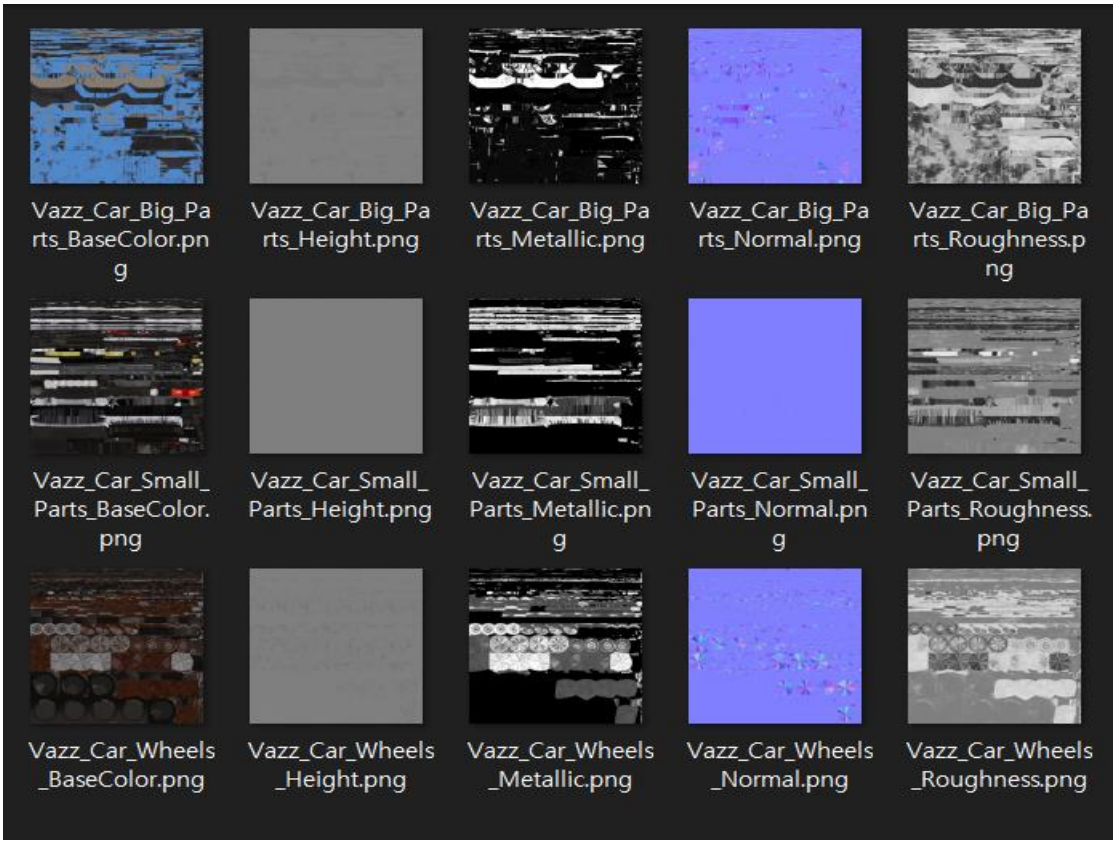
# Project 03

# Advanced Rendering


**컴퓨터그래픽스 01분반**

**디지털이미징 20172979 이효중**

## 1. 모델




이 모델은 자동차 모델로, albedo, normal, roughness, bump, metallic 등 다양한 추가정보를 제공하고 있다.

