シェーダコード

2025年度 プログラムワークショップIV（2）

目次

- シェーダ入門
- 簡単なシェーダ
- シェーダを読む
- HLSL入門

目次

- シェーダ入門
- 簡単なシェーダ
- シェーダを読む
- HLSL入門

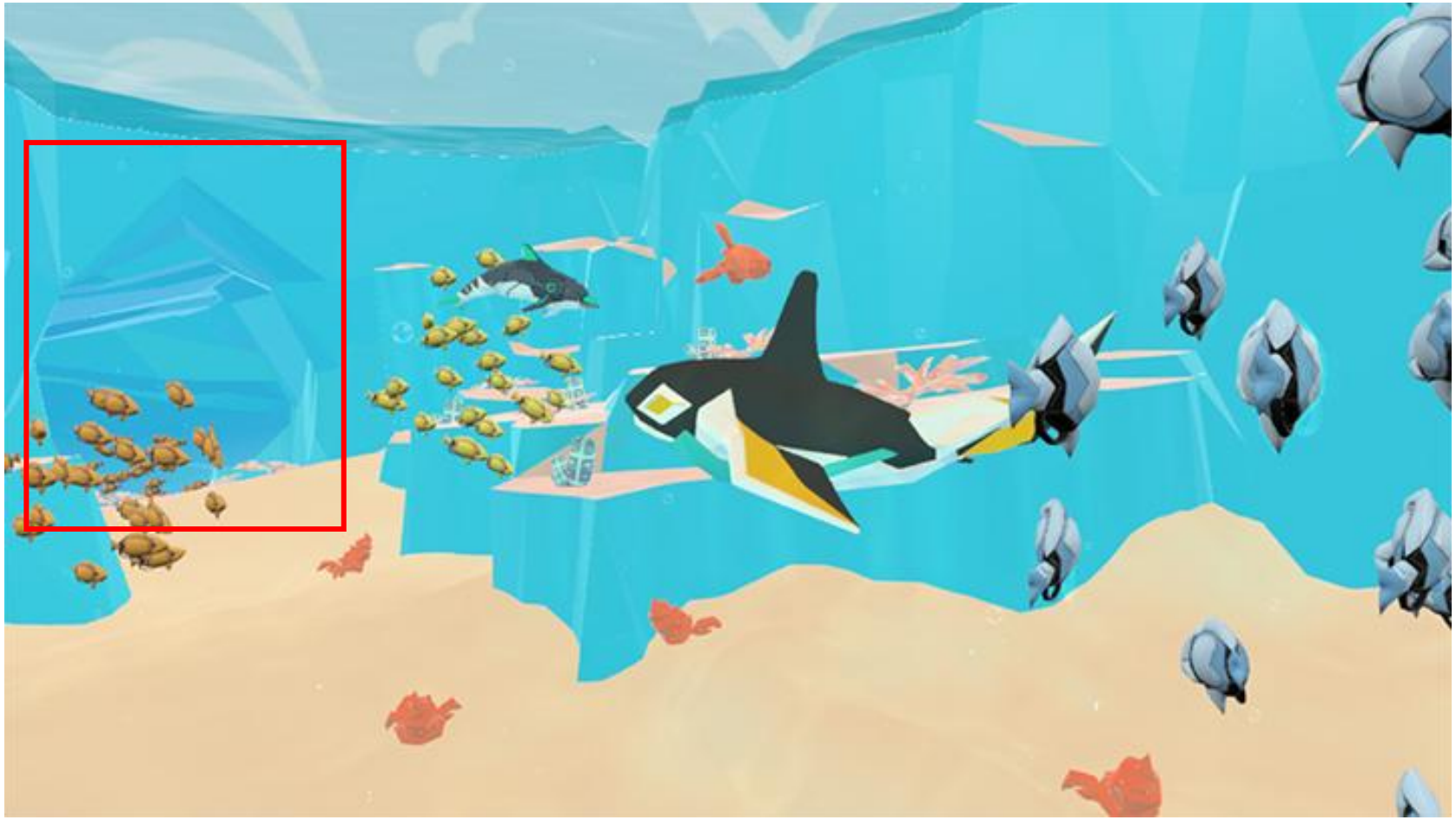
シェーダ

- シェーダー(英: shader)はグラフィックスパイプラインを構成する各ステージの挙動を記述したプログラムである。
- また狭義にはグラフィックスパイプライン中のシェーディング(陰影処理)に関する挙動を記述したプログラムを指す。

