-T6XX series) argue that
    - 4x FSAA
      - Performance is matined at 90%+
      - Power-efficient anti-aliasing affectively comes ‘for free’ in the hardware
    - 16x FSAA
      - There is an up to 4x increase in per-pixel rendering cost
      - This is a cost-effective path to very high image quality



No FSAA



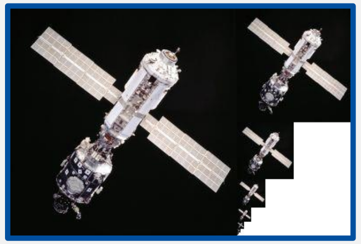
4x FSAA



16x FSAA

[images from ARM GPUs slides]

Using Mipmap in Modern OpenGL



OpenGL < 3.0

- Mipmap generation is NOT supported in the API level

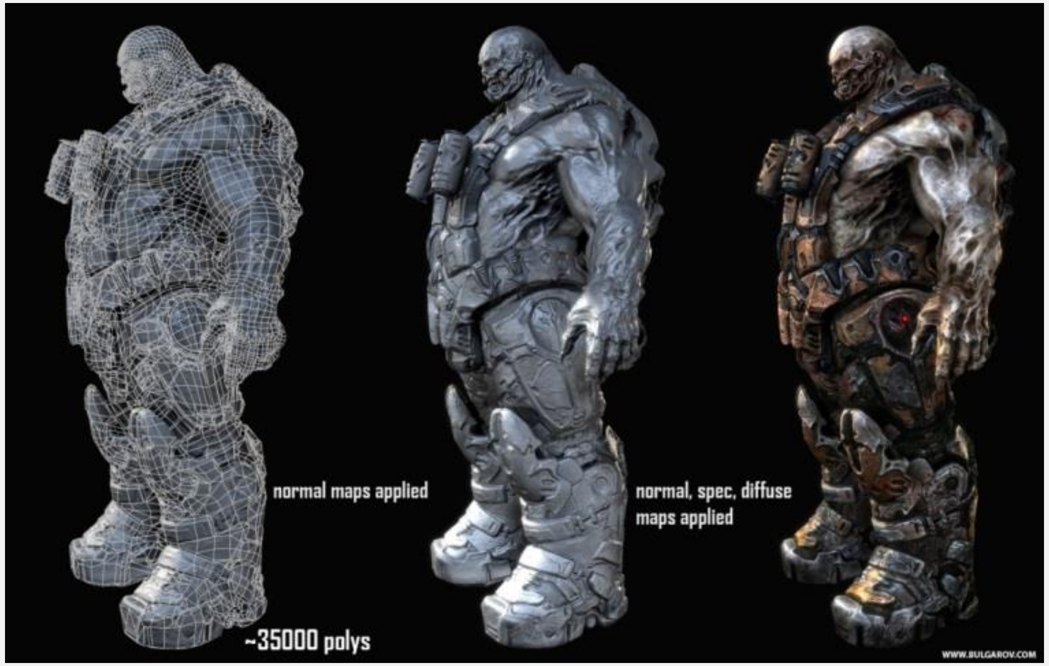
OpenGL 3.0+

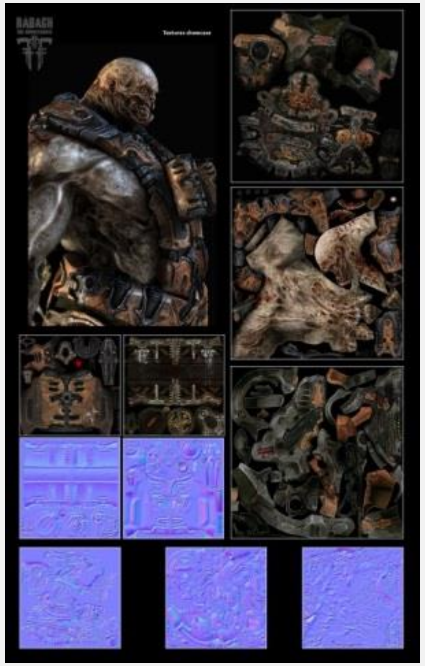
- Mipmap generation is supported in the API level
  - glGenerateMipmap()

```
// Bind texture
glBindTexture(GL_TEXTURE_2D, tex_id);

// Setting several texture parameters
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_NEAREST);

// Specify texture image data
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, texture_width, texture_height, 0, GL_RGBA, GL_UNSIGNED_BYTE, texture_data);

glGenerateMipmap(GL_TEXTURE_2D); // Generate a mipmap texture, Unavailable in OpenGL 2.1
```

Mipmap – Texture Generation is important

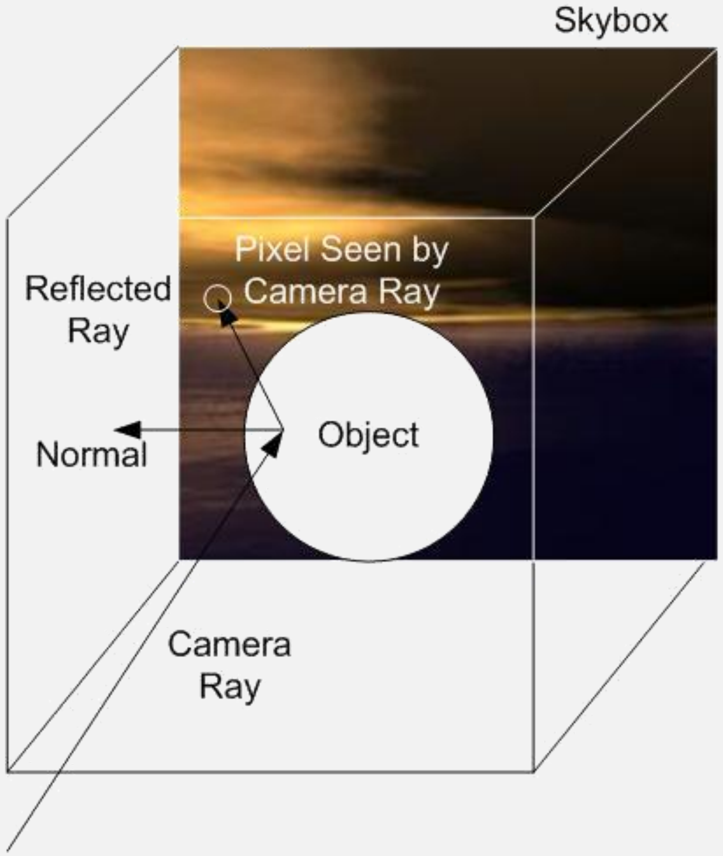
- When using mipmap, we may encouter artifacts
  - Mipmap generation may blend texture colors in the other parts
  - Mipmap sampling may sample the ill-blended texture colors