**/modelload/model_load.cpp**

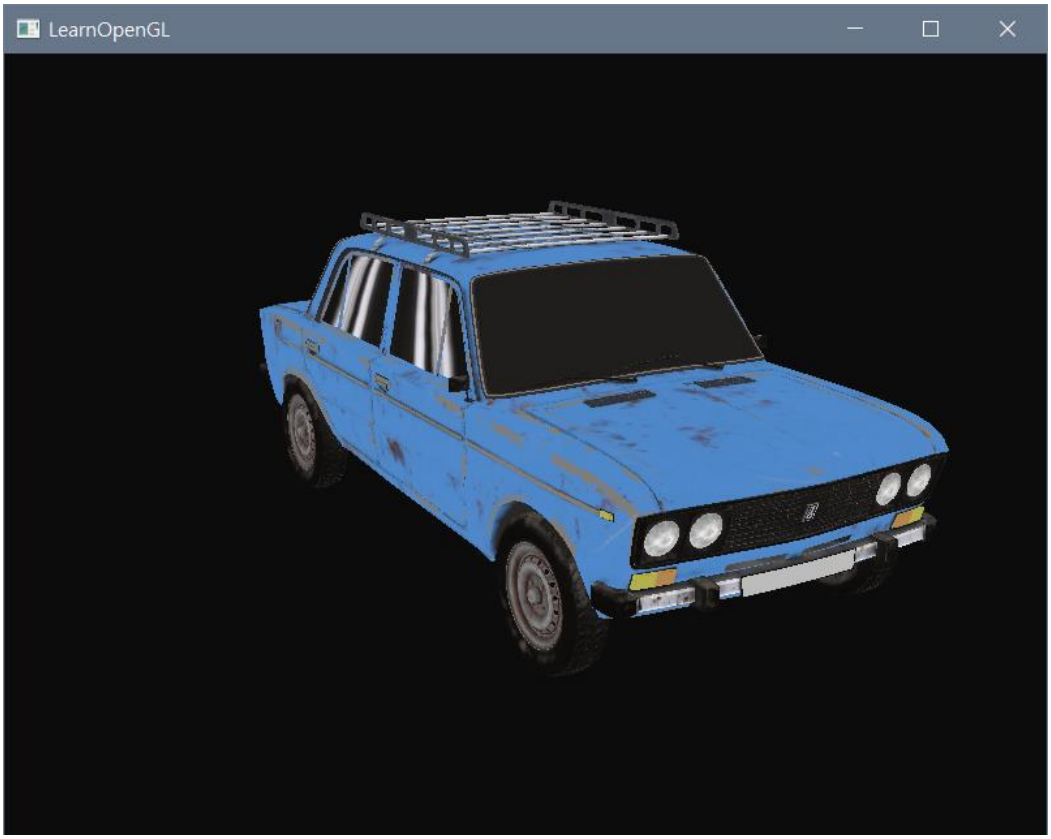
```
// tell stb_image.h to flip loaded texture's on the y-axis (before loading model).
//stbi_set_flip_vertically_on_load(true);

// configure global opengl state
// -----------------------------
glEnable(GL_DEPTH_TEST);

// build and compile shaders
// -------------------------
Shader ourShader("src/1.model_loading.vs", "src/1.model_loading.fs");

// load models
// -----------
Model ourModel(FileSystem::getPath("../resources/cd/Vazz.obj"));
Model ourModel2(FileSystem::getPath("../resources/cd/VazGlass.obj"));
```

Main 함수에서 while 문이 시작되기 전, Shader 를 선언하고 Model 을 불러오기 위한 변수를 선언하였다. 이 모델은 차체와 유리창 두개의 개별 모델로 이루어져 있어 두 개의 Model 변수를 선언하였다. 또한 stbi_set_flip_vertically_on_load() 메소드로 인해 텍스처가 뒤집혀 이상하게 출력되어 이는 주석처리 하였다.

```
// render the loaded model
        glm::mat4 model = glm::mat4(1.0f);
        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f)); // translate it down so it's at the center of
the scene
        model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f)); // it's a bit too big for our scene, so scale it
down
        model = glm::rotate(model, objRotate.pitch(), glm::vec3(1.0f, 0.0f, 0.0f)); //pitch
        model = glm::rotate(model, objRotate.yaw(), glm::vec3(0.0f, 1.0f, 0.0f)); //yaw
        ourShader.setMat4("model", model);
        ourModel.Draw(ourShader);
        ourModel2.Draw(ourShader);
```

또한 선언한 두 Model 을 while 문에 진입하여 ourShader 를 이용해 draw 하였다.

## - 실행 결과



## 2. 조명 Illumination

여기서는 의자 모델을 사용하였다. 손잡이와 다리가 금속 재질, 방석과 등판 부분이 가죽 재질로 이루어져 있다.


또한, 이렇게 albedo, metallic, normal, roughness, ao 에 대한 추가 텍스처 정보를 제공하고 있다.