https://ja.wikipedia.org/wiki/シェーダー

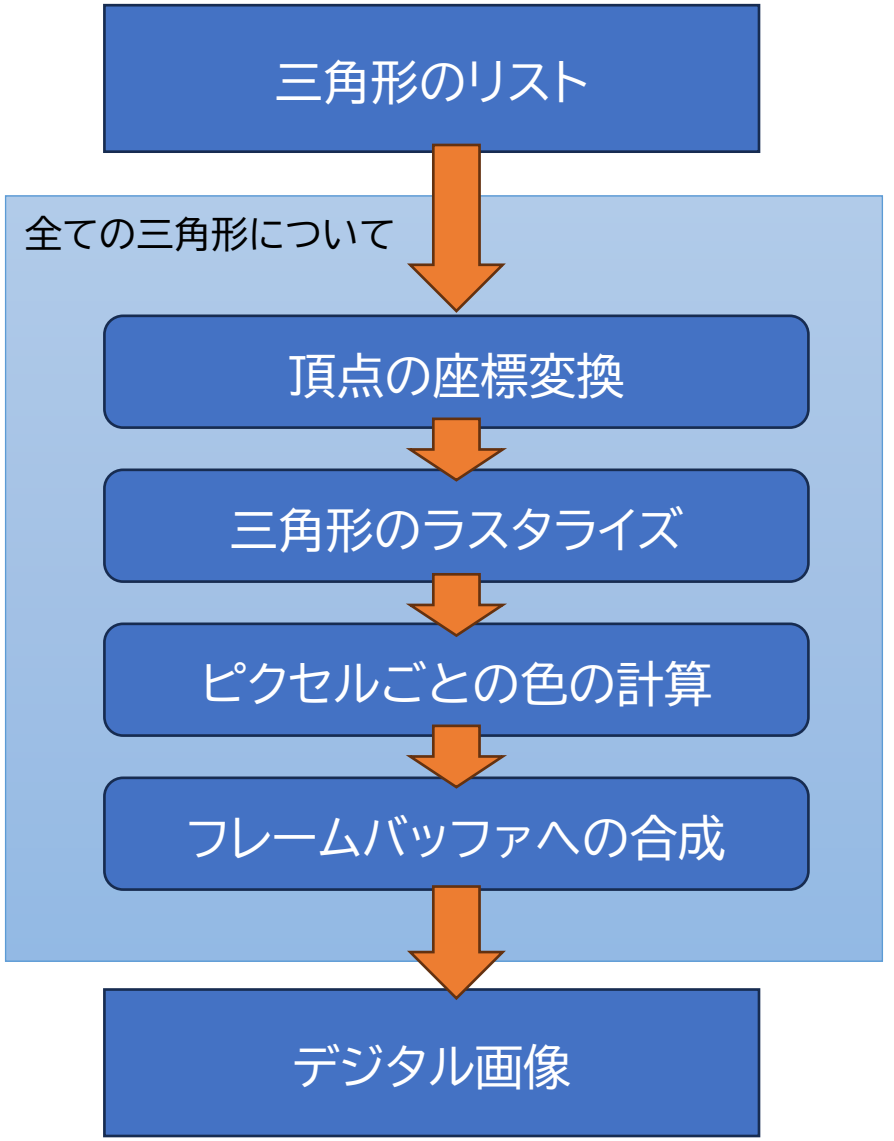
シェーダでできること

- 通路の先が暗くて先が見えなかったけれども、近づくと先が見えるようになる
  - CPUでも処理できるが、シェーダで実現した方が簡単・すなおに実現できる



描画処理

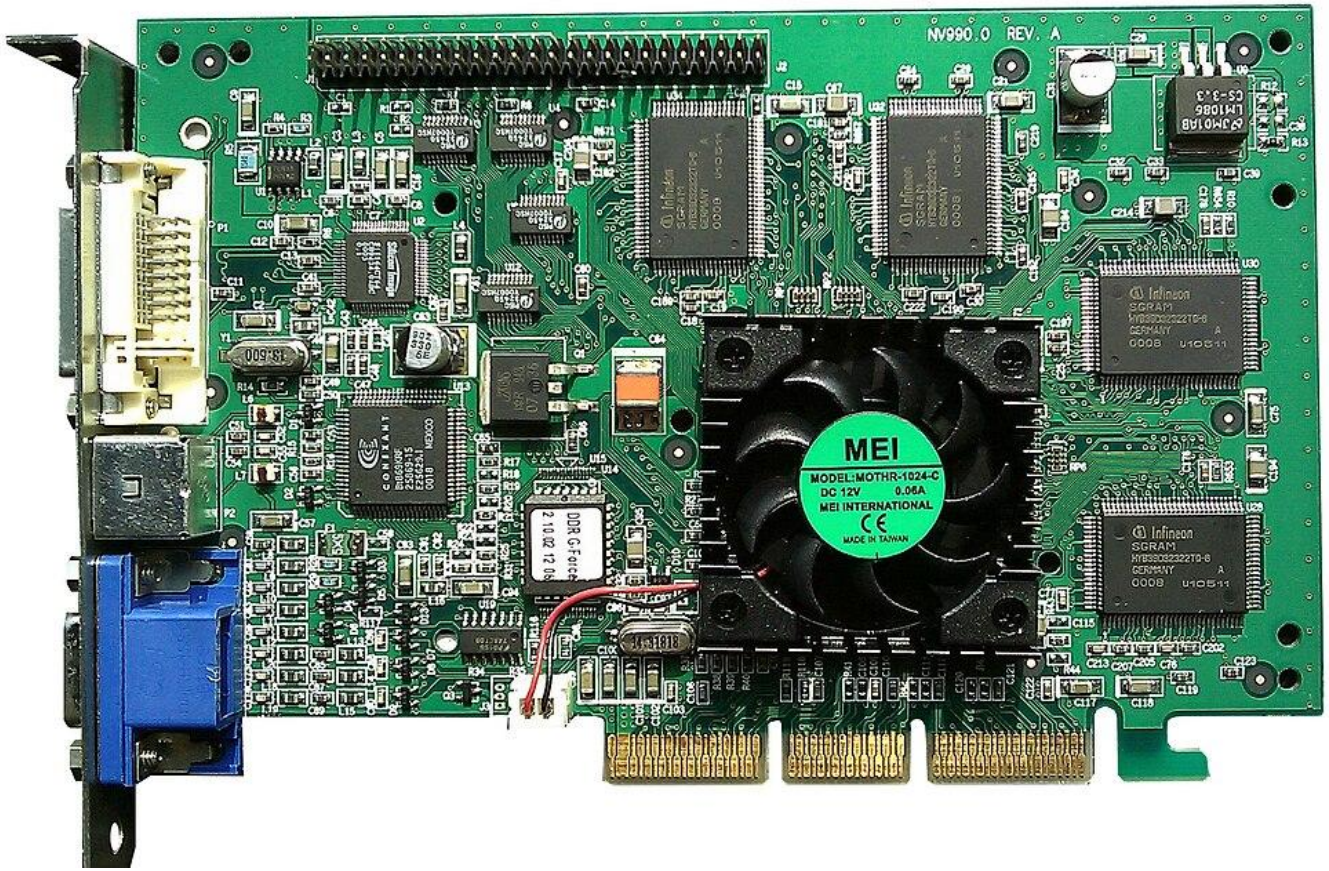
- モデルごとにポリゴン単位で描画



GPU: Graphics Processing Unit

- ハードウェア T&L（座標変換と光源処理）
  - CPU負荷の低減
  - 大量のポリゴン
- 初期は表示できる手法が限定的
  - テクスチャコンバイナー
    - 頂点の色を使う
    - テクスチャを張る
    - 頂点の色とテクスチャの色を掛ける
    - 鏡面反射

