심화전공실습 (CGL)

HW13_Texture_Mapping

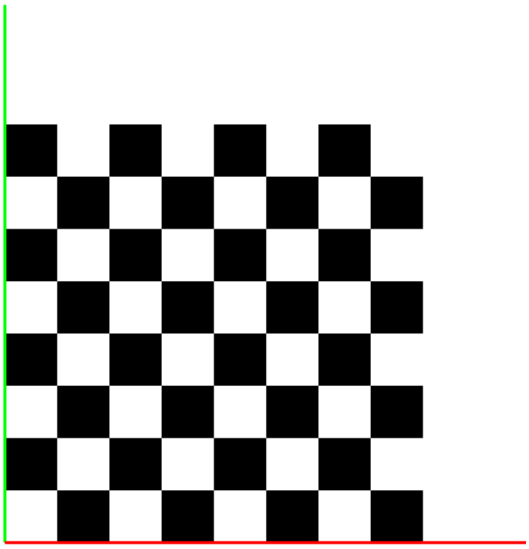
Self-scoring table

	P1	P2	P3	P4	E1	Total
Score	**1**	**1**	**1**	**1**	**1**	**5**

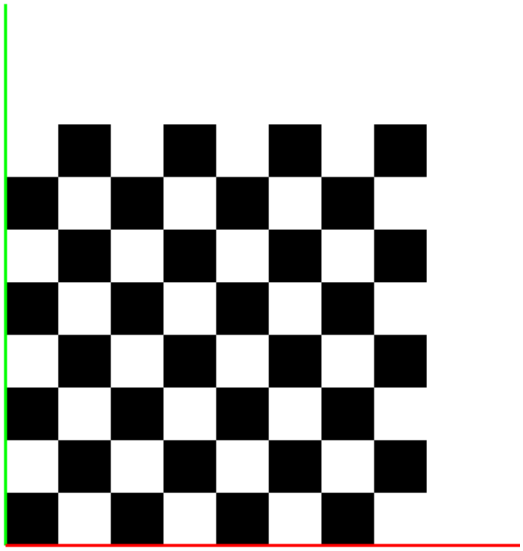
2018707068 김경환

KwangWoon University

Practice 01. Checkerboard texture:



Checkerboard texture mapping



Upside down

왼쪽과 같은 8x8 Checkerboard Pattern은 64x64 2D array로 만들 수 있다.

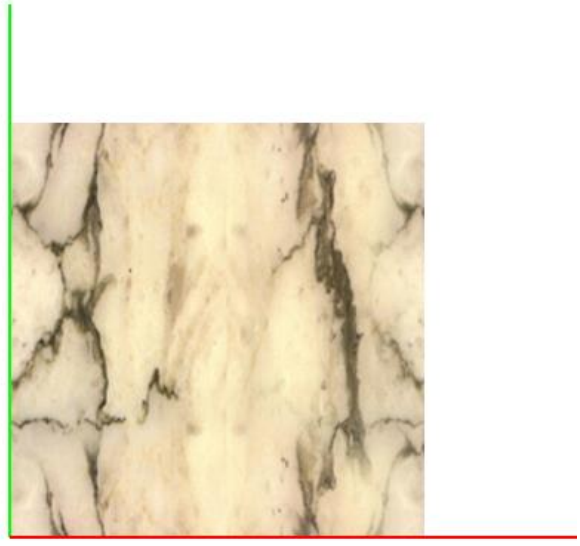
이는 64x64 array에 (((I & 0x8) == 0) ^ ((j & 08) == 0)) * 255;의 결과를 저장하고,

glTexImage2D()로 target = GL_TEXTURE_2D, level = 0, internal format = GL_RGB8, width, height = 64, 64, border = 0, format = GL_LUMINANCE, type = GL_UNSIGNED_BYTE, pixels = checkerboard를 지정한다.

여기서 64x64 array에 저장되는 texel의 경우 비트 연산을 통해 행과 열을 8 pixel 씩 block으로 나눠 XOR을 취해 0 또는 255가 저장된다. 또한 format을 GL_LUMINANCE로 줘서 흑백의 효과를 준다.

