UNIT03:SPRITE – LOCATION/DIMENSION

【学習要項】
☐Screen coordinates
☐Transformation from screen coordinates to NDC

【演習手順】
1．sprite クラスのコンストラクタで頂点バッファオブジェクトの生成方法を変更する
※GPU (読み取りのみ) と CPU (書き込みのみ) によるアクセスを可能にする

```
 1:  D3D11_BUFFER_DESC buffer_desc{};
 2:  buffer_desc.ByteWidth = sizeof(vertices);
*3:  buffer_desc.Usage = D3D11_USAGE_DYNAMIC;
 4:  buffer_desc.BindFlags = D3D11_BIND_VERTEX_BUFFER;
*5:  buffer_desc.CPUAccessFlags = D3D11_CPU_ACCESS_WRITE;
 6:  buffer_desc.MiscFlags = 0;
 7:  buffer_desc.StructureByteStride = 0;
 8:  D3D11_SUBRESOURCE_DATA subresource_data{};
 9:  subresource_data.pSysMem = vertices;
10:  subresource_data.SysMemPitch = 0;
11:  subresource_data.SysMemSlicePitch = 0;
12:  hr = device->CreateBuffer(&buffer_desc, &subresource_data, &vertex_buffer);
13:  _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
```

2．sprite クラスの render メンバ関数のインターフェイスを変更する

```
void sprite::render(ID3D11DeviceContext *,
  float dx, float dy,        // 矩形の左上の座標（スクリーン座標系）
  float dw, float dh,        // 矩形のサイズ（スクリーン座標系）
)
```

3．sprite クラスの render メンバ関数の実装を変更する
※下記の実装は render メンバ関数の先頭に記述する
①スクリーン（ビューポート）のサイズを取得する

```
D3D11_VIEWPORT viewport{};
UINT num_viewports{ 1 };
immediate_context->RSGetViewports(&num_viewports, &viewport);
```

②render メンバ関数の引数（dx, dy, dw, dh）から矩形の各頂点の位置（スクリーン座標系）を計算する

```
 1:  //  (x0, y0) *----* (x1, y1)
 2:  //           |   /|
 3:  //           |  / |
 4:  //           | /  |
 5:  //           |/   |
 6:  //  (x2, y2) *----* (x3, y3)
 7:
 8:  // left-top
 9:  float x0{ dx };
10:  float y0{ dy };
11:  // right-top
12:  float x1{ dx + dw };
13:  float y1{ dy };
14:  // left-bottom
15:  float x2{ dx };
16:  float y2{ dy + dh };
17:  // right-bottom
18:  float x3{ dx + dw };
19:  float y3{ dy + dh };
```

③スクリーン座標系から NDC への座標変換をおこなう

```
1:  x0 = 2.0f * x0 / viewport.Width - 1.0f;
2:  y0 = 1.0f - 2.0f * y0 / viewport.Height;
3:  x1 = 2.0f * x1 / viewport.Width - 1.0f;
4:  y1 = 1.0f - 2.0f * y1 / viewport.Height;
```

```
5:  x2 = 2.0f * x2 / viewport.Width - 1.0f;
6:  y2 = 1.0f - 2.0f * y2 / viewport.Height;
7:  x3 = 2.0f * x3 / viewport.Width - 1.0f;
8:  y3 = 1.0f - 2.0f * y3 / viewport.Height;
```

④計算結果で頂点バッファオブジェクトを更新する

```
 1:  HRESULT hr{ S_OK };
 2:  D3D11_MAPPED_SUBRESOURCE mapped_subresource{};
 3:  hr = immediate_context->Map(vertex_buffer, 0, D3D11_MAP_WRITE_DISCARD, 0, &mapped_subresource);
 4:  _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
 5:
 6:  vertex* vertices{ reinterpret_cast<vertex*>(mapped_subresource.pData) };
 7:  if (vertices != nullptr)
 8:  {
 9:    vertices[0].position = { x0, y0 , 0 };
10:    vertices[1].position = { x1, y1 , 0 };
11:    vertices[2].position = { x2, y2 , 0 };
12:    vertices[3].position = { x3, y3 , 0 };
13:    vertices[0].color = vertices[1].color = vertices[2].color = vertices[3].color = { 1, 1, 1, 1 };
14:  }
15:
16:  immediate_context->Unmap(vertex_buffer, 0);
```

4．framework クラスの render メンバ関数での sprite オブジェクトの描画方法を変更する
5．実行し、矩形を任意の位置・サイズで描画できることを確認する
※ピクセルシェーダーの main 関数を下記の通りに変更する

```
1:  #include "sprite.hlsli"
2:  float4 main(VS_OUT pin) : SV_TARGET
3:  {
4:      return pin.color;
5:  }
```

6．表示色を変更できるように sprite クラスの render メンバ関数のインターフェイスを変更し、実装を変更する

```
 1:  void sprite::render(ID3D11DeviceContext* immediate_context,
 2:      float dx, float dy, float dw, float dh,
*3:      float r, float g, float b, float a);
```

【評価項目】
□スクリーン座標系での、矩形の位置・サイズの制御
□矩形の表示色の制御