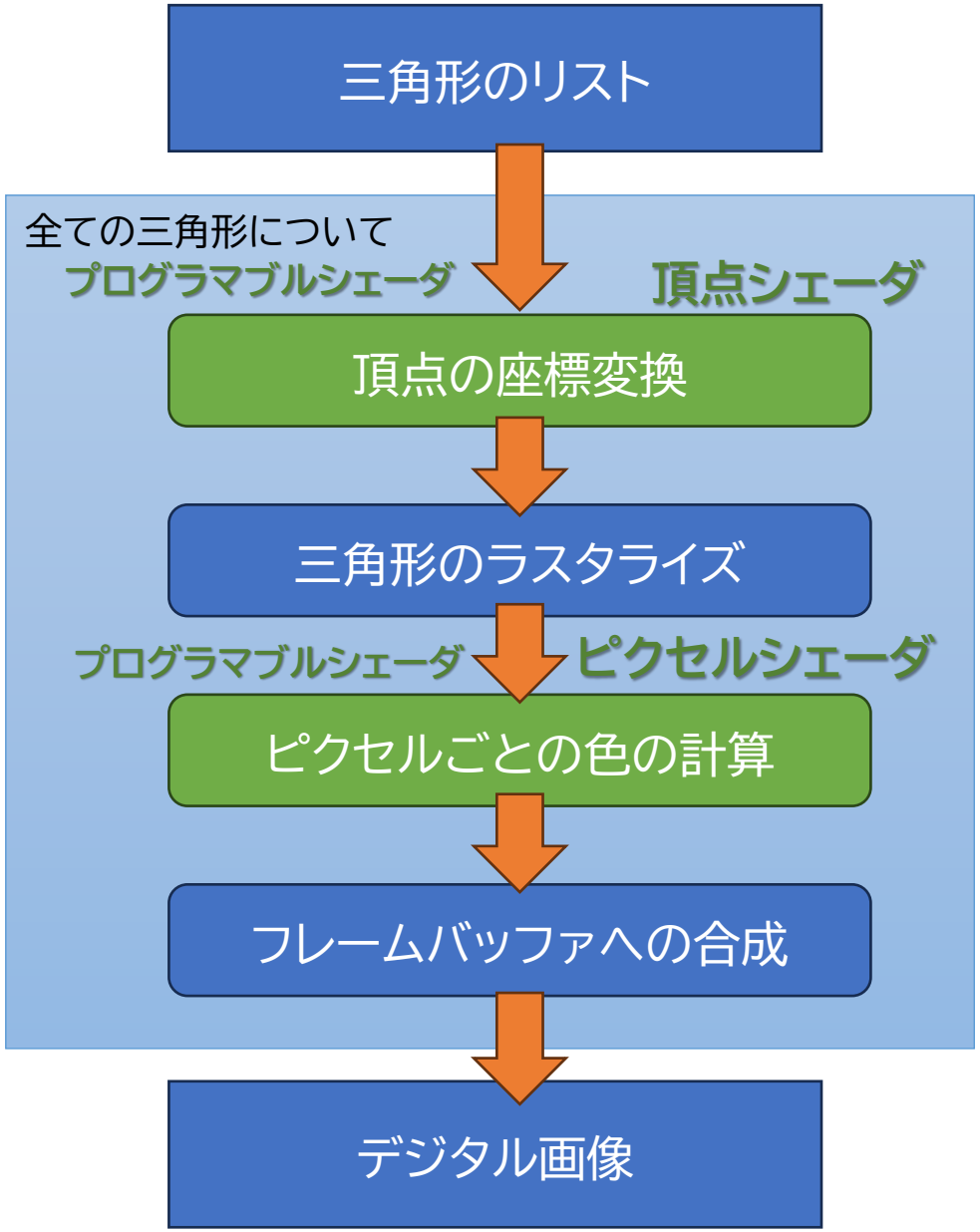
GeForce 256



https://ja.wikipedia.org/wiki/GeForce_256

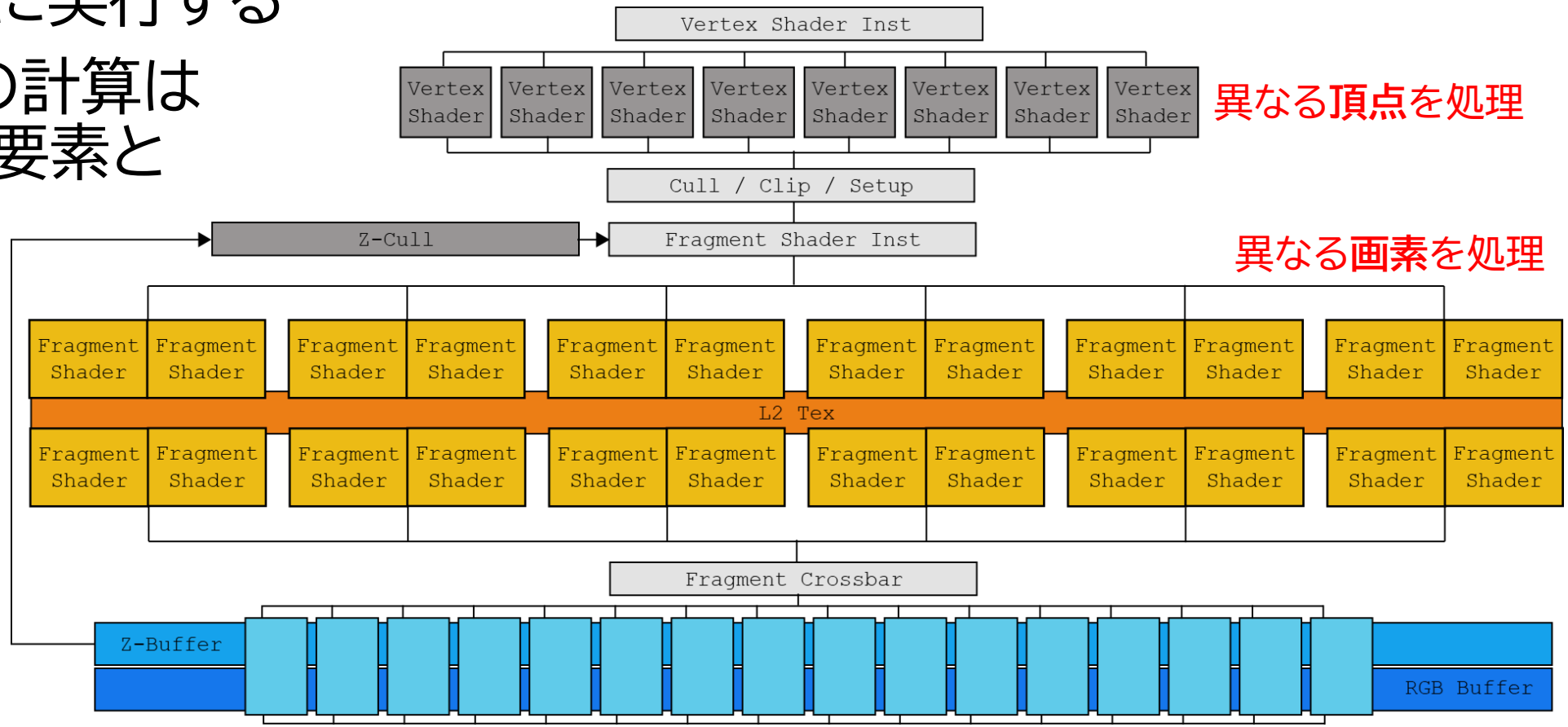
プログラマブルシェーダ

- 頂点処理と画素毎の陰影計算をプログラミングできるようにした
  - プログラミング言語でコードを書いて変更できるようにした



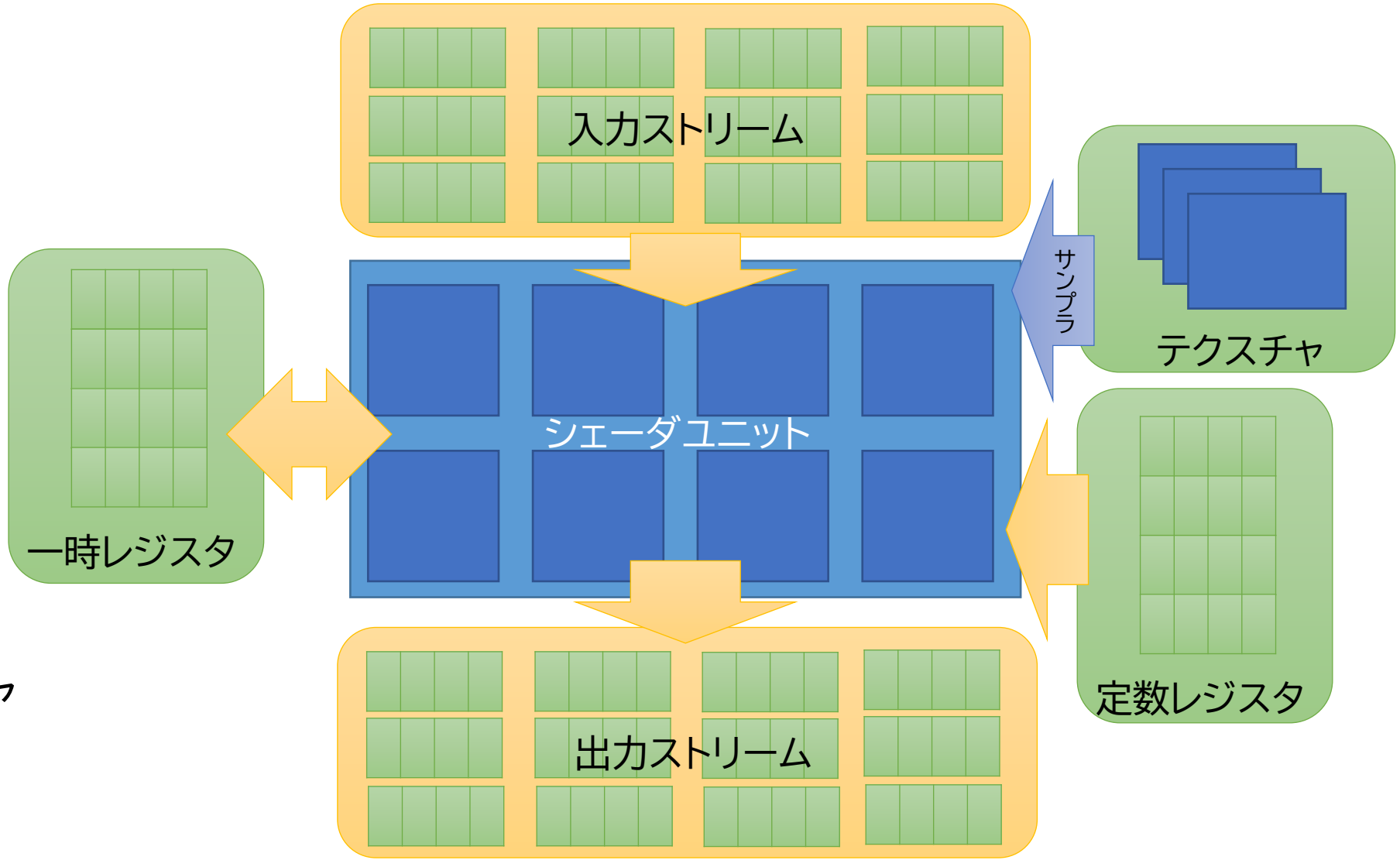
どうしてGPUが高速なのか？

- 処理を並列に実行する
- 頂点・画素の計算は(ほぼ)他の要素と関係がないので可能
- 可能な処理が少ない
  - 少ない命令数



シェーダユニット：論理的なシェーダの機構

- 考え方は、頂点シェーダもピクセルシェーダも同じ
  - 出力するのが頂点位置・色か、画素の色か
  - 昔は頂点処理ではテクスチャを読めなかったが現在は可能

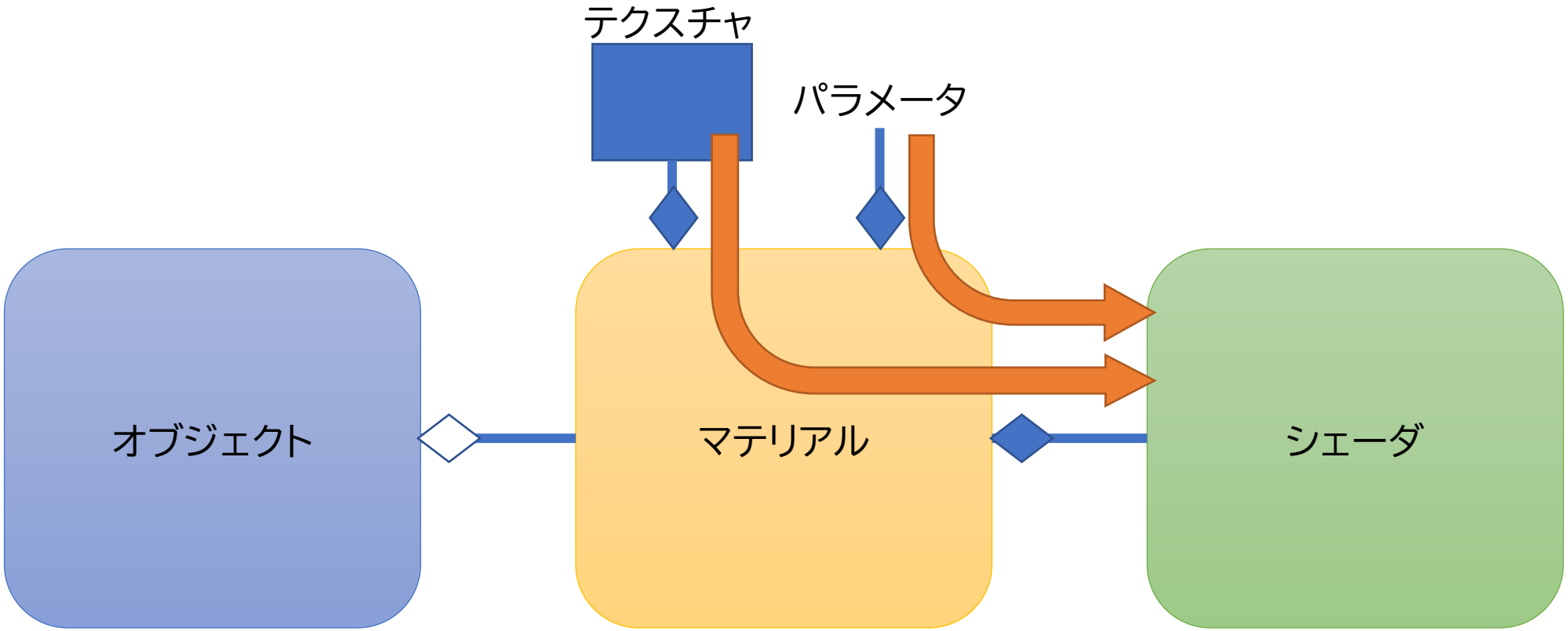


シェーダの歴史

- シェーダアセンブラ
  - 2001年ぐらいは直接書いていた…
- シェーディング言語
  - HLSL: High Level Shading Language
    - DirectX
  - GLSL: OpenGL Shading Language
    - OpenGL ARB
  - Cg
    - NVIDIA
  - Metal Shading Language
    - macOS
- ノードベース
  - シェーダグラフ（Unity）
  - ブループリント（Unreal Engine）