관습적으로 Image 좌표의 원점은 좌측상단이고, OpenGL에서 texture 좌표의 원점은 좌측하단이다. 따라서 두 좌표가 Upside down되어 있는 것을 알 수 있다. 그리하여 다시 이야기하면 위 사진에서 왼쪽은 texture를 Image 좌표에 맞춰 texture 좌표를 Upside down한 것이고, 오른쪽은 texture를 texture 좌표 그대로 그린 것을 알 수 있다. 이러한 Upside down은 코드에서 부여된 glVertex3f()의 좌표를 따라서 glTexCoord2f()의 좌표의 순서를 바꿔 구현할 수 있다.

Practice 02. Texture files in the raw format (marble and logo example):



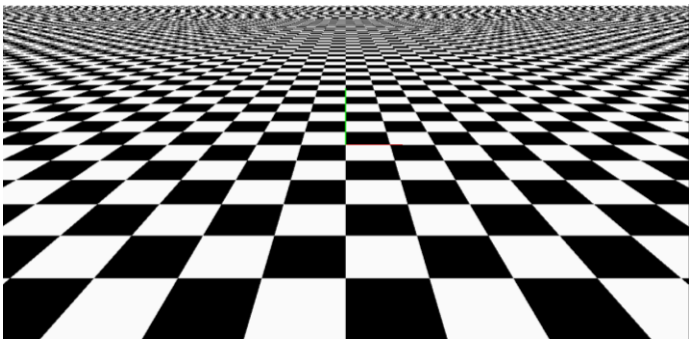
Marble texture mapping

Logo texture mapping

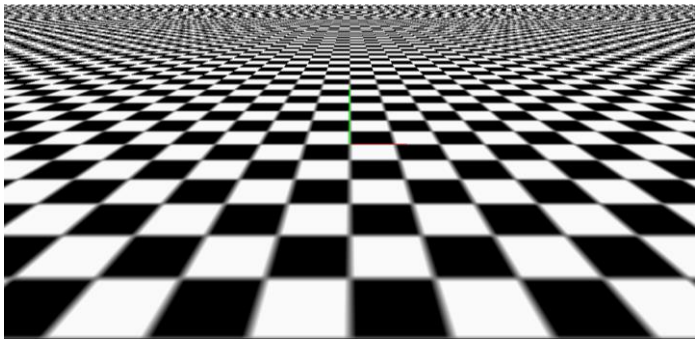
위 두 사진은 raw format의 .row file을 읽어 texture mapping한 모습이다.
이는 width x height의 .raw file에서 각 pixel마다 3 bytes씩 R, G, B value가 있으므로 [width x height x 3] 크기의 Glubyte를 읽고 저장하여 이를 glTexImage2D()의 pixel로 전달하면서 format을 GL_RGB로 바꿔 구현할 수 있다.

Practice 03. Antialiasing with mipmapping (floor example):

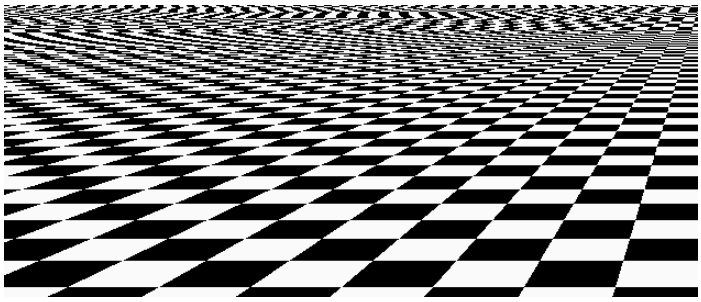
아래의 사진들은 checkerboard pattern으로 texture된 large floor이다.