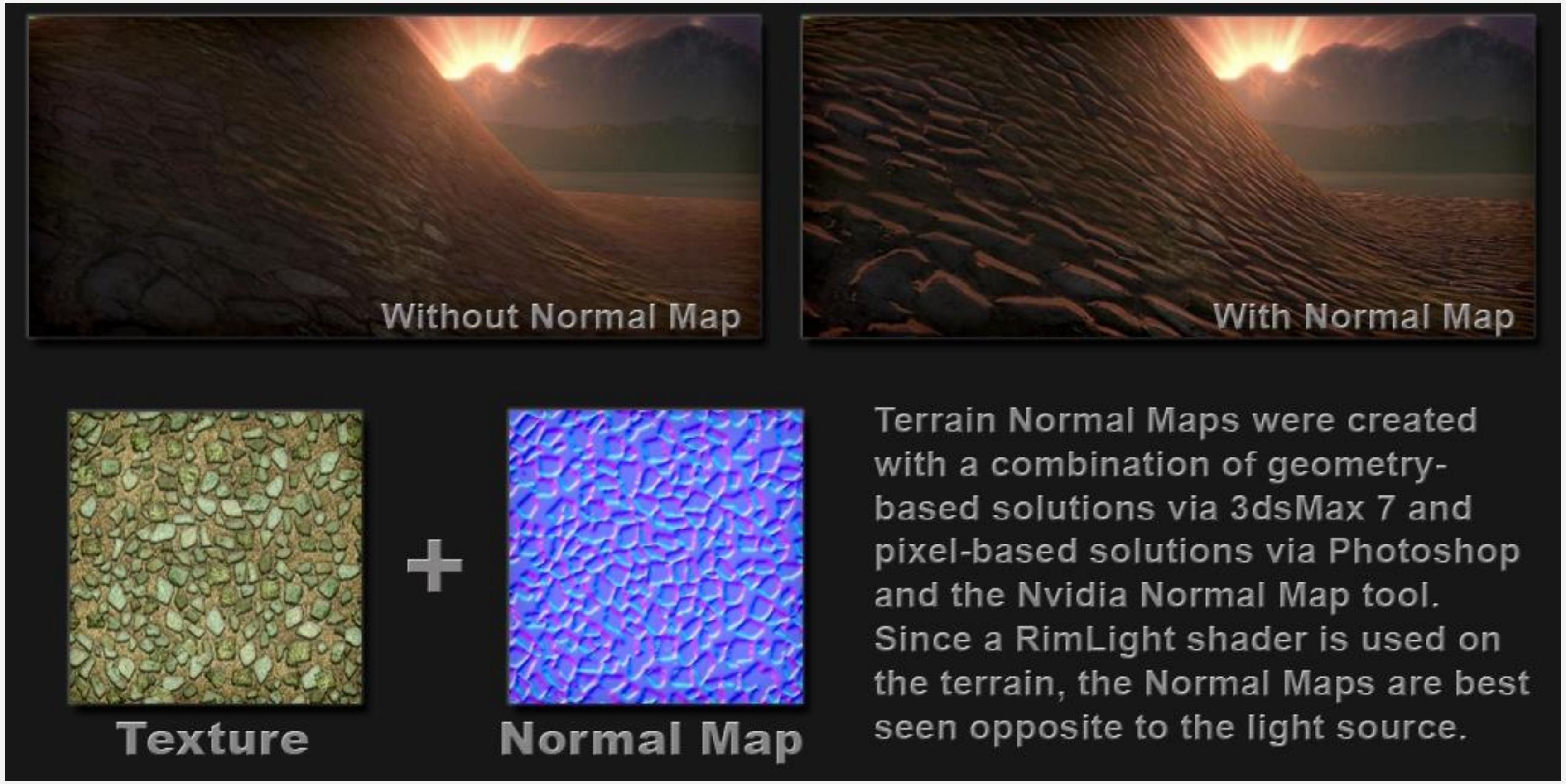
Environment Mapping

[Forza Motorsport 4]



Normal Mapping

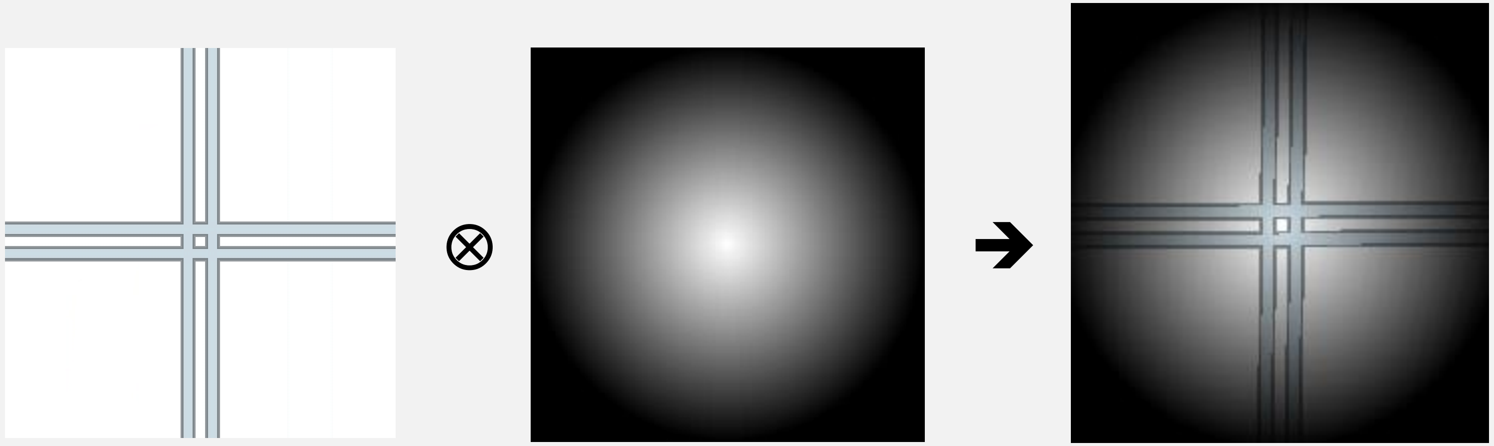
- Available w/ Pixel Shaders



Terrain Normal Maps were created with a combination of geometry-based solutions via 3dsMax 7 and pixel-based solutions via Photoshop and the Nvidia Normal Map tool. Since a RimLight shader is used on the terrain, the Normal Maps are best seen opposite to the light source.