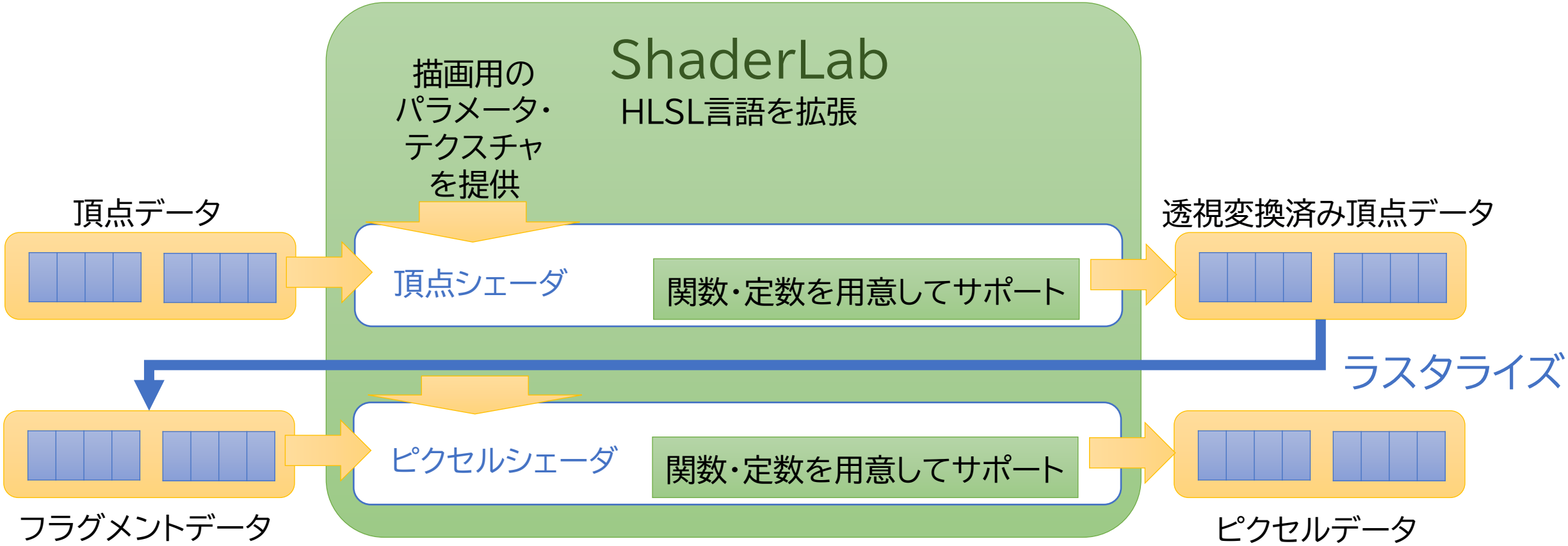
```
0001: ; blur.psh
0002: ;
0003: ;           0 0 0
0004: ; 周りの色を 0 1 1 の平均を取る
0005: ;           0 1 1
0006: ps.1.1
0007:
0008: def c0, 0.5f, 0.5f, 0.5f, 0.5f
0009:
0010: ; テクスチャーの色を引っ張ってくる
0011: tex t0      ; 0:0 0 0  1:0 0 0  2:0 0 0  3:0 0 0
0012: tex t1      ;   0 1 0    0 0 0    0 0 0    0 0 1
0013: tex t2      ;   0 0 0    0 1 0    0 0 1    0 0 0
0014: tex t3
0015:
0016: ; r0 = 0.5*(0.5*(t0+t1)+0.5*(t2+t3)) = (t0+t1+t2+t3)/4
0017: ; 色の平均を取ってくる
0018: lrp r0, c0, t0, t1
0019: lrp r1, c0, t2, t3
0020: lrp r0, c0, r0, r1
```

Unityでのシェーダ(エディタからみて)



同じシェーダプログラムでの
設定の違いの多様性を与える

Unityでのシェーダ（アーキテクチャとして）



Unityでのシェーディング言語

- HLSL言語ベース
  - メジャー
  - HLSLからGLSL, Metal への変換ツールあり
- ShaderLab
  - Unityでのシェーダのラップ
  - エディタとの連携
  - DirectX11を想定
  - DirectXでできることはできる
    - Compute shader
    - テッセレーション
- Shader Graph(今後、紹介)
  - ノードベースプログラミング

```
Shader "MyShader" {
    Properties {
        _MyTexture ("My Texture", 2D) = "white" { }
        // colors や vectors などの他のプロパティーもここに置けます
    }
    SubShader {
        // ここに
        // - サーフェスシェーダー　か
        // - 頂点シェーダーとフラグメントシェーダー　か
        // - 固定関数シェーダー　を置きます
    }
    SubShader {
        // 古いグラフィックスカードでも実行できるように
        // 簡単なバージョンの SubShader をここに置きます
    }
}
```

目次

- シェーダ入門
- 簡単なシェーダ
- シェーダを読む
- HLSL入門

これからの目的

- 簡単なシェーダを使ったオブジェクトの描画

ステップ

- GitHubリポジトリのクローン
- Unityを立ち上げる
- オブジェクトを作る
- シェーダを作る
- オブジェクトにシェーダを関連付ける
- テクスチャを貼る

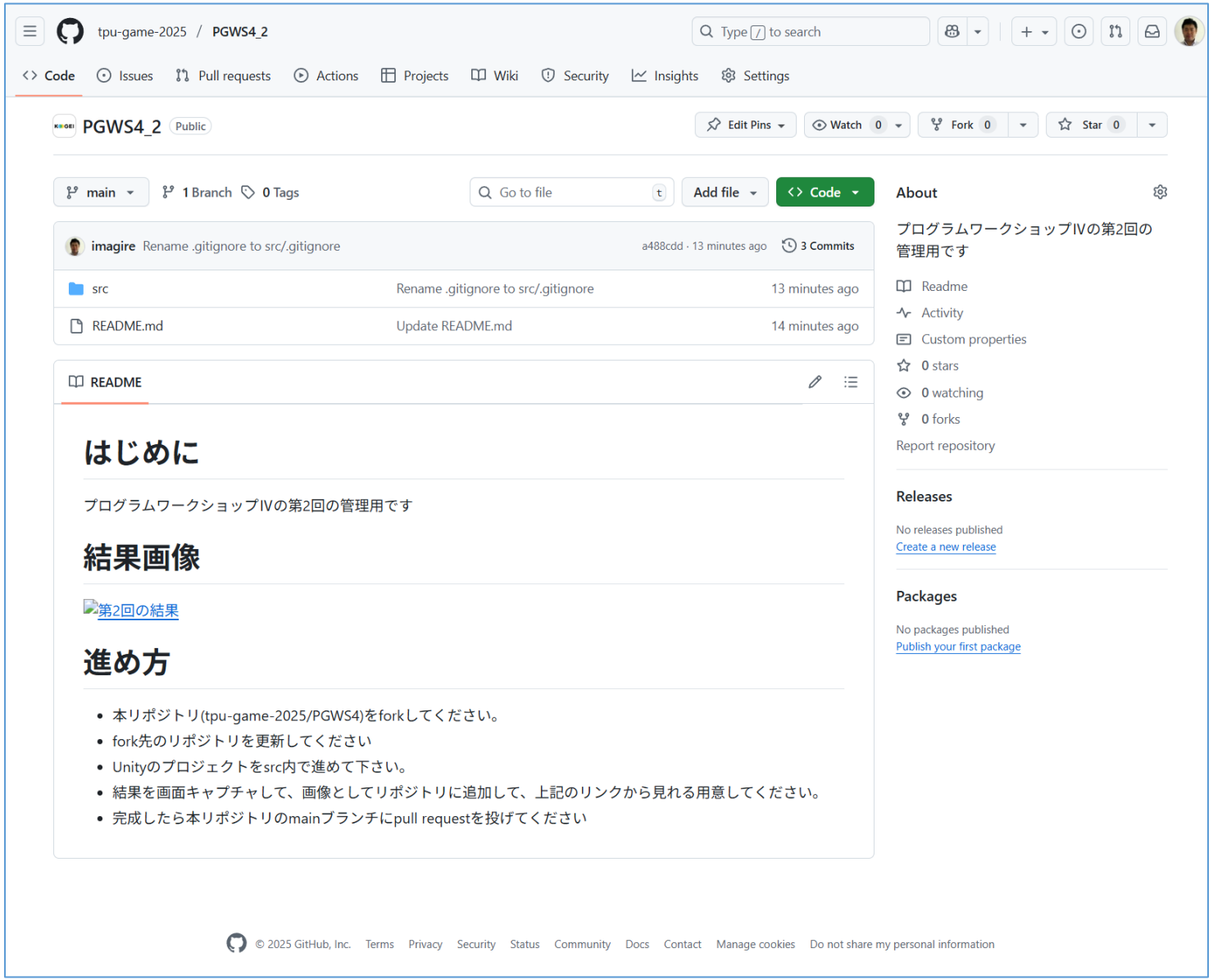
ステップ

- GitHubリポジトリのクローン
  - この項目は授業特有の内容になります
- Unityを立ち上げる
- オブジェクトを作る
- シェーダを作る
- オブジェクトにシェーダを関連付ける
- テクスチャを貼る

