UNIT36: GLTF MODEL - TEXTURES AND IMAGES

【学習要項】
□Textures
□Images

【演習手順】
1．テクスチャの取得
① gltf_model クラスに texture 構造体と image 構造体を定義する

```
struct texture
{
  std::string name;
  int source{ -1 };
};
std::vector<texture> textures;
struct image
{
  std::string name;
  int width{ -1 };
  int height{ -1 };
  int component{ -1 };
  int bits{ -1 };
  int pixel_type{ -1 };
  int buffer_view;
  std::string mime_type;
  std::string uri;
  bool as_is{ false };
};
std::vector<image> images;
std::vector<Microsoft::WRL::ComPtr<ID3D11ShaderResourceView>> texture_resource_views;
```

② gltf_model クラスに fetch_textures メンバ関数を実装する

```
void gltf_model::fetch_textures(ID3D11Device* device, const tinygltf::Model& gltf_model)
{
  HRESULT hr{ S_OK };
  for (const tinygltf::Texture& gltf_texture : gltf_model.textures)
  {
    texture& texture{ textures.emplace_back() };
    texture.name = gltf_texture.name;
    texture.source = gltf_texture.source;
  }
  for (const tinygltf::Image& gltf_image : gltf_model.images)
  {
    image& image{ images.emplace_back() };
    image.name = gltf_image.name;
    image.width = gltf_image.width;
    image.height = gltf_image.height;
    image.component = gltf_image.component;
    image.bits = gltf_image.bits;
    image.pixel_type = gltf_image.pixel_type;
    image.buffer_view = gltf_image.bufferView;
    image.mime_type = gltf_image.mimeType;
    image.uri = gltf_image.uri;
    image.as_is = gltf_image.as_is;

    if (gltf_image.bufferView > -1)
    {
      const tinygltf::BufferView& buffer_view{ gltf_model.bufferViews.at(gltf_image.bufferView) };
      const tinygltf::Buffer& buffer{ gltf_model.buffers.at(buffer_view.buffer) };
      const byte* data = buffer.data.data() + buffer_view.byteOffset;

      ID3D11ShaderResourceView* texture_resource_view{};
      hr = load_texture_from_memory(device, data, buffer_view.byteLength, &texture_resource_view);
      if (hr == S_OK)
      {
```

```
34:         texture_resource_views.emplace_back().Attach(texture_resource_view);
35:       }
36:     }
37:     else
38:     {
39:       const std::filesystem::path path(filename);
40:       ID3D11ShaderResourceView* shader_resource_view{};
41:       D3D11_TEXTURE2D_DESC texture2d_desc;
42:       std::wstring filename{
43:         path.parent_path().concat(L"/").wstring() +
44:         std::wstring(gltf_image.uri.begin(), gltf_image.uri.end()) };
45:       hr = load_texture_from_file(device, filename.c_str(), &shader_resource_view, &texture2d_desc);
46:       if (hr == S_OK)
47:       {
48:         texture_resource_views.emplace_back().Attach(shader_resource_view);
49:       }
50:     }
51:   }
52: }
```

③ load_texture_from_memory 関数の実装例

```
 1: HRESULT load_texture_from_memory(ID3D11Device* device, const void* data, size_t size,
 2:   ID3D11ShaderResourceView** shader_resource_view)
 3: {
 4:   HRESULT hr{ S_OK };
 5:   ComPtr<ID3D11Resource> resource;
 6:
 7:   hr = CreateDDSTextureFromMemory(device, reinterpret_cast<const uint8_t*>(data),
 8:     size, resource.GetAddressOf(), shader_resource_view);
 9:   if (hr != S_OK)
10:   {
11:     hr = CreateWICTextureFromMemory(device, reinterpret_cast<const uint8_t*>(data),
12:       size, resource.GetAddressOf(), shader_resource_view);
13:     _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
14:   }
15:
16:   return hr;
17: }
```

④ gltf_model クラスのコンストラクタから fetch_textures メンバ関数を呼び出す
⑤ gltf_model クラスの render メンバ関数の primitive ループの内側で texture_resource_views オブジェクトをバインドする

```
 1: const material& material{ materials.at(primitive.material) };
 2: const int texture_indices[]
 3: {
 4:   material.data.pbr_metallic_roughness.basecolor_texture.index,
 5:   material.data.pbr_metallic_roughness.metallic_roughness_texture.index,
 6:   material.data.normal_texture.index,
 7:   material.data.emissive_texture.index,
 8:   material.data.occlusion_texture.index,
 9: };
10: ID3D11ShaderResourceView* null_shader_resource_view{};
11: std::vector<ID3D11ShaderResourceView*> shader_resource_views(_countof(texture_indices));
12: for (int texture_index = 0; texture_index < shader_resource_views.size(); ++texture_index)
13: {
14:   shader_resource_views.at(texture_index) = texture_indices[texture_index] > -1 ?
15:     texture_resource_views.at(textures.at(texture_indices[texture_index]).source).Get() :
16:     null_shader_resource_view;
17: }
18: immediate_context->PSSetShaderResources(1, static_cast<UINT>(shader_resource_views.size()),
19:   shader_resource_views.data());
```

２．シェーダーの実装（変更）
① ピクセルシェーダー(gltf_model_ps.hlsl)にテクスチャとサンプラを定義する

```
#define BASECOLOR_TEXTURE 0
#define METALLIC_ROUGHNESS_TEXTURE 1
#define NORMAL_TEXTURE 2
#define EMISSIVE_TEXTURE 3
#define OCCLUSION_TEXTURE 4
Texture2D<float4> material_textures[5] : register(t1);

#define POINT 0
#define LINEAR 1
#define ANISOTROPIC 2
SamplerState sampler_states[3] : register(s0);
```

②ピクセルシェーダー(gltf_model_ps.hlsl)の main 関数を変更する

```
float4 main(VS_OUT pin) : SV_TARGET
{
  material_constants m = materials[material];

  float4 basecolor = m.pbr_metallic_roughness.basecolor_texture.index > -1 ?
    material_textures[BASECOLOR_TEXTURE].Sample(sampler_states[ANISOTROPIC], pin.texcoord) :
    m.pbr_metallic_roughness.basecolor_factor;

  float3 emmisive = m.emissive_texture.index > -1 ?
    material_textures[EMISSIVE_TEXTURE].Sample(sampler_states[ANISOTROPIC], pin.texcoord).rgb :
    m.emissive_factor;

  float3 N = normalize(pin.w_normal.xyz);
  float3 L = normalize(-light_direction.xyz);

  float3 color = max(0, dot(N, L)) * basecolor.rgb + emmisive;
  return float4(color, basecolor.a);
}
```

3．ビルド・実行し正しく描画されていることを確認する
※モデルデータを「DamagedHelmet」に変更する

【評価項目】
□テクスチャの取得
□テクスチャの描画