Multi-Texturing

- Texture blending
  - To generate a new texture image by blending given two or more textures



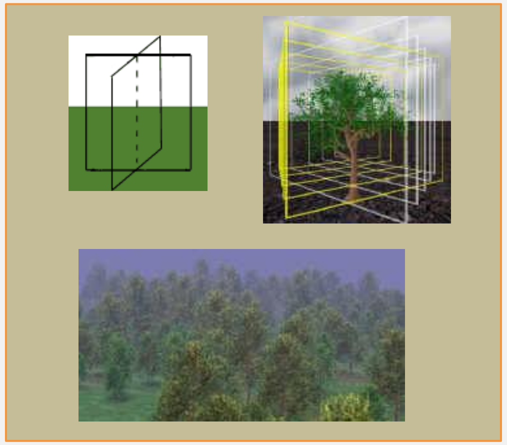
Multi-Texturing

- Quake 2
  - http://www.bigpanda.com/trinity/article1.html

Blending

Alpha Blending

- Billboard
  - To represent a 3D object with a textured 2D plane
  - Texture images must have alpha channels
- Alpha value represents opacity
  - 1: opque
  - 0: transparent

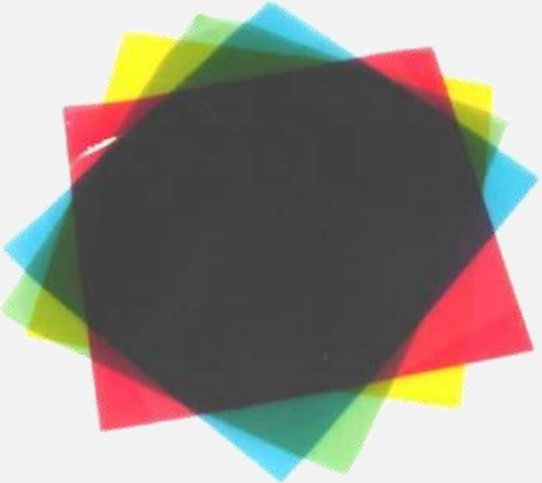


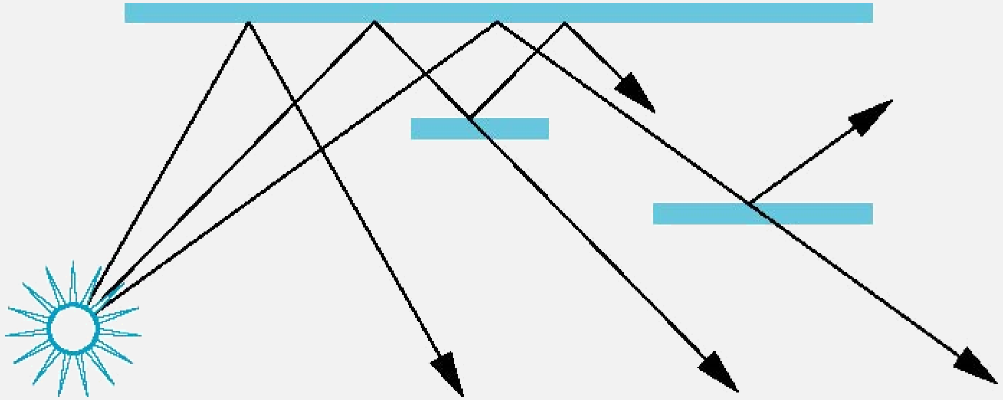
VRay Core 1.49.73 | Render Time 0h 1m 10.4s |

[from google image]

Alpha Blending

- Physically correct model
  - We need to consider complicated inter-surface reflections
  - Difficult in real-time

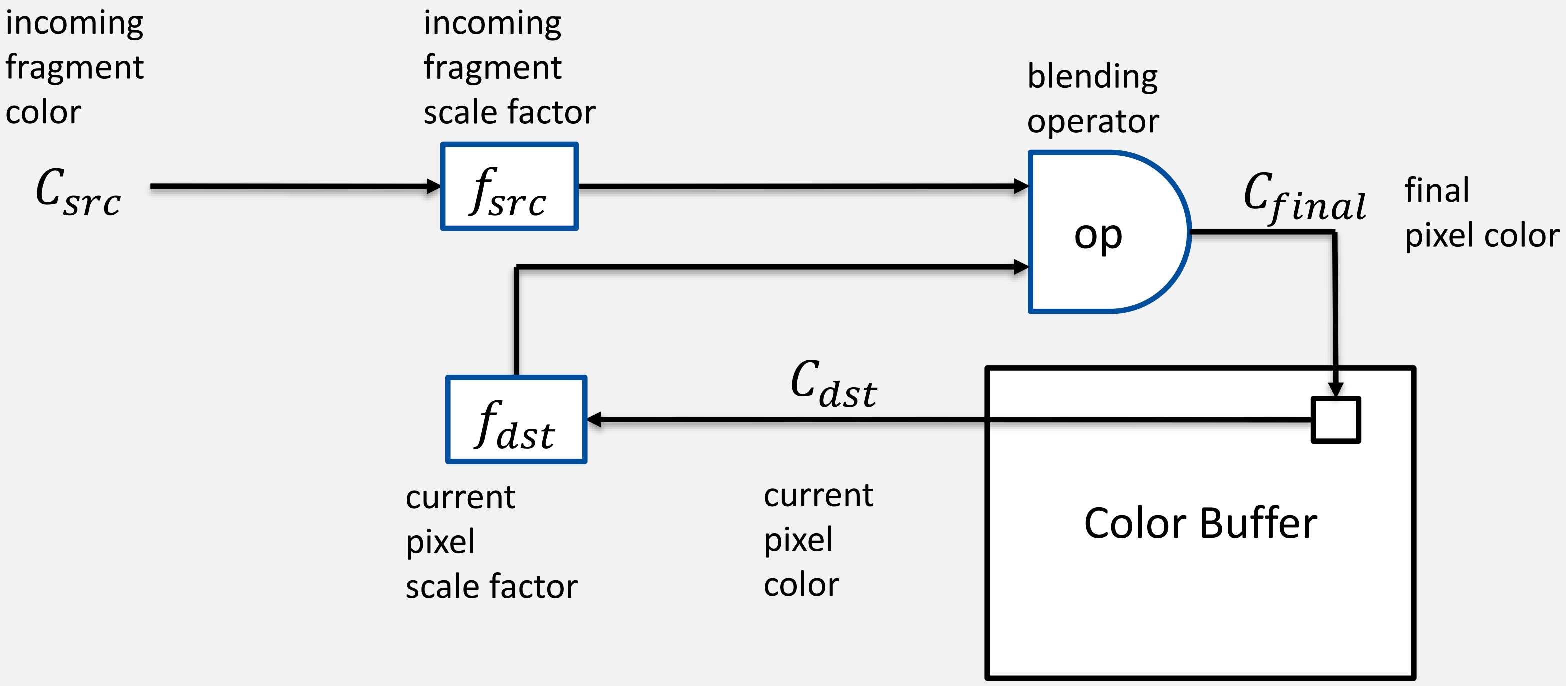




Alpha Blending

- Blending equation in Modern OpenGL

$$\boxed{C_{final} = f_{src}C_{src} \text{ op } f_{dst}C_{dst}}$$



Alpha Blending – Painter's Algorithm

- We render polygons a back to front order for alpha blending
  - Even though graphics HW supports the z-buffer algorithm, we should use painter's algorithm for alpha blending

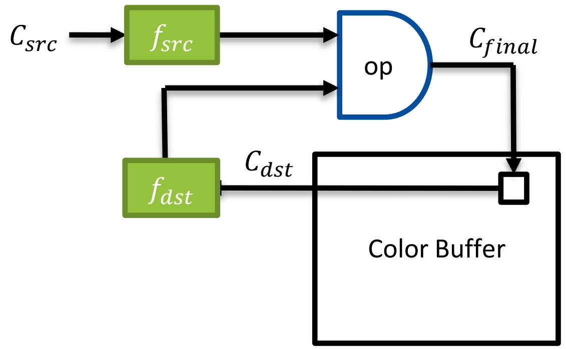
ATI
RADEON
PREMIUM
GRAPHICS

[AMD DirectX 11 Demo for H/W accelerated alpha blending]
(video, youtube)