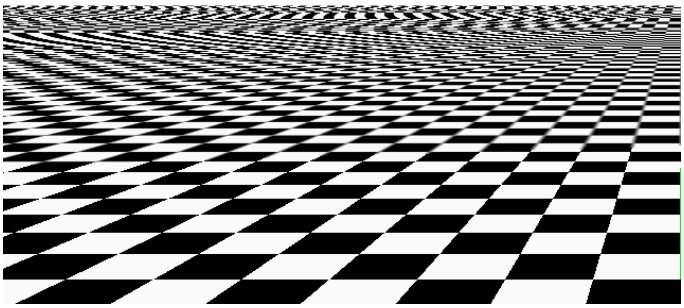
Magnification의 방법으로 GL_NEAREST를 준 모습



Magnification의 방법으로 GL_LINEAR를 준 모습

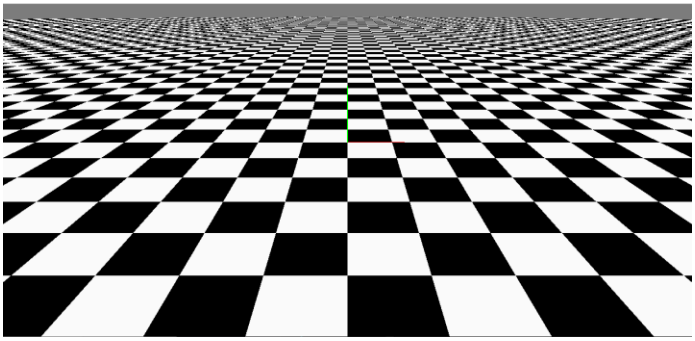


Minification의 방법으로 GL_NEAREST를 준 모습

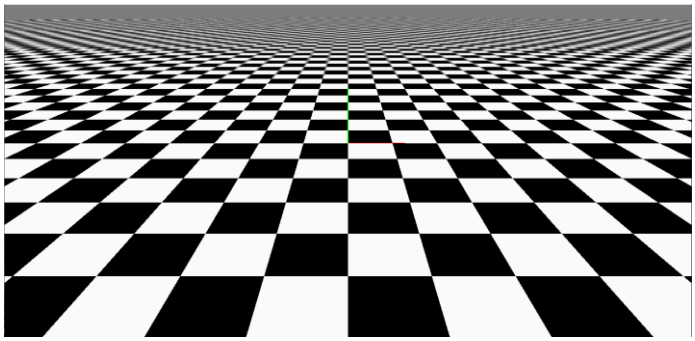


Minification의 방법으로 GL_LINEAR를 준 모습

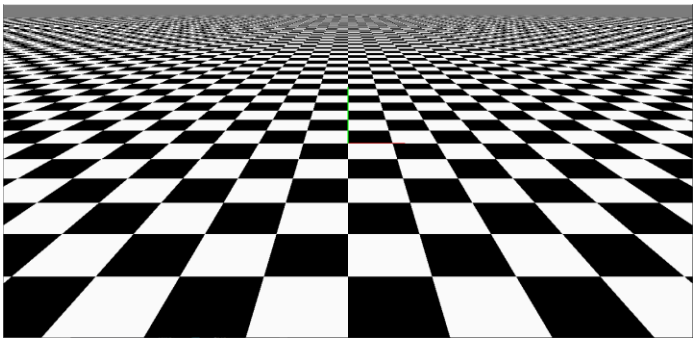
GL_NEAREST의 경우에는 texture 좌표와 가장 가까운 texel을 선택하는 point sampling이 설정된다.
GL_LINEAR의 경우에는 texture 좌표의 이웃한 texel에서 보간된 값을 가져와 texel 사이의 색상을 근사치로 가져오는 bilinear filtering으로 설정된다.
이러한 효과에서 GL_NEAREST는 텍스처를 형성하는 픽셀들을 명확히 볼 수 있는 차단된 패턴을 생성하는 반면 GL_LINEAR는 개별 픽셀들이 덜 보이는 더 매끄러운 패턴을 생성한다.
그리하여 위 4개의 사진에서 왼쪽보다 오른쪽이 더 매끄러운 패턴을 생성하지만 blurring이 보인다.



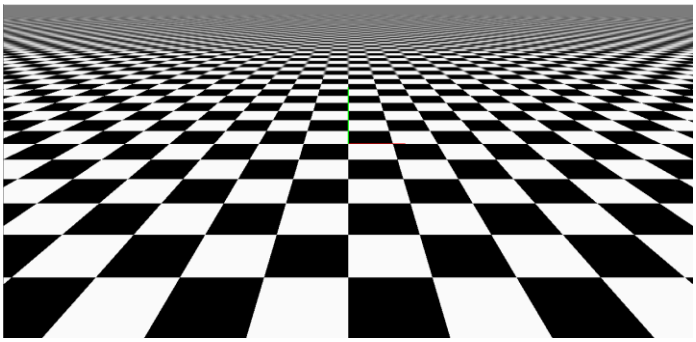
GL_NEAREST_MIPMAP_NEAREST



GL_LINEAR_MIPMAP_NEAREST



GL_NEAREST_MIPMAP_LINEAR



GL_LINEAR_MIPMAP_LINEAR

OpenGL에서 Mipmapping은 Minification을 위한 것이므로 위 4개 모두 Minification에 대한 설정이다.