**- Specular part of the reflectance equation**

$$L_o(p,\omega_o) = \int_\Omega (k_d\frac{c}{\pi} + k_s\frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)})L_i(p,\omega_i)n \cdot \omega_i d\omega_i$$

위와 같은 reflectance equation 의 specular part 에 따라 shader 들이 작동한다.

```
// 물체의 성질
vec3 albedo = pow(texture(albedoMap, TexCoords).rgb, vec3(2.2));
float metallic = texture(metallicMap, TexCoords).r;
float roughness = texture(roughnessMap, TexCoords).r;
float ao = texture(aoMap, TexCoords).r;
```

위와 같이 albedo, metallic, roughness, ao 값을 texture 에서 가져오는데,
albedo 에는 2.2 제곱을 해주어 sRGB 값을 linear RGB 로 바꾸는 과정을 더해준다.

```
// reflectance equation
    // 반사 공식
    vec3 Lo = vec3(0.0);
    for(int i = 0; i < 4; ++i)
    {
        // calculate per-light radiance
        vec3 L = normalize(lightPositions[i] - WorldPos);
        vec3 H = normalize(V + L);
        float distance = length(lightPositions[i] - WorldPos);
        // 거리에 의한 빛의 세기
        float attenuation = 1.0 / (distance * distance);
        vec3 radiance = lightColors[i] * attenuation;

        // Cook-Torrance BRDF
        // 각 빛 마다 DFG 각각을 구함
        float NDF = DistributionGGX(N, H, roughness);
        float G   = GeometrySmith(N, V, L, roughness);
        vec3 F    = fresnelSchlick(max(dot(H, V), 0.0), F0);

        vec3 numerator    = NDF * G * F;
        float denominator = 4 * max(dot(N, V), 0.0) * max(dot(N, L), 0.0) + 0.0001; // + 0.0001 to prevent divide by
zero
        vec3 specular = numerator / denominator;

         // kS is equal to Fresnel
         // 정반사
        vec3 kS = F;
        // for energy conservation, the diffuse and specular light can't
        // be above 1.0 (unless the surface emits light); to preserve this
        // relationship the diffuse component (kD) should equal 1.0 - kS.
        // 난반사
        vec3 kD = vec3(1.0) - kS;
        // multiply kD by the inverse metalness such that only non-metals
        // have diffuse lighting, or a linear blend if partly metal (pure metals
        // have no diffuse light).
        kD *= 1.0 - metallic;

        // scale light by NdotL
        float NdotL = max(dot(N, L), 0.0);

        // add to outgoing radiance Lo
        Lo += (kD * albedo / PI + specular) * radiance * NdotL; // note that we already multiplied the BRDF by the
Fresnel (kS) so we won't multiply by kS again
    }
```

그 다음, 공식에 따라 반복문을 사용해 Lo 의 값의 적분을 구한다.

식에서 Ks 에 곱해진 D, F, G 를 각각 구하기 위해 차례로, 미세면이 halfway 벡터에 얼마나 정렬 되어 있는지를 나타내는 분포 함수인 DistributionGGX(), 빛의 방향 및 시선 방향 모두를 고려한 geometry function 인 GeometrySmith(), 평균적인 비금속 물질의 F0 값 0.04 와 linear interpolation 한 값을 공식에 적용하는 fresnelSchlick() 등을 활용하고, 그 셋의 곱을 Specular-IBL 공식에 대입하여 Lo 를 구한다.

```
// ambient lighting (we now use IBL as the ambient term)
    vec3 F = fresnelSchlickRoughness(max(dot(N, V), 0.0), F0, roughness);

    vec3 kS = F;
    vec3 kD = 1.0 - kS;
    kD *= 1.0 - metallic;

    vec3 irradiance = texture(irradianceMap, N).rgb;
    vec3 diffuse      = irradiance * albedo;

    // sample both the pre-filter map and the BRDF lut and combine them together as per the Split-Sum approximation to get the IBL specular part.
    const float MAX_REFLECTION_LOD = 4.0;
    vec3 prefilteredColor = textureLod(prefilterMap, R,  roughness * MAX_REFLECTION_LOD).rgb;
    vec2 brdf  = texture(brdfLUT, vec2(max(dot(N, V), 0.0), roughness)).rg;
    vec3 specular = prefilteredColor * (F * brdf.x + brdf.y);

    vec3 ambient = (kD * diffuse + specular) * ao;

    vec3 color = ambient + Lo;

    // HDR tonemapping
    color = color / (color + vec3(1.0));
    // gamma correct
    // linear rgb to srgb
    color = pow(color, vec3(1.0/2.2));

    FragColor = vec4(color , 0.0);
```

그 다음 fresnelSchlickRoughness()를 사용해 ambient lighting 를 구한다. Ks 는 Fresnel 과 같고, 에너지 보존법칙에 의해 Kd 는 합인 1.0 에서 Ks 를 뺀 값을 사용한다. Kd 는 순수 금속은 diffuse 한 reflection 이 없다는 점을 활용하여 1.0-metallic 값을 곱해준다. Irradiance 도 irradianceMap 에서 가져오게 되고, diffuse 는 albedo 에 irradiance 를 곱한 값으로 한다.

