UNIT10: ADVANCED SPRITE

【学習要項】
□Completion

【演習手順】
1．テクスチャのロードをモジュール化する(texture.h, texture.cpp)
※sprite, sprite_batch クラスのコンストラクタで、テクスチャロードのコードをこの関数を使ったものに書き換える

```
#include <WICTextureLoader.h>
using namespace DirectX;

#include <wrl.h>
using namespace Microsoft::WRL;

#include <string>
#include <map>
using namespace std;

static map<wstring, ComPtr<ID3D11ShaderResourceView>> resources;
HRESULT load_texture_from_file(ID3D11Device* device, const wchar_t* filename,
    ID3D11ShaderResourceView** shader_resource_view, D3D11_TEXTURE2D_DESC* texture2d_desc)
{
    HRESULT hr{ S_OK };
    ComPtr<ID3D11Resource> resource;

    auto it = resources.find(filename);
    if (it != resources.end())
    {
        *shader_resource_view = it->second.Get();
        (*shader_resource_view)->AddRef();
        (*shader_resource_view)->GetResource(resource.GetAddressOf());
    }
    else
    {
        hr = CreateWICTextureFromFile(device, filename, resource.GetAddressOf(), shader_resource_view);
        _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
        resources.insert(make_pair(filename, *shader_resource_view));
    }

    ComPtr<ID3D11Texture2D> texture2d;
    hr = resource.Get()->QueryInterface<ID3D11Texture2D>(texture2d.GetAddressOf());
    _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
    texture2d->GetDesc(texture2d_desc);

    return hr;
}
void release_all_textures()
{
    resources.clear();
}
```

2．シェーダーファイルのロードをモジュール化する(shader.h, shader.cpp)
※sprite, sprite_batch クラスのコンストラクタで、シェーダーファイルロードのコードをこの関数を使ったものに書き換える
※必要に応じてヘッダファイルをインクルードする

```
HRESULT create_vs_from_cso(ID3D11Device* device, const char* cso_name, ID3D11VertexShader** vertex_shader,
  ID3D11InputLayout** input_layout, D3D11_INPUT_ELEMENT_DESC* input_element_desc, UINT num_elements)
{
  FILE* fp{ nullptr };
  fopen_s(&fp, cso_name, "rb");
  _ASSERT_EXPR_A(fp, "CSO File not found");

  fseek(fp, 0, SEEK_END);
  long cso_sz{ ftell(fp) };
  fseek(fp, 0, SEEK_SET);

  unique_ptr<unsigned char[]> cso_data{ make_unique<unsigned char[]>(cso_sz) };
```

```
    fread(cso_data.get(), cso_sz, 1, fp);
    fclose(fp);

    HRESULT hr{ S_OK };
    hr = device->CreateVertexShader(cso_data.get(), cso_sz, nullptr, vertex_shader);
    _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));

    if (input_layout)
    {
      hr = device->CreateInputLayout(input_element_desc, num_elements,
        cso_data.get(), cso_sz, input_layout);
      _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
    }

    return hr;
}

HRESULT create_ps_from_cso(ID3D11Device* device, const char* cso_name, ID3D11PixelShader** pixel_shader)
{
    FILE* fp{ nullptr };
    fopen_s(&fp, cso_name, "rb");
    _ASSERT_EXPR_A(fp, "CSO File not found");

    fseek(fp, 0, SEEK_END);
    long cso_sz{ ftell(fp) };
    fseek(fp, 0, SEEK_SET);

    unique_ptr<unsigned char[]> cso_data{ make_unique<unsigned char[]>(cso_sz) };
    fread(cso_data.get(), cso_sz, 1, fp);
    fclose(fp);

    HRESULT hr{ S_OK };
    hr = device->CreatePixelShader(cso_data.get(), cso_sz, nullptr, pixel_shader);
    _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));

    return hr;
}
```

3．COM オブジェクトを ComPtr スマートポインターテンプレートを使った変数宣言に変更する

①必要なヘッダーファイル

```
#include <wrl.h>
```

②変数宣言(例)

```
ID3D11Device* device;    ->    Microsoft::WRL::ComPtr<ID3D11Device> device;
```

③関数の引数に渡す場合(例)

```
device          ->    device.Get()
&device         ->    device.GetAddressOf()
```

④メンバ関数の呼び出しは同じ
⑤Release メンバ関数の呼び出しは不要

4．sprite, sprite_batch オブジェクト等をスマートポインターを使った変数宣言に変更する

5．sprite, sprite_batch クラスの render メンバ関数をオーバーロードする
※画面上の描画位置とサイズの指定のみでテクスチャ全体を描画する

```
void render(ID3D11DeviceContext* immediate_context, float dx, float dy, float dw, float dh);
```

6．sprite_batch オブジェクト描画時、別途ロードしたピクセルシェーダーに差替える仕組みを考える

7．sprite_batch オブジェクト描画時、別途ロードしたシェーダリソースビュー(テクスチャ)に差替える仕組みを考える

８．sprite クラスにフォント画像ファイルを使用し任意の文字列を画面に出力する機能(textout)を追加する
※フォント画像ファイルはアスキーコード順に１６ｘ１６の文字が配置された画像ファイル
※フォント画像ファイルは fonts フォルダのものを使用する

```
 1:  void sprite::textout(ID3D11DeviceContext* immediate_context, std::string s,
 2:      float x, float y, float w, float h, float r, float g, float b, float a)
 3:  {
 4:      float sw = static_cast<float>(texture2d_desc.Width / 16);
 5:      float sh = static_cast<float>(texture2d_desc.Height / 16);
 6:      float carriage = 0;
 7:      for (const char c : s)
 8:      {
 9:          render(immediate_context, x + carriage, y, w, h, r, g, b, a, 0,
10:              sw * (c & 0x0F), sh * (c >> 4), sw, sh);
11:          carriage += w;
12:      }
13:  }
```

９．Release モードでビルド・実行できることを確認する

１０．２Ｄゲームを開発することでプログラムの品質・性能向上を目指す

【評価項目】
□スプライトクラスの改良
□２Ｄゲームの作成