

## UNIT01:INITIALIZATION

---

### 【学習要項】

- ☐Direct3D 11
- ☐COM (Component Object Model)
- ☐Device and Device context
- ☐Textures and Data Resource Formats
- ☐The Swap Chain and Page Flipping
- ☐Depth Buffering
- ☐Texture Resource Views
- ☐Multisampling Theory
- ☐Feature Levels
- ☐Viewport
- ☐Assertions

### 【演習手順】

1. ファイルサーバーより x3dgp.00 をコピーする
2. 3dgp.sln を開く
3. framework.h を開き、先頭に下記のインクルード文を入力する

```
#include <d3d11.h>
```

4. framework クラスのメンバ変数として以下の COM オブジェクトのポインタ変数を定義する

```
ID3D11Device* device;  
ID3D11DeviceContext* immediate_context;  
IDXGISwapChain* swap_chain;  
ID3D11RenderTargetView* render_target_view;  
ID3D11DepthStencilView* depth_stencil_view;
```

5. framework.cpp を開く
6. framework クラスの initialize メンバ関数で以下の処理を実装する

①デバイス・デバイスコンテキスト・スワップチェーンの作成

```
1:  HRESULT hr{ S_OK };  
2:  
3:  UINT create_device_flags{ 0 };  
4:  #ifdef _DEBUG  
5:      create_device_flags |= D3D11_CREATE_DEVICE_DEBUG;  
6:  #endif  
7:  
8:  D3D_FEATURE_LEVEL feature_levels{ D3D_FEATURE_LEVEL_11_0 };  
9:  
10:  DXGI_SWAP_CHAIN_DESC swap_chain_desc{};  
11:  swap_chain_desc.BufferCount = 1;  
12:  swap_chain_desc.BufferDesc.Width = SCREEN_WIDTH;  
13:  swap_chain_desc.BufferDesc.Height = SCREEN_HEIGHT;  
14:  swap_chain_desc.BufferDesc.Format = DXGI_FORMAT_R8G8B8A8_UNORM;  
15:  swap_chain_desc.BufferDesc.RefreshRate.Numerator = 60;  
16:  swap_chain_desc.BufferDesc.RefreshRate.Denominator = 1;  
17:  swap_chain_desc.BufferUsage = DXGI_USAGE_RENDER_TARGET_OUTPUT;  
18:  swap_chain_desc.OutputWindow = hwnd;  
19:  swap_chain_desc.SampleDesc.Count = 1;  
20:  swap_chain_desc.SampleDesc.Quality = 0;  
21:  swap_chain_desc.Windowed = !FULLSCREEN;  
22:  hr = D3D11CreateDeviceAndSwapChain(NULL, D3D_DRIVER_TYPE_HARDWARE, NULL, create_device_flags,  
23:      &feature_levels, 1, D3D11_SDK_VERSION, &swap_chain_desc,  
24:      &swap_chain, &device, NULL, &immediate_context);  
25:  _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
```

②レンダーターゲットビューの作成

```
1:  ID3D11Texture2D* back_buffer{};  
2:  hr = swap_chain->GetBuffer(0, __uuidof(ID3D11Texture2D), reinterpret_cast<LPVOID*>(&back_buffer));  
3:  _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));  
4:  
5:  hr = device->CreateRenderTargetView(back_buffer, NULL, &render_target_view);
```

```
6: _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
7:
8: back_buffer->Release();
```

### ③深度ステンシルビューの作成

```
1: ID3D11Texture2D* depth_stencil_buffer{};
2: D3D11_TEXTURE2D_DESC texture2d_desc{};
3: texture2d_desc.Width = SCREEN_WIDTH;
4: texture2d_desc.Height = SCREEN_HEIGHT;
5: texture2d_desc.MipLevels = 1;
6: texture2d_desc.ArraySize = 1;
7: texture2d_desc.Format = DXGI_FORMAT_D24_UNORM_S8_UINT;
8: texture2d_desc.SampleDesc.Count = 1;
9: texture2d_desc.SampleDesc.Quality = 0;
10: texture2d_desc.Usage = D3D11_USAGE_DEFAULT;
11: texture2d_desc.BindFlags = D3D11_BIND_DEPTH_STENCIL;
12: texture2d_desc.CPUAccessFlags = 0;
13: texture2d_desc.MiscFlags = 0;
14: hr = device->CreateTexture2D(&texture2d_desc, NULL, &depth_stencil_buffer);
15: _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
16:
17: D3D11_DEPTH_STENCIL_VIEW_DESC depth_stencil_view_desc{};
18: depth_stencil_view_desc.Format = texture2d_desc.Format;
19: depth_stencil_view_desc.ViewDimension = D3D11_DSV_DIMENSION_TEXTURE2D;
20: depth_stencil_view_desc.Texture2D.MipSlice = 0;
21: hr = device->CreateDepthStencilView(depth_stencil_buffer, &depth_stencil_view_desc, &depth_stencil_view);
22: _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
23:
24: depth_stencil_buffer->Release();
```