Alpha Blending

- glBlendFunc(): specify pixel arithmetic
- glBlendFuncSeperate(): specify pixel arithmetic for RGB / A separately

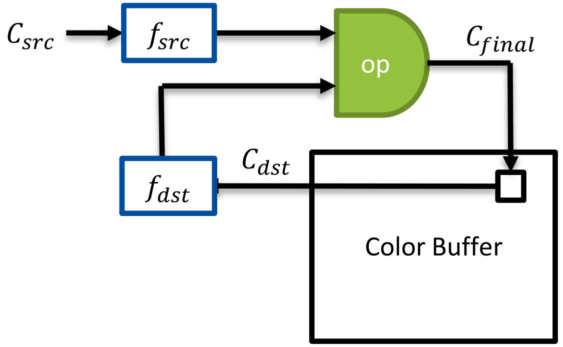
```
// sfactor  specifies how the red, green, blue, and alpha source blending factors are computed.
// dfactor  specifies how the red, green, blue, and alpha destination blending factors are computed.
//
//          Parameters                          RGBA blending factors
//          GL_ZERO                             0 0 0 0
//          GL_ONE                              1 1 1 1
//          GL_SRC_COLOR                        Rs Gs Bs As
//          GL_ONE_MINUS_SRC_COLOR              1-Rs 1-Gs 1-Bs 1-As
//          GL_SRC_ALPHA                        As As As As
//          GL_ONE_MINUS_SRC_ALPHA              1-As 1-As 1-As 1-As
//          GL_DST_COLOR                        Rd Gd Bd Ad
//          GL_ONE_MINUS_DST_COLOR              1-Rd 1-Gd 1-Bd 1-Ad
//          GL_DST_ALPHA                        Ad Ad Ad Ad
//          GL_ONE_MINUS_DST_ALPHA              1-Ad 1-Ad 1-Ad 1-Ad
//          GL_CONSTANT_COLOR                   Rc Gc Bc Ac             constant blending color
//          GL_ONE_MINUS_CONSTANT_COLOR         1-Rc 1-Gc 1-Bc 1-Ac     (Rc, Gc, Bc, Ac)
//          GL_CONSTANT_ALPHA                   Ac Ac Ac Ac             comes from
//          GL_ONE_MINUS_CONSTANT_ALPHA         1-Ac 1-Ac 1-Ac 1-Ac     glBlendColor()
//          GL_SRC_ALPHA_SATURATE               min(As, 1-Ad)  1

void glBlendFunc(GLenum sfactor, GLenum dfactor);
void glBlendFuncSeparate(GLenum srcRGB, GLenum dstRGB, GLenum srcAlpha, GLenum dstAlpha);
```



Alpha Blending

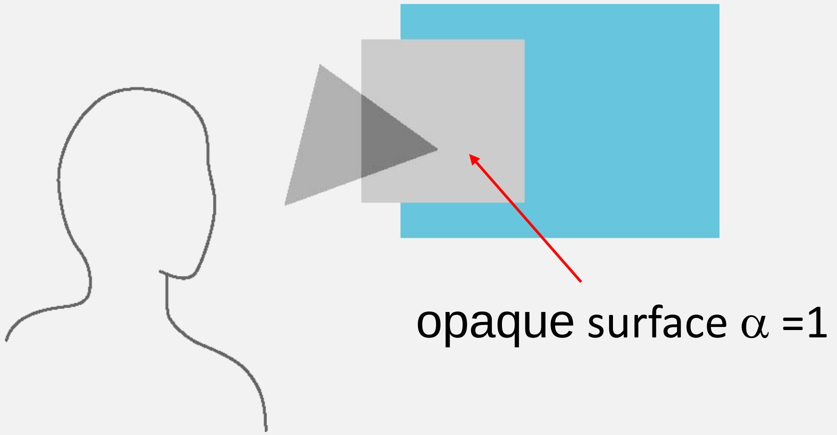
- glBlendEquation(): specify the blend equation for RGBA
- glBlendEquationSeperate(): specify the blend equations for RGB / A separately

```
// mode     specifies how source and destination colors are combined.
//
//          Parameters                          Equation
//          GL_FUNC_ADD                         SRC + DST
//          GL_FUNC_SUBTRACT                    SRC - DST
//          GL_FUNC_REVERSE_SUBTRACT            DST - SRC
//
// For these equations all color components are understood
// to have values in the range [0, 1].
// The results of these equations are clamped to the range [0, 1].


void glBlendEquation(GLenum mode);
void glBlendEquationSeparate(GLenum modeRGB, GLenum modeAlpha);
```



Alpha Blending

- Alpha Blending in OpenGL
  - z-buffer test?
  - Rendering order about several objects?
    - Blending functions are order dependent
    - Opaque polygons block all polygos behind them and affect the depth buffer
    - Translucent polygons should not affect depth buffer
    - Sort polygons first to remove order dependency



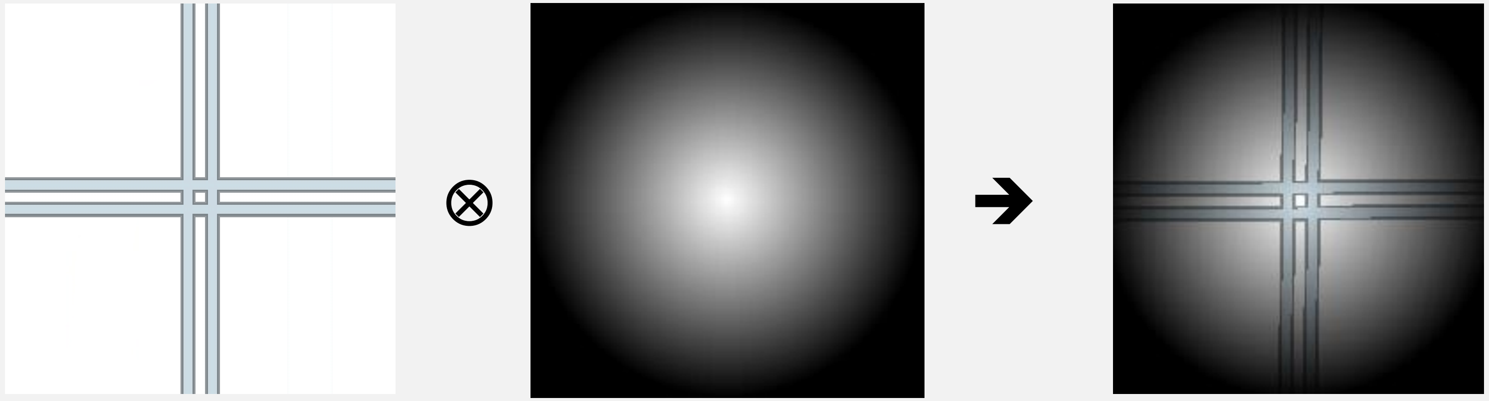
Using Alpha Blending in Modern OpenGL

- Set Pixelformt of frame buffer as RGBA
- Draw an Object with alpha channel
  - When using texture, texel should have alpha channel
- Disable depth test in OpenGL
  - glDepthMask(GL_FALSE) or glDisable(GL_DEPTH_TEST);
- Enable blending function in OpenGL
  - glBlendFunc(…) or glBlendFuncSeparate(…)
  - glBlendEquation(…) or glBlendEquationSeparate(…)
  - glEnable(GL_BLEND);
- Draw objects with a carefully selected order
- Disable blending function in OpenGL
  - glDisable(GL_BLEND);
- Enable depth test in OpenGL
  - glDepthMask(GL_TRUE) or glEnable(GL_DEPTH_TEST);