この頁は授業特有の内容になります

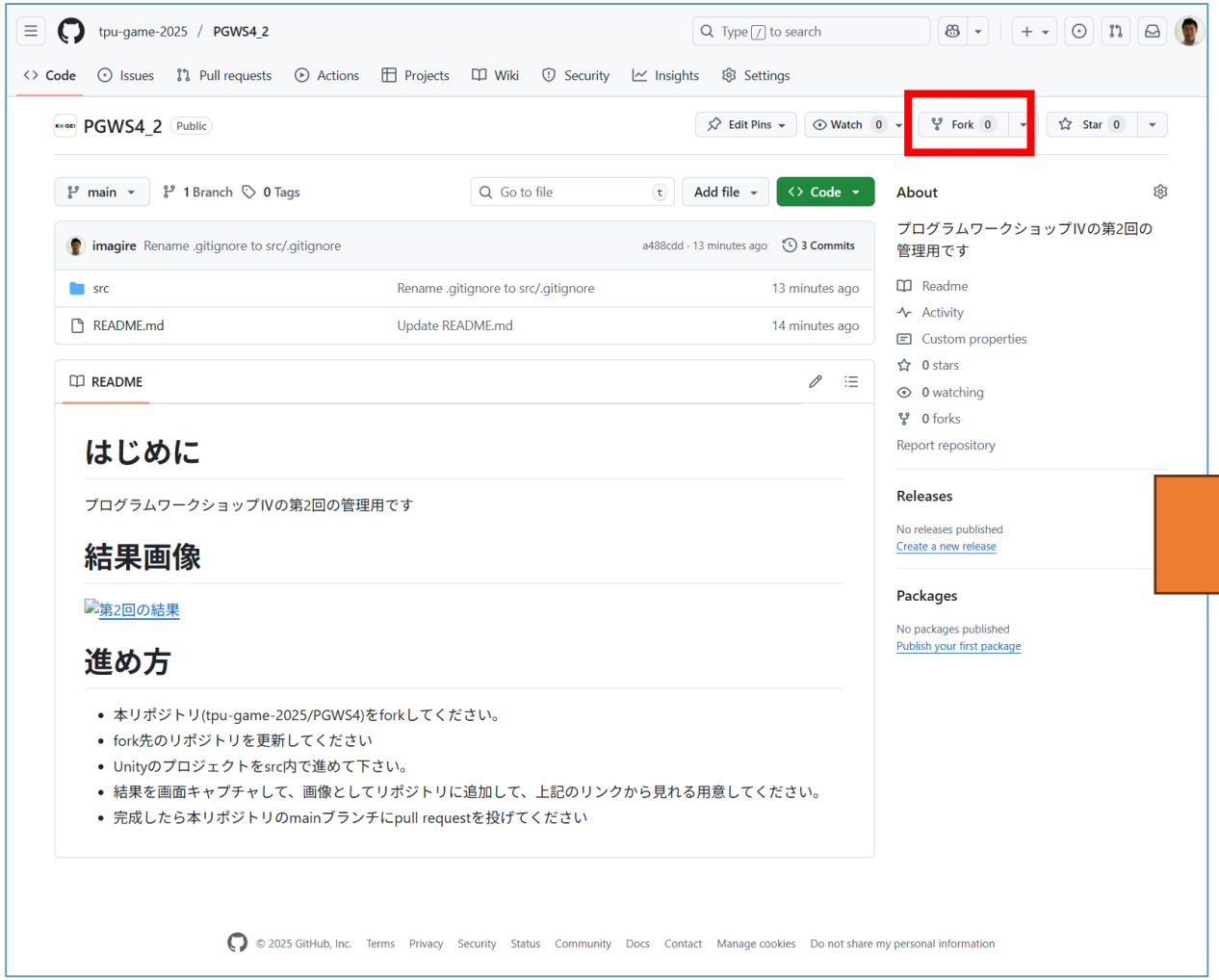
GitHubを開く

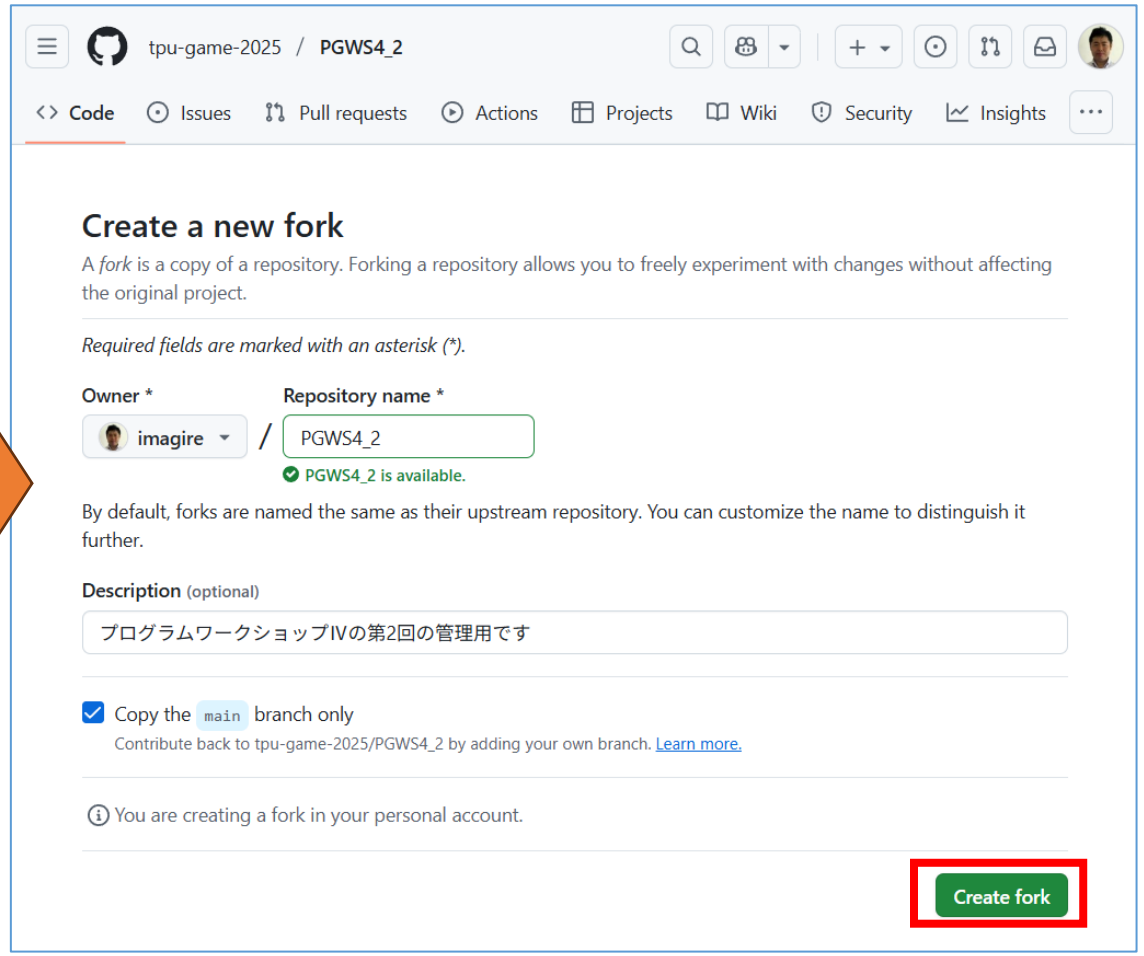
- https://github.com/tpu-game-2025/PGWS4_2/