### ④ビューポートの設定

```
1: D3D11_VIEWPORT viewport{};
2: viewport.TopLeftX = 0;
3: viewport.TopLeftY = 0;
4: viewport.Width = static_cast<float>(SCREEN_WIDTH);
5: viewport.Height = static_cast<float>(SCREEN_HEIGHT);
6: viewport.MinDepth = 0.0f;
7: viewport.MaxDepth = 1.0f;
8: immediate_context->RSSetViewports(1, &viewport);
```

## 7. framework クラスの render メンバ関数で以下の処理を実装する

※sync\_interval に 1 をセットすると 60FPS の固定フレームレートで動作する

※今後の課題は可変フレームレートでの動作を前提に行う

```
1: void framework::render(float elapsed_time/*Elapsed seconds from last frame*/)
2: {
*3:     HRESULT hr{ S_OK };
4:
*5:     FLOAT color[]{ 0.2f, 0.2f, 0.2f, 1.0f };
*6:     immediate_context->ClearRenderTargetView(render_target_view, color);
*7:     immediate_context->ClearDepthStencilView(depth_stencil_view,
*8:         D3D11_CLEAR_DEPTH | D3D11_CLEAR_STENCIL, 1.0f, 0);
*9:     immediate_context->OMSetRenderTargets(1, &render_target_view, depth_stencil_view);
10:
11:     #ifdef USE_IMGUI
12:         ImGui::Render();
13:         ImGui_ImplDX11_RenderDrawData(ImGui::GetDrawData());
14:     #endif
15:
*16:     UINT sync_interval{ 0 };
*17:     swap_chain->Present(sync_interval, 0);
18: }
```

## UNIT01:INITIALIZATION

---

8. 3dgp プロパティページの「リンカー」→「入力」→「追加の依存…」に d3d11.lib と dxgi.lib を追加
9. 実行し、画面が指定した色でクリアされていることを確認する
10. COM オブジェクトが正しく解放されていることを確認する

※framework クラスの `uninitialize` メンバ関数ですべての COM オブジェクトを解放する

※解放が正しく出来ていない場合は、出力ウィンドウに "D3D11 WARNING" が出力される

```
1: bool framework::uninitialize()
2: {
3:     device->Release();
4:     immediate_context->Release();
5:     swap_chain->Release();
6:     render_target_view->Release();
7:     depth_stencil_view->Release();
8: }
```

11. ウィンドウサイズ (SCREEN\_WIDTH, SCREEN\_HEIGHT) を変更してみる

12. フルスクリーンモード (FULLSCREEN) に切り替えてみる

※実行時「ALT」+「ENTER」でもスクリーンモードの切り替えが行える

※フルスクリーン時に ESC キーで終了するとクラッシュするので、`framework.h` の下記のマクロブロックを有効にする

```
*1: #if 1
2:     BOOL fullscreen{};
3:     swap_chain->GetFullscreenState(&fullscreen, 0);
4:     if (fullscreen)
5:     {
6:         swap_chain->SetFullscreenState(FALSE, 0);
7:     }
8: #endif
```

13. ImGui を有効にする

①3dgp プロパティページの「C/C++」→「プリプロセッサ」→「プリプロセッサの定義」に `USE_IMGUI` を追加

②実行画面上に ImGui のウィンドウが表示されていることを確認する

14. MSDN で COM オブジェクト、メソッド、関数、構造体などを調べる
15. コメントを書く

### 【評価項目】

- ☐ Direct3D の初期化処理
- ☐ COM オブジェクトの生成と解放
- ☐ バックバッファのクリア色の変更
- ☐ 固定フレームレートの設定
- ☐ ウィンドウサイズの変更
- ☐ フルスクリーンモードへの変更