Multi-Texturing with Programmable Rendering Pipeline

Multi-Texturing

- Texture blending
  - To generate a new texture image by blending given two or more textures
  - Very common operation in fragment shaders
    - ex) Precomputed lighting, normal maps

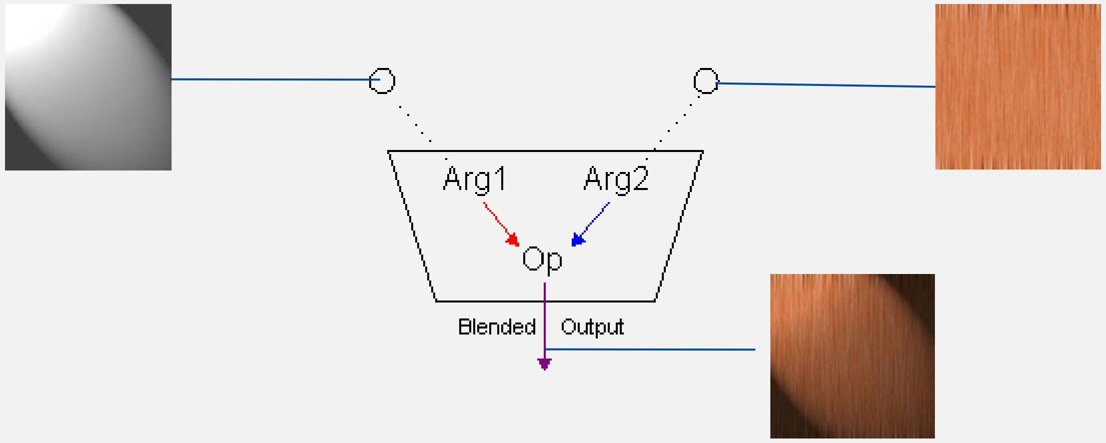


Multi-Texturing

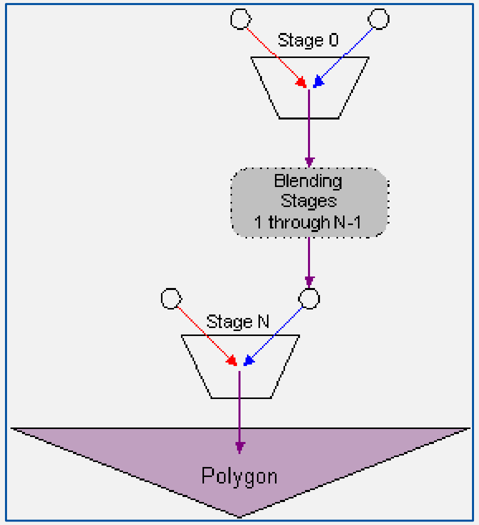
- Quake 2
  - http://www.bigpanda.com/trinity/article1.html

Multi-Texturing

- You can imagine a coffee machine to understand multi-texturing
  - In the fixed rendering pipeline of DirectX 9 and OpenGL



1-texture stage



2-texture stages

Multi-Texturing

Fragment Shaders

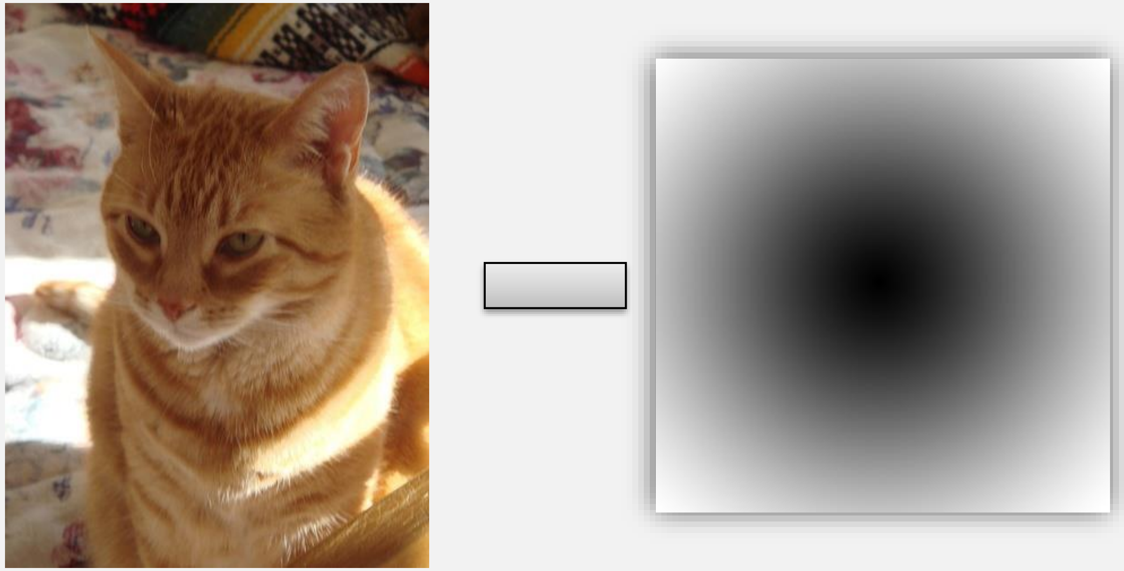
```
/// Multitexture Fragment Shader

#version 120                  // GLSL 1.20

uniform sampler2D s_basemap;
uniform sampler2D s_lightmap;

varying vec2 v_texcoord;

void main()
{
  vec4 base_color;
  vec4 light_color;

  base_color  = texture2D(s_basemap, v_texcoord);
  light_color = texture2D(s_lightmap, v_texcoord);

  glFragColor = base_color * (light_color + 0.25);
}
```

Modern OpenGL codes (C/C++)

```
GLuint  basemap_id, lightmap_id;
GLuint  loc_s_basemap, loc_s_lightmap;


// Bind the base map
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, basemap_id);

// Set the base map sampler to texture unit 0
glUniform1i(loc_s_basemap, 0);


// Bind the light map
glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_2D, lightmap_id);

// Set the base map sampler to texture unit 1
glUniform1i(loc_s_lightmap, 1);
```