この頁は授業特有の内容になります

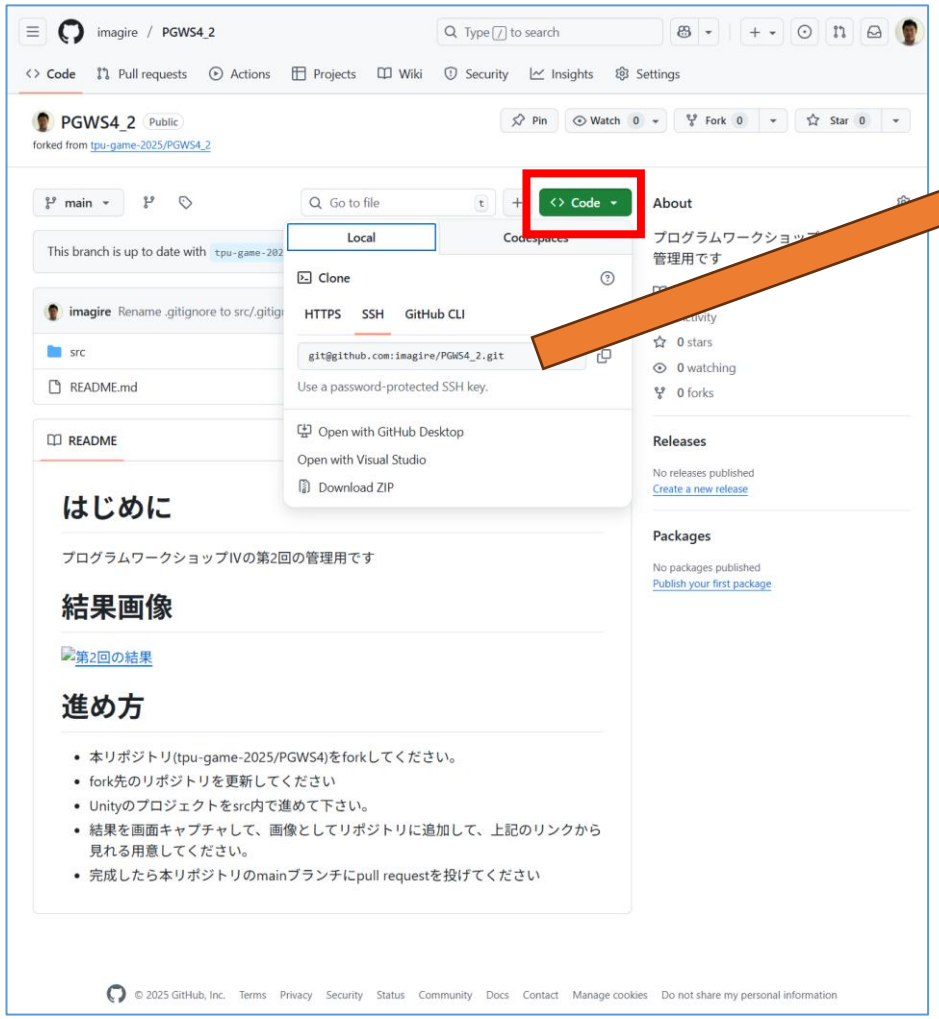
Fork

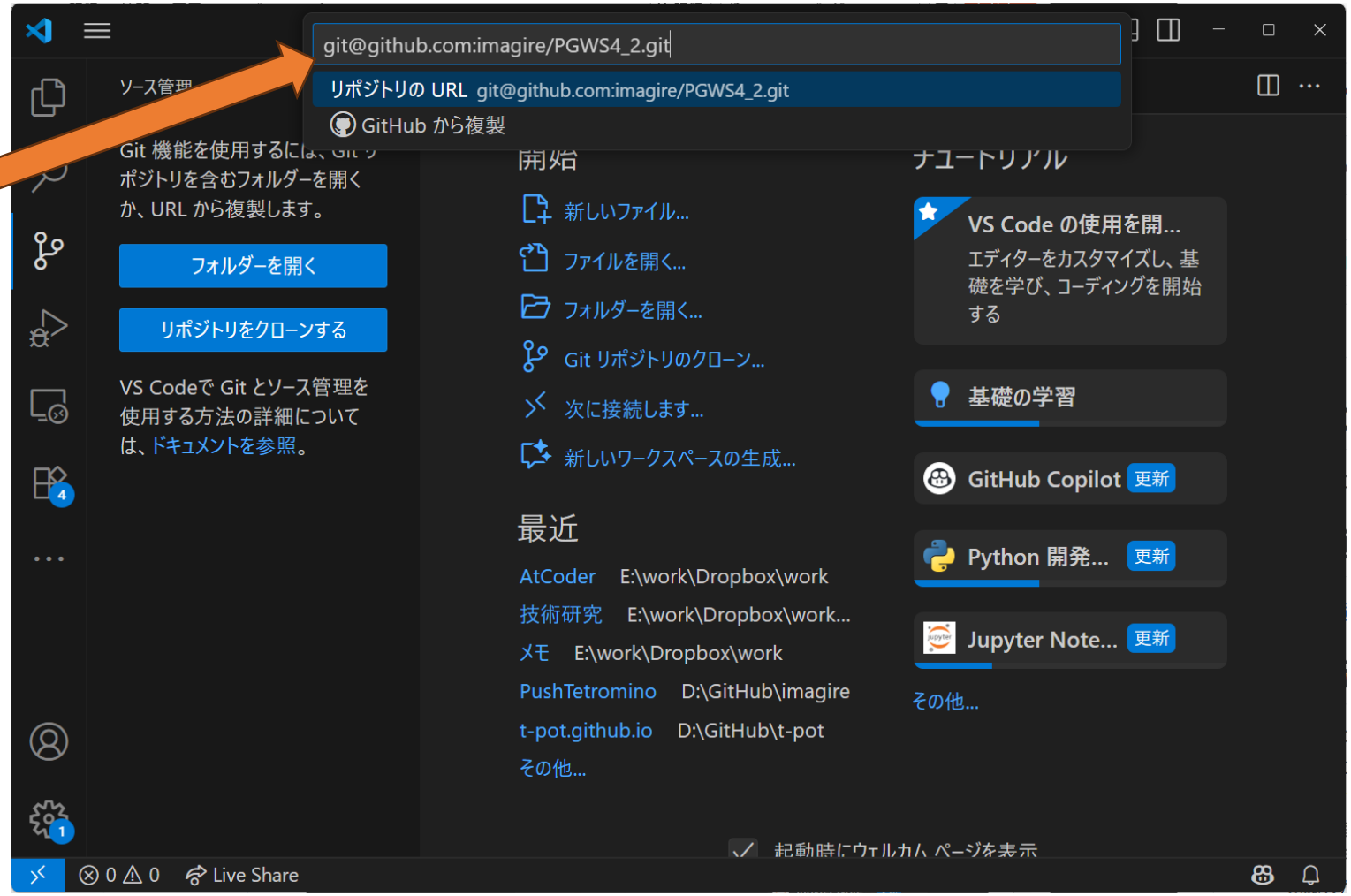




この頁は授業特有の内容になります

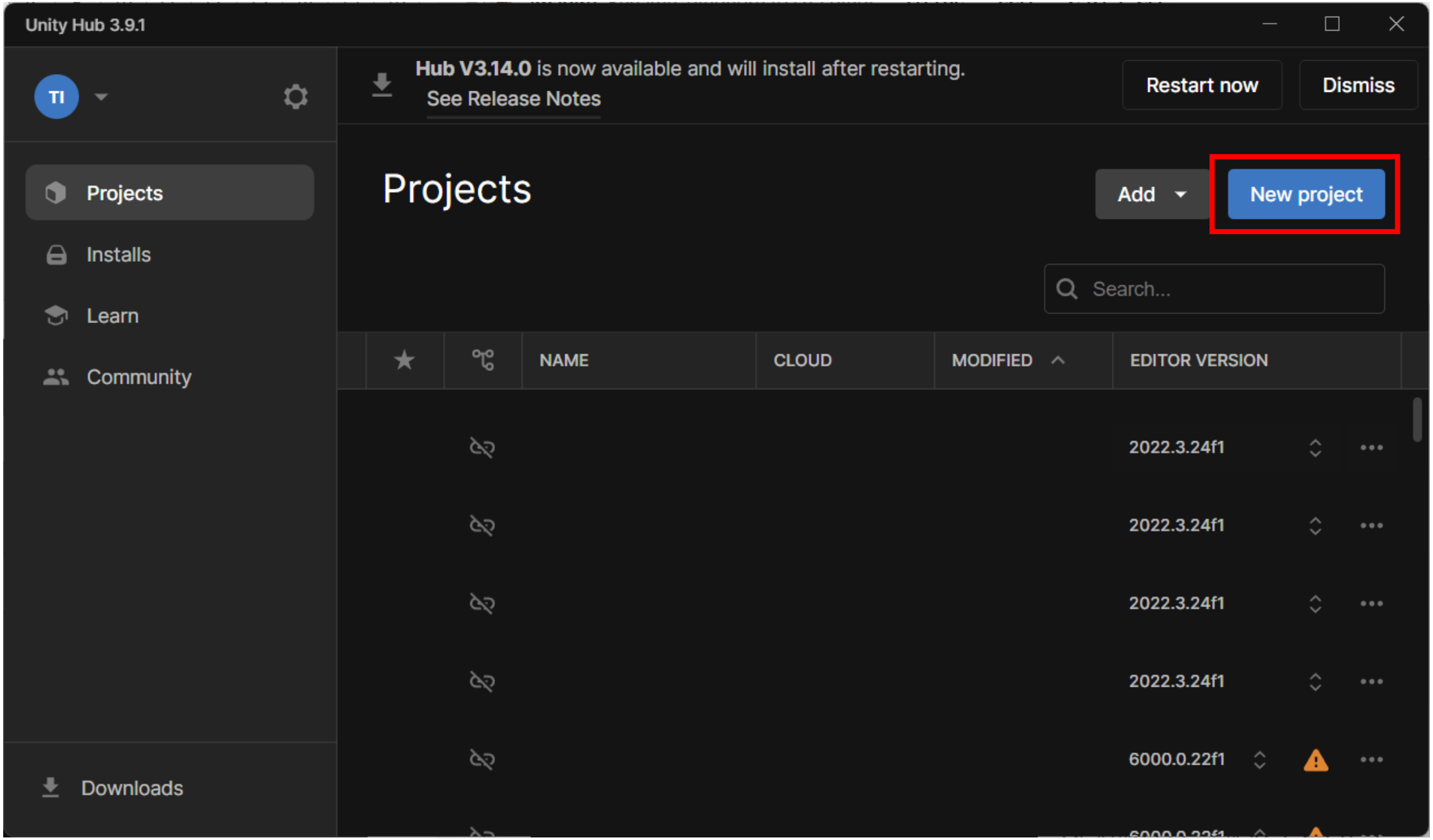
Clone：あなたの方法で

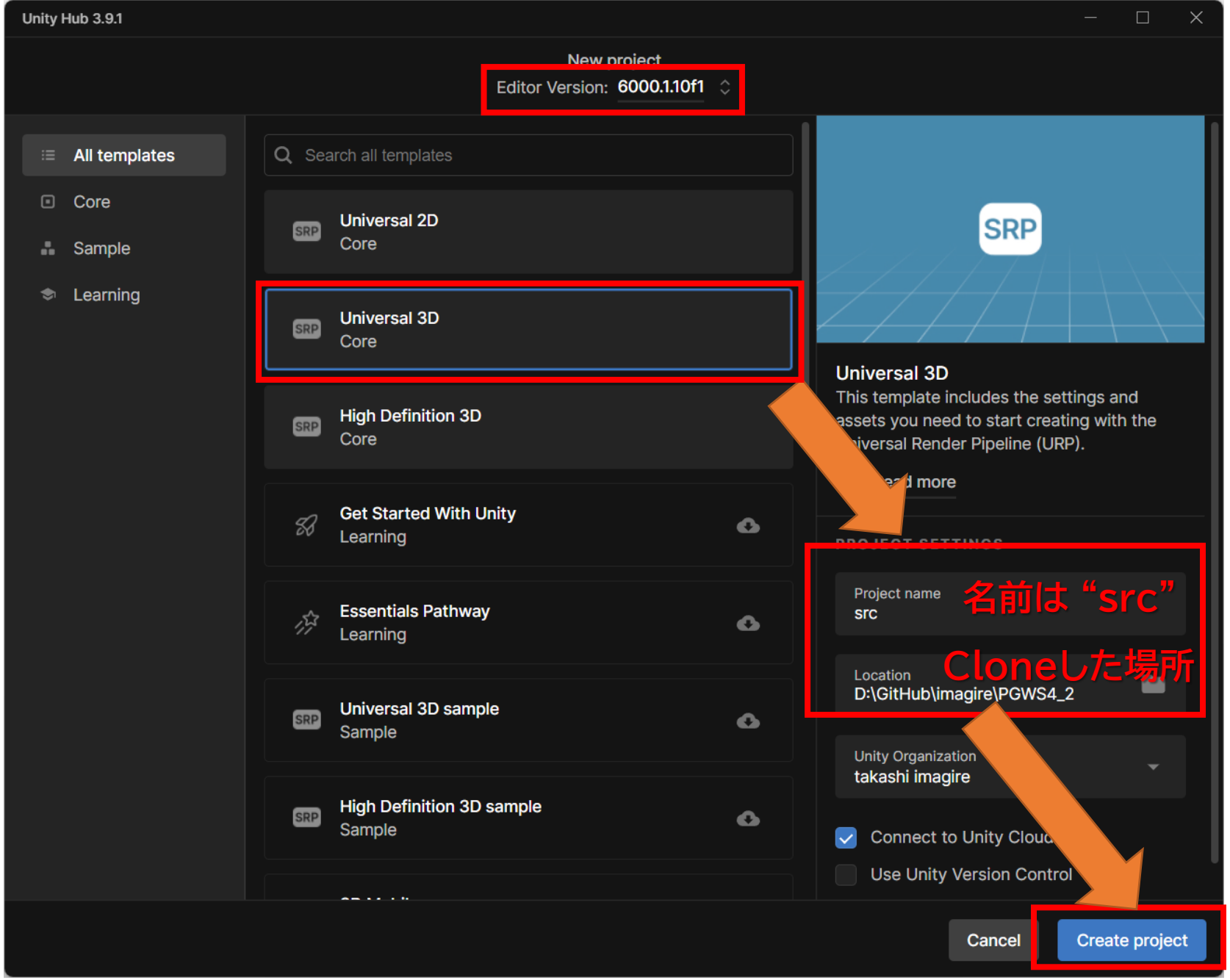




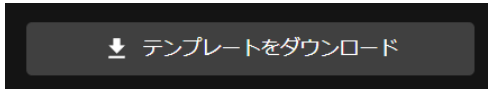
ステップ

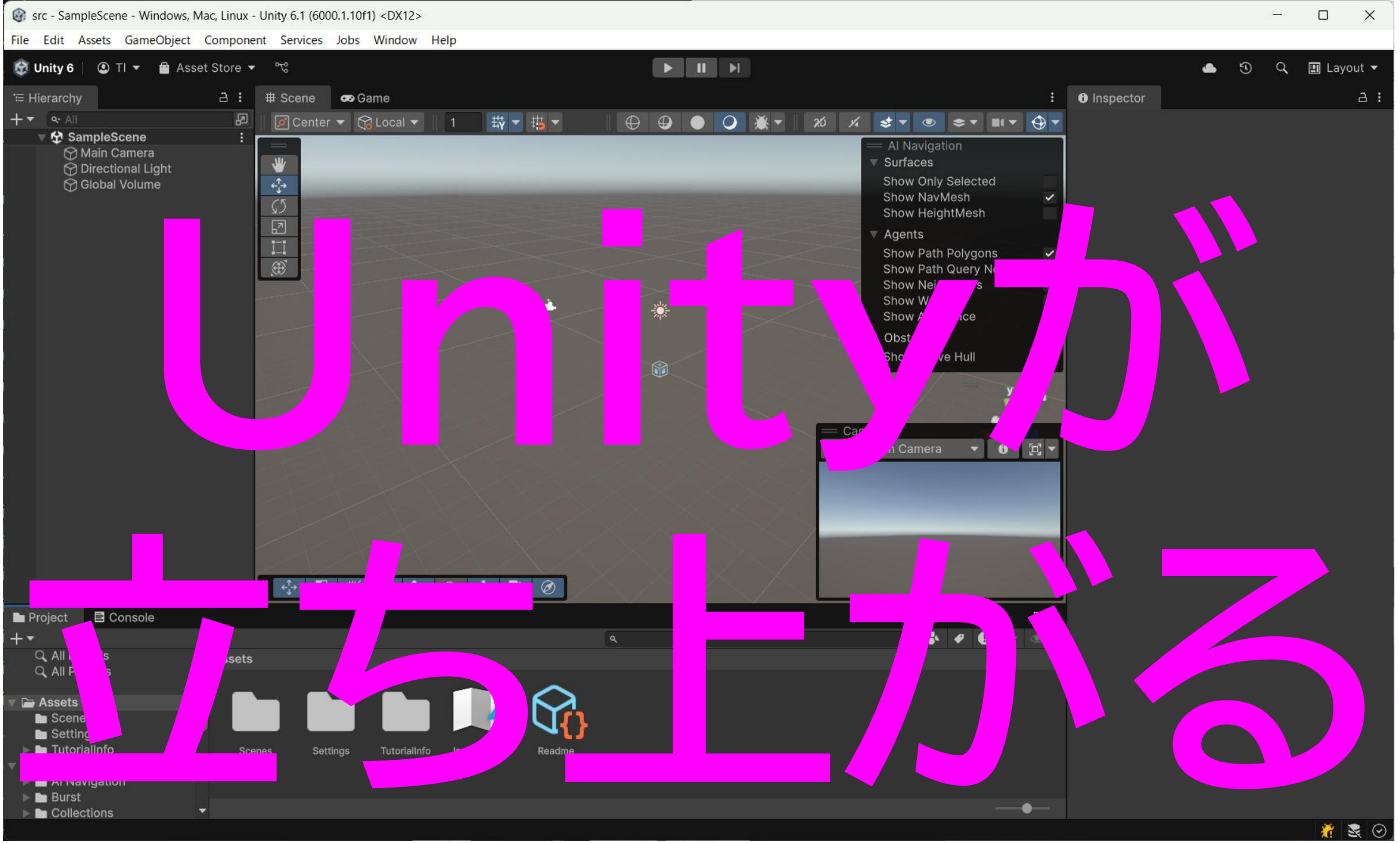
- GitHubリポジトリのクローン
- Unityを立ち上げる
- オブジェクトを作る
- シェーダを作る
- オブジェクトにシェーダを関連付ける
- テクスチャを貼る


Unity Hub 3.9.1
TI
Hub V3.14.0 is now available and will install after restarting.
See Release Notes
Restart now
Dismiss
Projects
Installs
Learn
Community
Downloads
Projects
Add
New project
Search...
NAME
CLOUD
