UNIT14: STATIC MESH - TEXTURE

【学習要項】
□MTL File format
□MTL File parser

【演習手順】
1．static_mesh オブジェクト生成のコンストラクタ引数を.¥¥resources¥¥Cup¥¥cup.obj に変更する
2．static_mesh クラスをテクスチャが扱えるように拡張する

① static_mesh クラスにメンバ変数を追加する

```
std::wstring texture_filename;
Microsoft::WRL::ComPtr<ID3D11ShaderResourceView> shader_resource_view;
```

②頂点構造体(vertex)にテクスチャ座標を追加する
※インプットレイアウトの生成で D3D11_INPUT_ELEMENT_DESC 構造体変数の値を変更すること

```
1:  struct vertex
 2:  {
 3:      DirectX::XMFLOAT3 position;
 4:      DirectX::XMFLOAT3 normal;
*5:      DirectX::XMFLOAT2 texcoord;
 6:  };
```

③ コンストラクタの OBJ ファイルパーサー部でテクスチャ座標とマテリアルファイル名を取得する

```
  1:  static_mesh::static_mesh(ID3D11Device* device, const wchar_t* obj_filename)
  2:  {
  3:    std::vector<vertex> vertices;
  4:    std::vector<uint32_t> indices;
  5:    uint32_t current_index{ 0 };
  6:
  7:    std::vector<XMFLOAT3> positions;
  8:    std::vector<XMFLOAT3> normals;
* 9:    std::vector<XMFLOAT2> texcoords;
*10:    std::vector<wstring> mtl_filenames;
 11:
 12:    std::wifstream fin(obj_filename);
 13:    _ASSERT_EXPR(fin, L"'OBJ file not found.");
 14:    wchar_t command[256];
 15:    while (fin)
 16:    {
 17:      fin >> command;
 18:        :
 19:        : 省略
 20:        :
*21:      else if (0 == wcscmp(command, L"vt"))
*22:      {
*23:        float u, v;
*24:        fin >> u >> v;
*25:        texcoords.push_back({ u, v });
*26:        fin.ignore(1024, L'¥n');
*27:      }
 28:      else if (0 == wcscmp(command, L"f"))
 29:      {
 30:        for (size_t i = 0; i < 3; i++)
 31:        {
 32:          vertex vertex;
 33:          size_t v, vt, vn;
 34:
 35:          fin >> v;
 36:          vertex.position = positions.at(v - 1);
 37:          if (L'/' == fin.peek())
 38:          {
 39:            fin.ignore(1);
 40:            if (L'/' != fin.peek())
 41:            {
```

```
 42:             fin >> vt;
*43:             vertex.texcoord = texcoords.at(vt - 1);
 44:           }
 45:           if (L'/' == fin.peek())
 46:           {
 47:             fin.ignore(1);
 48:             fin >> vn;
 49:             vertex.normal = normals.at(vn - 1);
 50:           }
 51:         }
 52:         vertices.push_back(vertex);
 53:         indices.push_back(current_index++);
 54:       }
 55:       fin.ignore(1024, L'¥n');
 56:     }
*57:     else if (0 == wcscmp(command, L"mtllib"))
*58:     {
*59:       wchar_t mtllib[256];
*60:       fin >> mtllib;
*61:       mtl_filenames.push_back(mtllib);
*62:     }
 63:     else
 64:     {
 65:       fin.ignore(1024, L'¥n');
 66:     }
 67:   }
 68:   fin.close();
 69:       :
 70:       : 省略（この場所に④MTL ファイルパーサー部の実装を挿入する）
 71:       :
 72: }
```

④ MTL ファイルパーサー部の実装（MTL ファイルをロードし、テクスチャファイル名を取得）

※MTL ファイルの仕様書（Mtl Specification.txt）を参照する
※filesystem ヘッダファイルをインクルードする
※filesystem クラスは C++17 以上で使用可能である
※プロジェクトのプロパティの「C/C++」→「言語」→「C++言語標準」で ISO C++17 Standard に変更する

```
 70: std::filesystem::path mtl_filename(obj_filename);
 71: mtl_filename.replace_filename(std::filesystem::path(mtl_filenames[0]).filename());
 72:
 73: fin.open(mtl_filename);
 74: _ASSERT_EXPR(fin, L"'MTL file not found.");
 75:
 76: while (fin)
 77: {
 78:     fin >> command;
 79:     if (0 == wcscmp(command, L"map_Kd"))
 80:     {
 81:       fin.ignore();
 82:       wchar_t map_Kd[256];
 83:       fin >> map_Kd;
 84:
 85:       std::filesystem::path path(obj_filename);
 86:       path.replace_filename(std::filesystem::path(map_Kd).filename());
 87:       texture_filename = path;
 88:       fin.ignore(1024, L'¥n');
 89:     }
 90:     else
 91:     {
 92:       fin.ignore(1024, L'¥n');
 93:     }
 94: }
 95: fin.close();
```

⑤テクスチャのロード、シェーダーリソースビューオブジェクトの生成をおこなう
※UNIT.10 に掲載した load_texture_from_file 関数を使用した実装例

```
1:  D3D11_TEXTURE2D_DESC texture2d_desc{};
2:  load_texture_from_file(device, texture_filename.c_str(),
3:    shader_resource_view.GetAddressOf(), &texture2d_desc);
```

⑥ static_mesh クラスの render メンバ関数で生成したシェーダーリソースビューをピクセルシェーダーにバインドする
※スロット番号は 0 にすること

3．テクスチャを扱えるようにシェーダーを変更する
① static_mesh.hlsli の VS_OUT を変更する

```
1:  struct VS_OUT
2:  {
3:      float4 position : SV_POSITION;
4:      float4 color : COLOR;
*5:     float2 texcoord : TEXCOORD;
6:  };
```

② static_mesh_vs.hlsl の変更

```
* 1:  VS_OUT main(float4 position : POSITION, float4 normal : NORMAL, float2 texcoord : TEXCOORD)
  2:  {
  3:      VS_OUT vout;
  4:      vout.position = mul(position, mul(world, view_projection));
  5:
  6:      normal.w = 0;
  7:      float4 N = normalize(mul(normal, world));
  8:      float4 L = normalize(-light_direction);
  9:
 10:      vout.color.rgb = material_color.rgb * max(0, dot(L, N));
 11:      vout.color.a = material_color.a;
 12:
*13:      vout.texcoord = texcoord;
 14:
 15:      return vout;
 16:  }
```

③ static_mesh_ps.hlsl の変更
※使用するサンプラーステートを変更し、効果を確認する

```
1:  Texture2D color_map : register(t0);
2:  SamplerState point_sampler_state : register(s0);
3:  SamplerState linear_sampler_state : register(s1);
4:  SamplerState anisotropic_sampler_state : register(s2);
5:
6:  float4 main(VS_OUT pin) : SV_TARGET
7:  {
8:    return color_map.Sample(anisotropic_sampler_state, pin.texcoord) * pin.color;
9:  }
```

4．描画に成功したら、.¥¥resources¥¥Bison¥¥Bison.obj をロードしテストする
※テクスチャが正しく貼られていないことを確認する
①UV 座標系は、OBJ ファイルを書き出した DCC ツールによって、原点の場所・V 軸の向きが違う場合がある
②V 座標値を変換（上下反転）する

```
texcoords.push_back({ u, 1.0f - v }); //2．③ 25:行目
```

③フラグ変数を導入し変換の有無を制御する

【評価項目】
□MTL ファイルのロード・テクスチャマッピング描画
□テクスチャ座標系の理解