Practice – Multi-Texturing

- Multi-texture
  - Activate texture units
  - Operation between textures



Multi-Texturing

Texture mapping

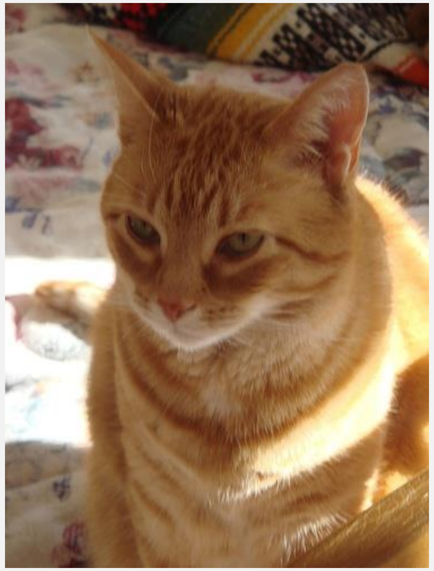
- Uniform / attribute
  - model-view-projection matrix
  - Vertex, texture coordinate
- Vertex shader
  - Transform vertex
  - Send texture coordinate
- Fragment shader
  - Access a texture using sampler
  - Mix teture colors

Vertex / Fragment Shaders

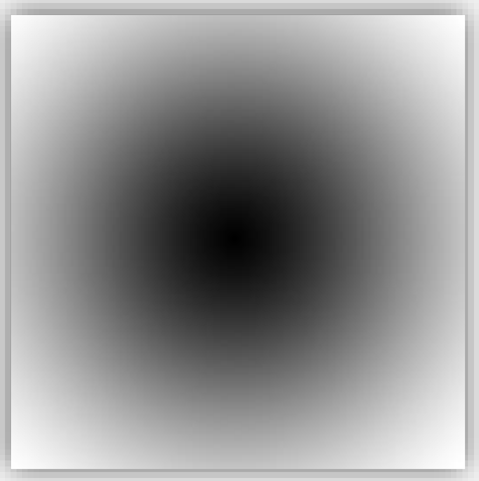
```
#version 120      // GLSL 1.20

uniform mat4 u_PVM;

attribute vec4 a_vertex;
attribute vec2 a_texcoord;

varying vec2 v_texcoord;

void main()
{
  gl_Position = u_PVM * a_vertex;
  v_texcoord = a_texcoord;
}
```

```
#version 120      // GLSL 1.20

uniform sampler2D s_cat;
uniform sampler2D s_gradient;

varying vec2 v_texcoord;

void main()
{
  vec4 cat_color = texture2D(s_cat, v_texcoord);
  vec4 gradient = texture2D(s_gradient, v_texcoord);
  gl_FragColor = cat_color - gradient;
}
```