MODIFIED
EDITOR VERSION
2022.3.24f1
2022.3.24f1
2022.3.24f1
2022.3.24f1
6000.0.22f1



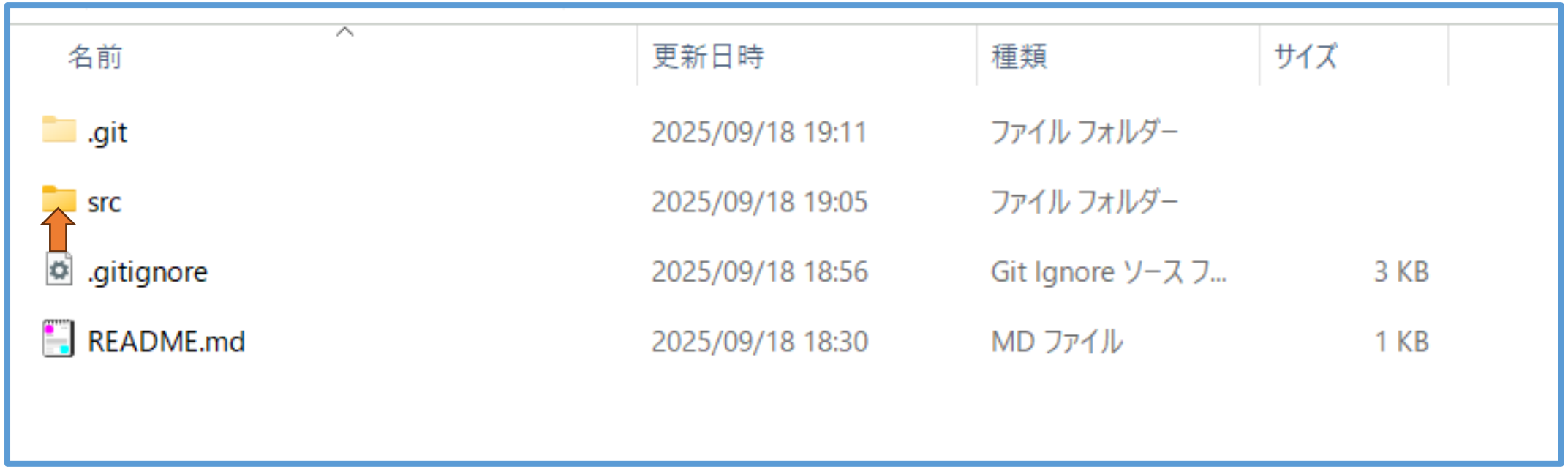
テンプレート
がない場合は、
ダウンロードの
ボタンが出るので
ダウンロードして
作成




src - SampleScene - Windows, Mac, Linux - Unity 6.1 (6000.1.10f1) <DX12>
File Edit Assets GameObject Component Services Jobs Window Help
Unity 6
Asset Store
Layout
Hierarchy
SampleScene
Main Camera
Directional Light
Global Volume
Scene
Game
Inspector
AI Navigation
Surfaces
Show Only Selected
Show NavMesh
Show HeightMesh
Agents
Show Path Polygons
Project
Console
Assets
Scenes
Settings
TutorialInfo
Readme
Burst
Collections
Unityが
立ち上がる