GL_NEAREST_MIPMAP_NEAREST은 visible divion의 각 Level 내에서는 point sampling(GL_NEAREST)을 하고 MIPMAP에서는 가장 가까운 level 하나만을 사용한다.

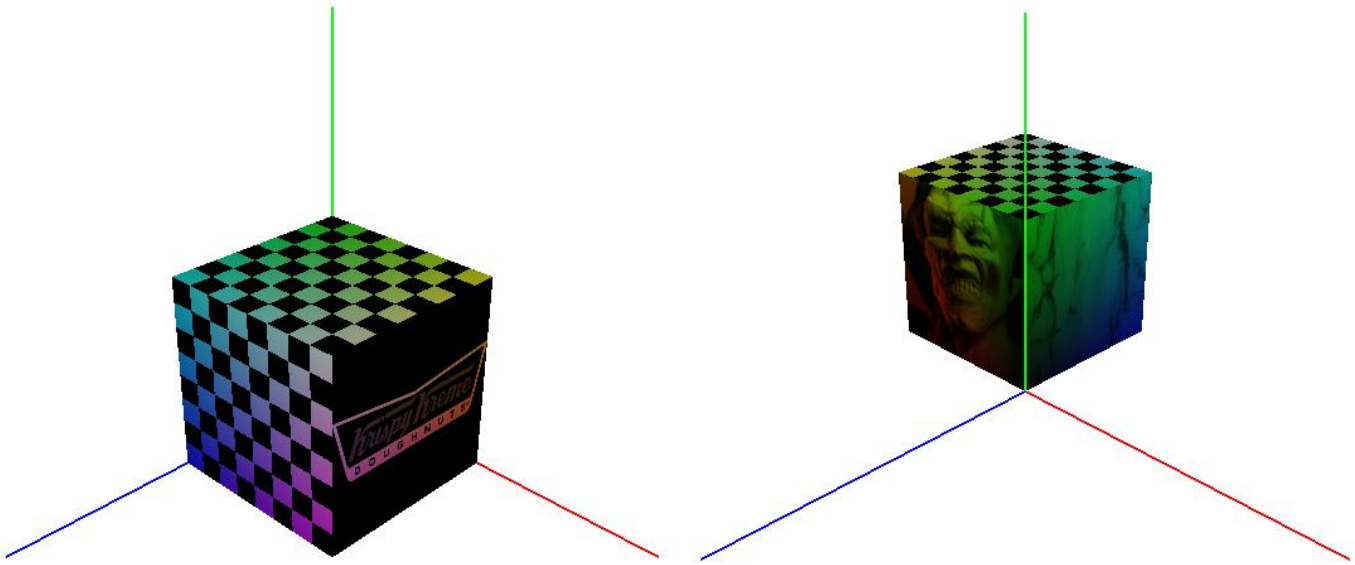
GL_LINEAR _MIPMAP_NEAREST은 visible divion의 각 Level 내에서는 linear interpolation(GL_LINEAR)을 하고 MIPMAP에서는 가장 가까운 level 하나만을 사용한다.

GL_NEAREST_MIPMAP_LINEAR은 visible divion의 각 Level 내에서는 point sampling(GL_NEAREST)을 하고 MIPMAP에서는 가장 가까운 두 level 사이의 linear interpolation (GL_LINEAR)을 한다.

GL_LINEAR_MIPMAP_LINEAR은 visible divion의 각 Level 내에서는 linear interpolation(GL_LINEAR)을 하고 MIPMAP에서는 가장 가까운 두 level 사이의 linear interpolation(GL_LINEAR)을 한다.

결과를 보면 GL_NEAREST_MIPMAP_NEAREST은 linear interpolation을 하지 않기 때문에 visible division이 심하게 나타나고, GL_LINEAR _MIPMAP_NEAREST은 bilinear interpolation을 하기 때문에 어느정도 괜찮은 모습을 보인다. 그리고 GL_NEAREST_MIPMAP_LINEAR linear interpolation을 하지만 biliner는 아니기 때문에 권장할만한 모습은 보이지 않는다. 마지막으로 GL_LINEAR_MIPMAP_LINEAR는 trilinear interpolation을 하기 때문에 Level of Detail 사이의 "seams"가 제거된 것을 볼 수 있다.