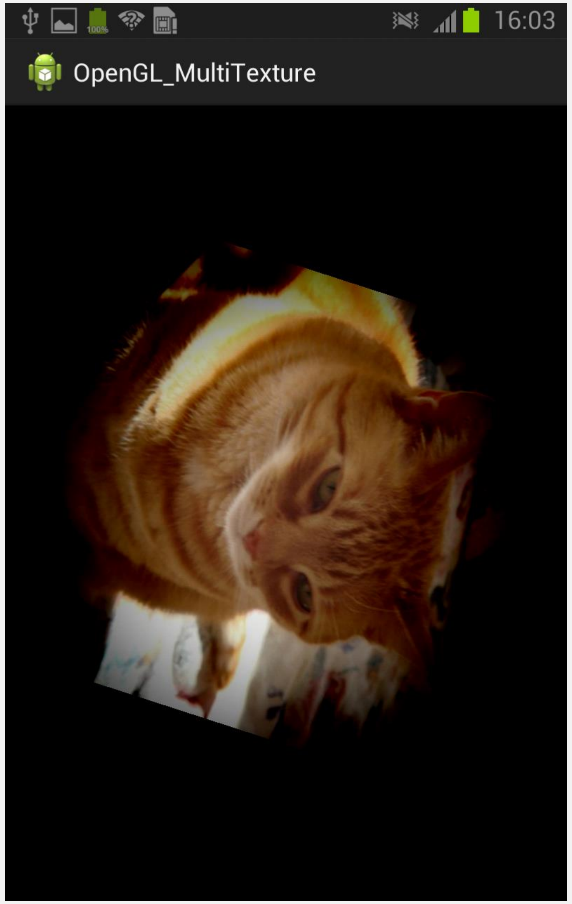
Multi-Texturing



−

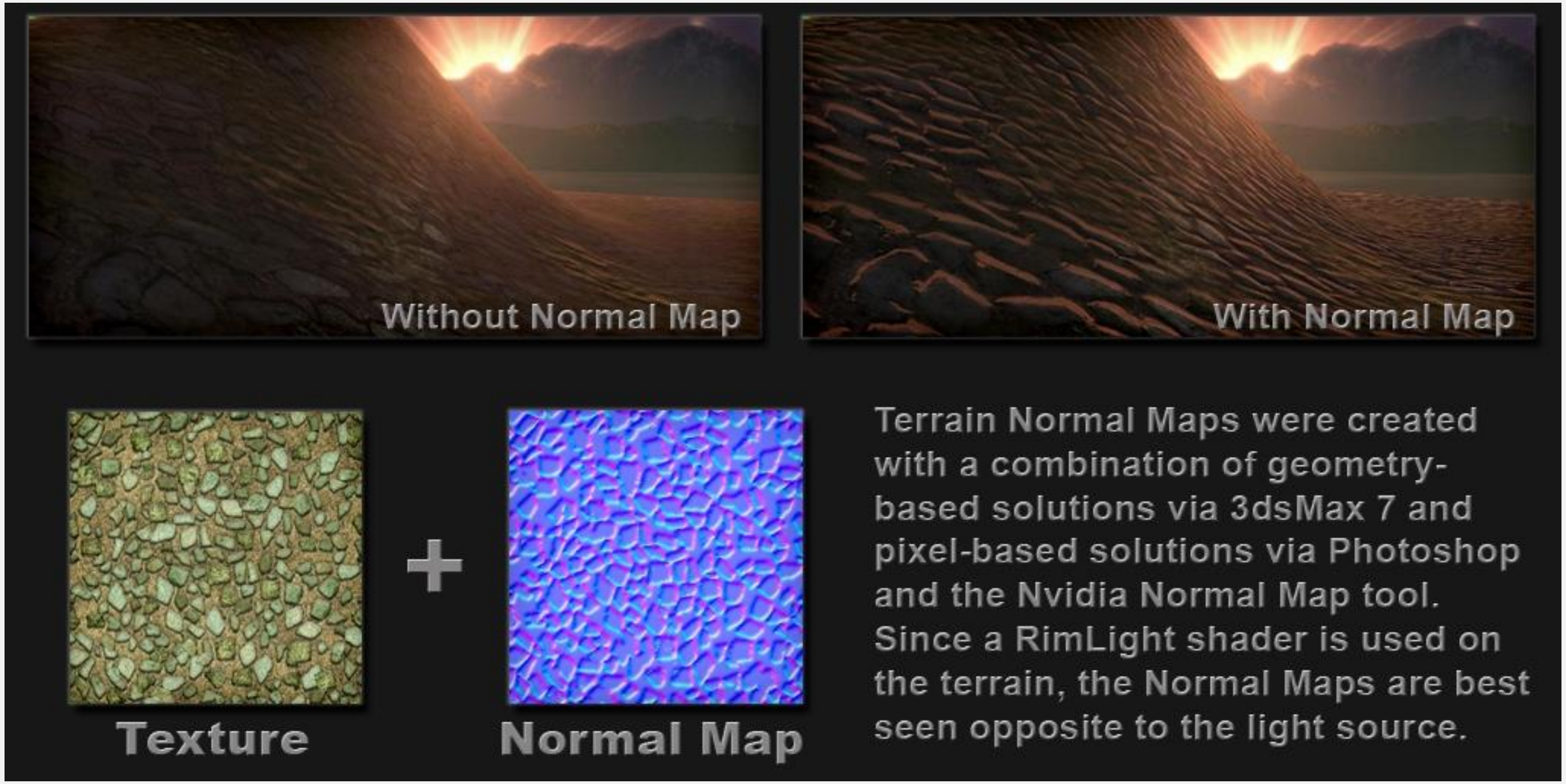


=



Normal Mapping

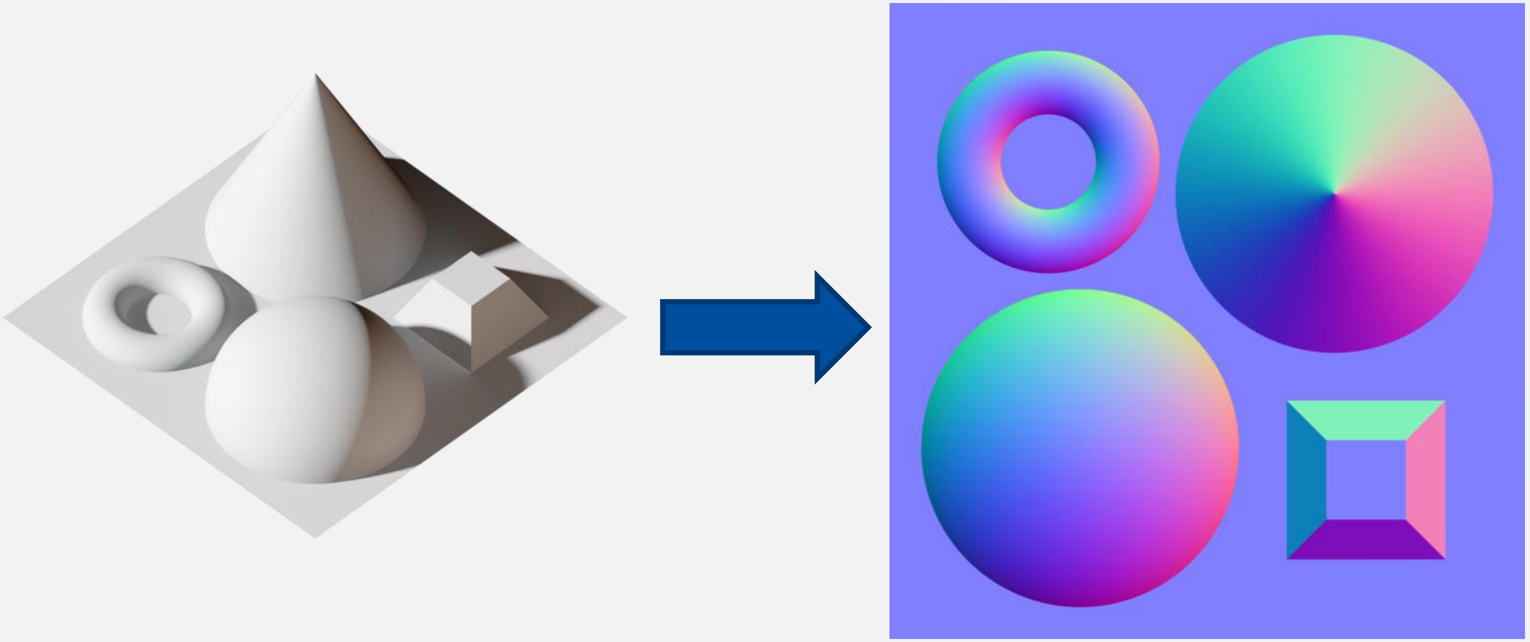
- Available w/ Pixel Shaders



Terrain Normal Maps were created with a combination of geometry-based solutions via 3dsMax 7 and pixel-based solutions via Photoshop and the Nvidia Normal Map tool. Since a RimLight shader is used on the terrain, the Normal Maps are best seen opposite to the light source.

Practice – Normal Mapping

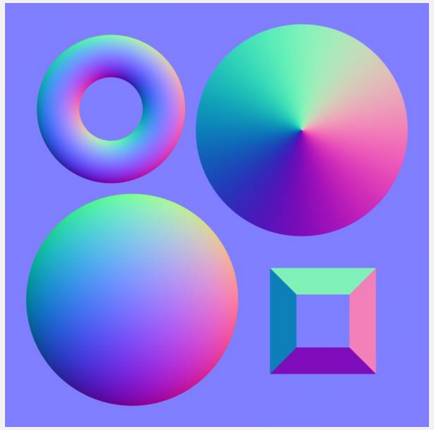
- Normal map
  - Image
  - RGB as a normal vector