그 다음, Pre-filtermap 과 BRDF LUT 를 split-sum approximation 에 따라 각각 곱해서 IBL Specular part 를 얻는다.

그 후 HDR tone 보정, 감마 보정(linear RGB -> sRGB)의 과정을 거쳐 FragColor 를 결정하게 된다.

여기서 Texture 를 적용하기 위해서는 shader 에 다음과 같이 변화를 줘야하는 데,

```
// material maps
// 면의 기본 색상(금속 = F0)
// 법선 방향
// 금속성 / 비금속성
// 면의 거친정도
// Ambient Occlusion
uniform sampler2D albedoMap;
uniform sampler2D normalMap;
uniform sampler2D metallicMap;
uniform sampler2D roughnessMap;
uniform sampler2D aoMap;
```

먼저, 물체의 다섯가지 속성을 나타내는 sampler2D 변수를 선언한다. 이는 각각의 텍스처를 불러와 저장할 변수가 된다. 그 다음 normal mapping 을 사용하기 위해 getNormalFromMap() 함수를 정의한다.

```
vec3 getNormalFromMap()
{
    vec3 tangentSpaceNormal = texture(normalMap, TexCoords).xyz * 2.0 - 1.0;

    vec3 Q1  = dFdx(WorldPos);
    vec3 Q2  = dFdy(WorldPos);
    vec2 st1 = dFdx(TexCoords);
    vec2 st2 = dFdy(TexCoords);

    vec3 N   = normalize(Normal);
    vec3 T  = normalize(Q1*st2.t - Q2*st1.t);
    vec3 B  = -normalize(cross(N, T));
    mat3 TBN = mat3(T, B, N);

    return normalize(TBN * tangentSpaceNormal);
}
```

normalMap 에서 가져온 normal 을 TBN 을 곱하여 tangent space 의 normal 을 world space 의 normal 로 변환한다.

그리고 C 소스 코드에서는

```
// build and compile shaders
    // -------------------------
    Shader pbrShader("src/2.2.1.pbr.vs", "src/2.2.1.pbr.fs");
    Shader equirectangularToCubemapShader("src/2.2.1.cubemap.vs", "src/2.2.1.equirectangular_to_cubemap.fs");
    Shader irradianceShader("src/2.2.1.cubemap.vs", "src/2.2.1.irradiance_convolution.fs");
    Shader prefilterShader("src/2.2.1.cubemap.vs", "src/2.2.1.prefilter.fs");
    Shader brdfShader("src/2.2.1.brdf.vs", "src/2.2.1.brdf.fs");
    Shader backgroundShader("src/2.2.1.background.vs", "src/2.2.1.background.fs");

    // Model load
    Model ourModel(FileSystem::getPath("../resources/chair/old chair.obj"));

    pbrShader.use();
    pbrShader.setInt("irradianceMap", 0);
    pbrShader.setInt("prefilterMap", 1);
    pbrShader.setInt("brdfLUT", 2);
    pbrShader.setInt("albedoMap", 3);
    pbrShader.setInt("normalMap", 4);
    pbrShader.setInt("metallicMap", 5);
    pbrShader.setInt("roughnessMap", 6);
    pbrShader.setInt("aoMap", 7);

    backgroundShader.use();
    backgroundShader.setInt("environmentMap", 0);

    // load PBR material textures
    // --------------------------
    // chair texture
    unsigned int chairAlbedoMap = loadTex(FileSystem::getPath("../resources/chair/textures/old_chiar_BaseColor.png").c_str());
    unsigned int chairNormalMap = loadTex(FileSystem::getPath("../resources/chair/textures/old_chiar_Normal.png").c_str());
    unsigned int chairMetallicMap = loadTex(FileSystem::getPath("../resources/chair/textures/old_chiar_Metalness.png").c_str());
    unsigned int chairRoughnessMap = loadTex(FileSystem::getPath("../resources/chair/textures/old_chiar_Roughness.png").c_str());
    unsigned int chairAOMap = loadTex(FileSystem::getPath("../resources/chair/textures/old_chiar_AO.png").c_str())
```