Practice 04. Texture mapping to a cube:



Cube의 6면에서 2면은 checkerboard texture를, 2면은 demon texture를, 1면은 marble texture를, 나머지

1면은 logo texture를 mapping한 모습이다.

Checkboard, marble, logo의 경우에는 위에서 mapping을 하였지만 demon은 mapping된 적이 없다.

Demon texture는 marble, logo와 다르게 raw format의 file이 아닌 header file에 저장되어있다.

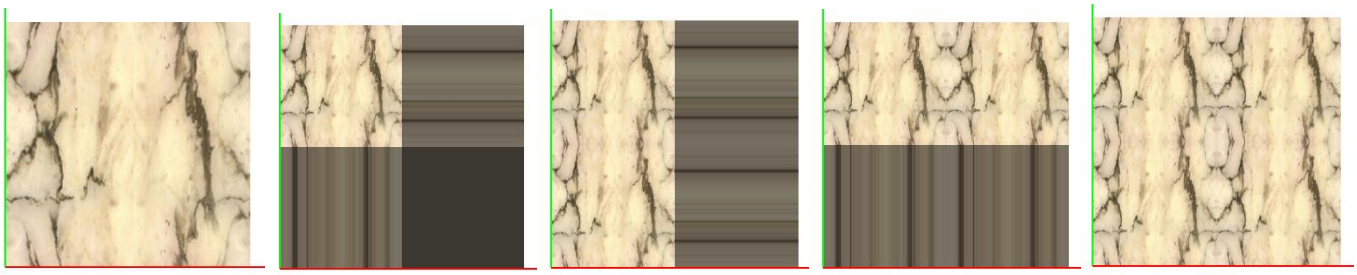
또한, 128x128 GL_RGB8의 이미지이기 때문에 glTexImage2D()로 marble, logo와 같이 format을 GL_RGB으로 설정하고 pixel은 [128x128x3] 크기의 GLubyte를 전달한다.

위 두 사진 중 왼쪽은 MSAA(Multi-Sample Anti-Aliasing)이 적용되지 않은 모습이고, 오른쪽은 MSAA가 적용된 모습이다.

이는 polygon의 edge에 있는 각각의 pixel에 대해서 여러 번 sampling하는 것으로 screen coordinates에서 pixel보다 작은 slight offset을 준다. 그리하여 이러한 edge 근처에서 촘촘히 samling하여 얻은 것을 averaging하여 색상의 smooth transition을 만들어낸다.

그리하여 왼쪽 사진은 polygon의 edge에서 계단현상이 나타나지만 오른쪽 사진은 매끄럽다.

Exercise 01. Texture wrapping using a marble texture:



왼쪽부터 순서대로 사진의 번호를 1~5번으로 부여하겠다.

1번의 경우 texture 좌표를 0~1의 범위로 준 marble texture이다.

2번의 경우는 texture 좌표를 0~2의 범위로 부여하고 s축과 t축으로 값이 1을 벗어나면 GL_CLAMP를 하도록 설정한 것이다.

3번의 경우는 texture 좌표를 0~2의 범위로 부여하고 s축으로는 값이 1을 벗어나면 GL_CLAMP, t축으로

는 값이 1을 벗어나면 GL_REPEAT를 하도록 설정한 것이다

4번의 경우는 texture 좌표를 0~2의 범위로 부여하고 s축으로는 값이 1을 벗어나면 GL_REPEAT, t축으로는 값이 1을 벗어나면 GL_CLAMP를 하도록 설정한 것이다

5번의 경우는 texture 좌표를 0~2의 범위로 부여하고 s축과 t축으로 값이 1을 벗어나면 GL_REPEAT를 하도록 설정한 것이다.

Texture 좌표의 범위를 1보다 크게 부여할 때는 glVertex3f()를 부여하기전에 설정하는 glTexCoord2f()의 좌표를 원하는 s와 t의 repetition값으로 부여하면된다. (Ex, glTexCoord2f(0, 2); glVertex3f(0, 0, 0);)