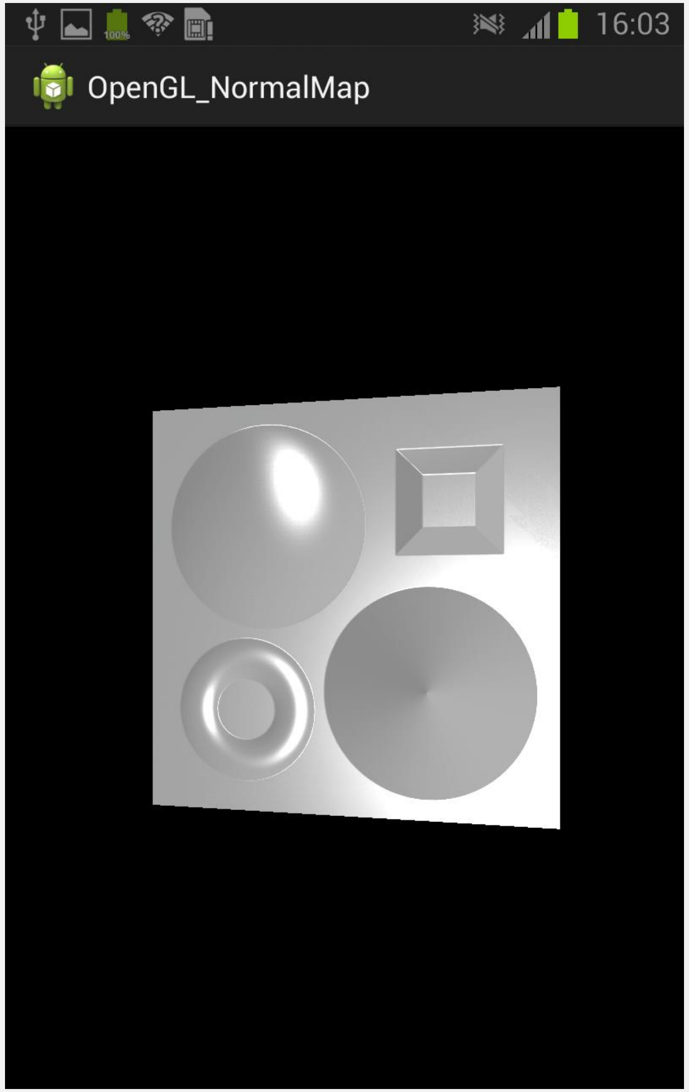


http://en.wikipedia.org/wiki/Normal_mapping

Normal Mapping

- Lighting with normal map





16:03

OpenGL_NormalMap

Image Processing

- Image contrast, brightness, blur, sharpness, saturation processing

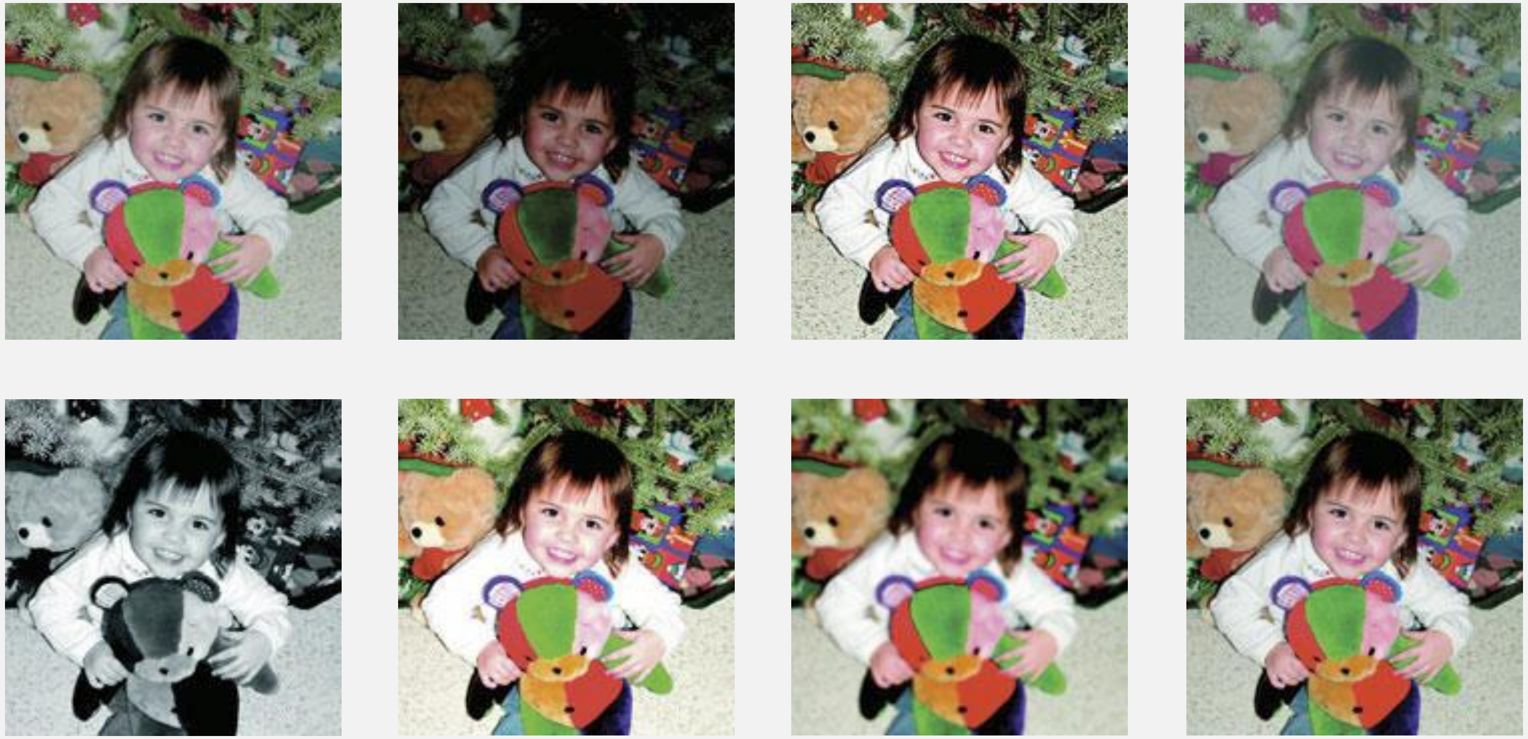


Image Processing

- Convolution
  - Common image processing operation
  - Simple image filtering
  - Smoothing, edge detection, sharpening …

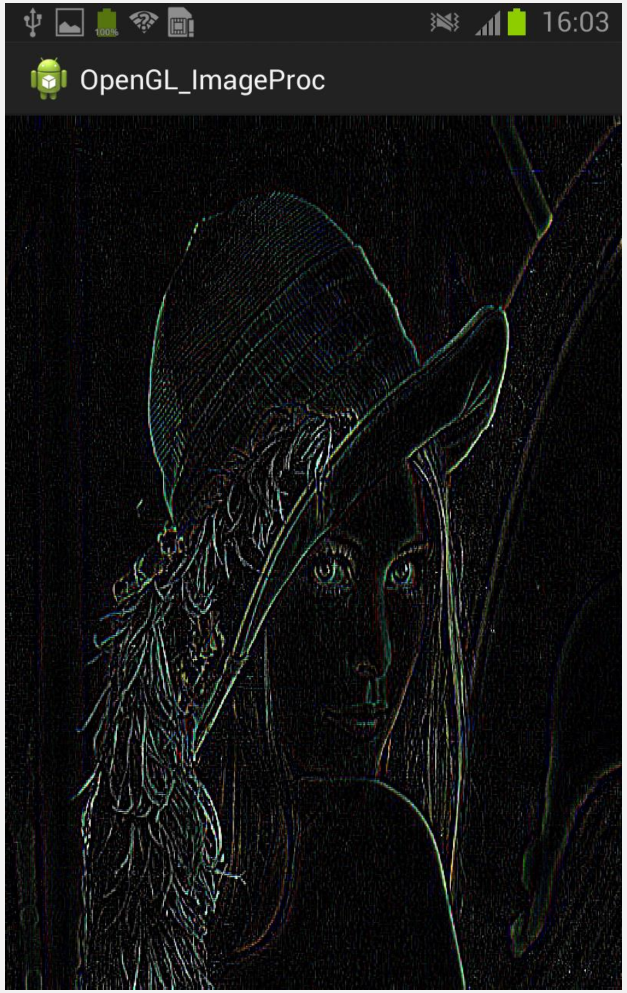
×

0	-1	0
-1	4	-1
0	-1	0

edge detection kernel

Image Processing

- Highlighted edges


16:03
OpenGL_ImageProc