main()에서 우리가 사용할 의자 model 을 선언하고 shader 에 전달하기 위해 albedo, normal, roughness, ao map 변수들을 연결해준다. 그리고 loadTex 함수를 사용하여 텍스처를 load 하는데,

```
unsigned int loadTex(char const* path)
{
    unsigned int texID;
    glGenTextures(1, &texID);

    int w, h, nrComp;
    unsigned char* data = stbi_load(path, &w, &h, &nrComp, 0);
    if (data)
    {
        GLenum format;
        if (nrComp == 1)
            format = GL_RED;
        else if (nrComp == 3)
            format = GL_RGB;
        else if (nrComp == 4)
            format = GL_RGBA;

        glBindTexture(GL_TEXTURE_2D, texID);
        glTexImage2D(GL_TEXTURE_2D, 0, format, w, h, 0, format, GL_UNSIGNED_BYTE, data);
        glGenerateMipmap(GL_TEXTURE_2D);

        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

        stbi_image_free(data);
    }
    else
    {
        std::cout << "텍스쳐 불러오기 실패 " << path << std::endl;
        stbi_image_free(data);
    }

    return texID;
}
```

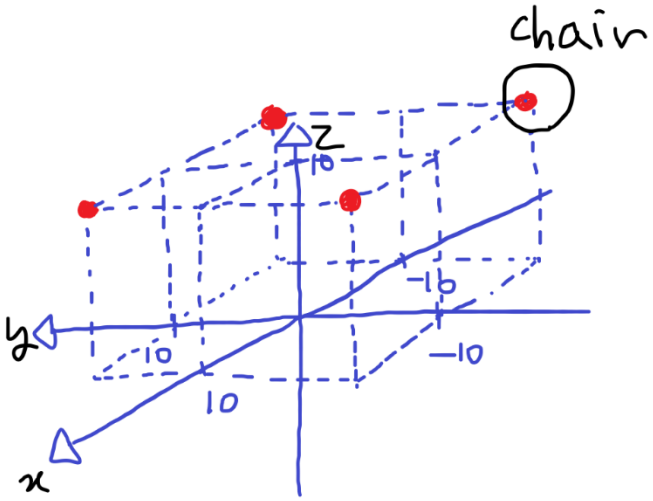
위 함수는 텍스처를 불러와서 색상 구성을 확인한다음, 이를 bind 하고 mipmap 을 생성하여 그 textureID 를 반환하는 함수이다.

**(다음 페이지에 계속)**

이렇게 텍스처 까지 모두 load 했다면

```
// render scene, supplying the convoluted irradiance map to the final shader.
// ------------------------------------------------------------------------------------------
pbrShader.use();
glm::mat4 view = camera.GetViewMatrix();
pbrShader.setMat4("view", view);
pbrShader.setVec3("camPos", camera.Position);

// bind pre-computed IBL data
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_CUBE_MAP, irradianceMap);
glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_CUBE_MAP, prefilterMap);
glActiveTexture(GL_TEXTURE2);
glBindTexture(GL_TEXTURE_2D, brdfLUTTexture);

// chair texture
glActiveTexture(GL_TEXTURE3);
glBindTexture(GL_TEXTURE_2D, chairAlbedoMap);
glActiveTexture(GL_TEXTURE4);
glBindTexture(GL_TEXTURE_2D, chairNormalMap);
glActiveTexture(GL_TEXTURE5);
glBindTexture(GL_TEXTURE_2D, chairMetallicMap);
glActiveTexture(GL_TEXTURE6);
glBindTexture(GL_TEXTURE_2D, chairRoughnessMap);
glActiveTexture(GL_TEXTURE7);
glBindTexture(GL_TEXTURE_2D, chairAOMap);

// Render model
glm::mat4 model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(-10.0f, -10.0f, 10.0f)); // translate it down so it's at the center of the scene
model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f)); // it's a bit too big for our scene, so scale it down
pbrShader.setMat4("model", model)
ourModel.Draw(pbrShader);
```

irradianceMap, prefilterMap, brdfLUTTexture 를 활용하여 Cube 형태의 배경을 그려주고 나서 glActiveTexture(), glBindTexture()를 활용하여 chairAlbedoMap, chairNormalMap, chairMetallicMap, chairRoughnessMap, chairAOMap 을 각각 bind 한다.




그리고 모델을

-10.0, -10.0f, 10.0f 위치로 translate 하여 렌더링하면

그림과 같이 한 광원 위에 의자가 위치하게 된다.

## - 실행 결과




시점이 이동함에 따라 specular point 가 이동하고 조명을 등지고 있는 등받이 뒷부분 같은 곳은 빛이 거의 닿지 않기 때문에 어둡게 잘 표현된 것을 확인할 수 있었다. 또한 금속 부분이 가죽 부분보다 더 매끈하게 잘 반사 되는 것이 표현되는 것을 확인할 수 있었다.