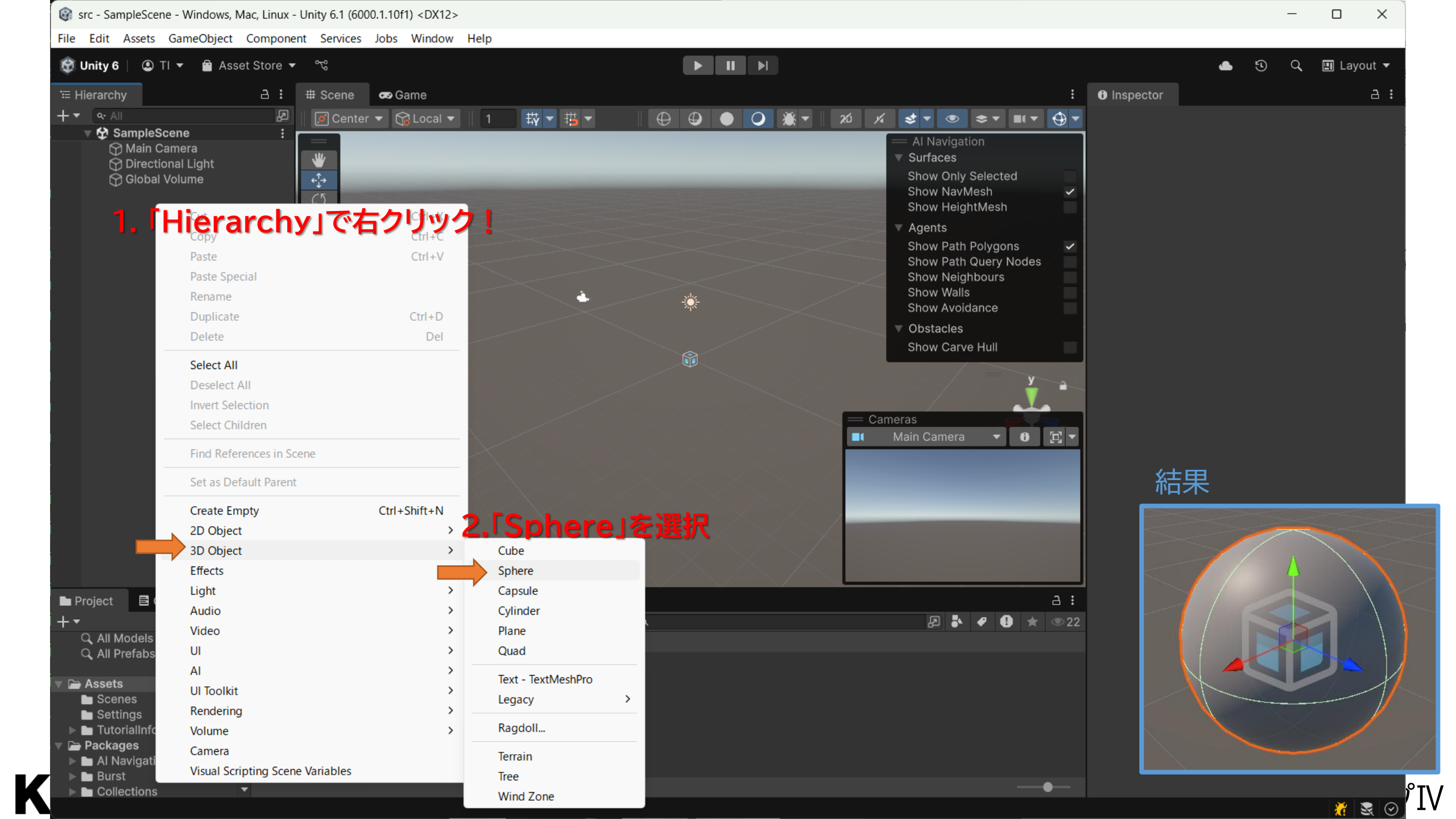
.gitignore

- .gitignore をsrc内に移動する



ステップ

- GitHubリポジトリのクローン
- Unityを立ち上げる
- オブジェクトを作る
- シェーダを作る
- オブジェクトにシェーダを関連付ける
- テクスチャを貼る

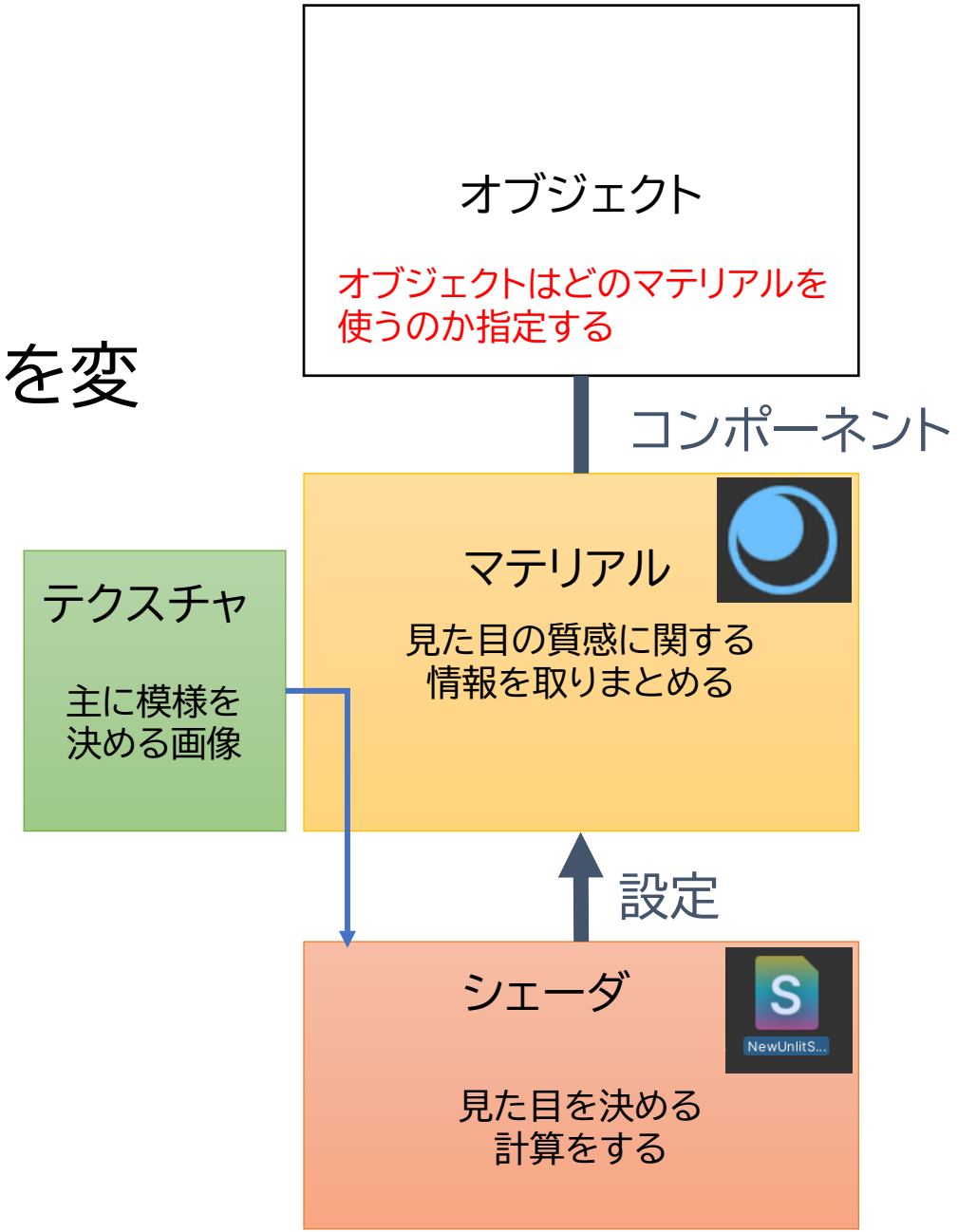

src - SampleScene - Windows, Mac, Linux - Unity 6.1 (6000.1.10f1) <DX12>
File Edit Assets GameObject Component Services Jobs Window Help
Unity 6
Asset Store
Layout
Hierarchy
SampleScene
Main Camera
Directional Light
Global Volume
Scene
Game
Inspector
AI Navigation
Surfaces
Show Only Selected
Show NavMesh
Show HeightMesh
Agents
Show Path Polygons
Show Path Query Nodes
Show Neighbours
Show Walls
Show Avoidance
Obstacles
Show Carve Hull
Cameras
Main Camera
1. 「Hierarchy」で右クリック！
Copy Ctrl+C
Paste Ctrl+V
Paste Special
Rename
Duplicate Ctrl+D
Delete Del
Select All
Deselect All
Invert Selection
Select Children
Find References in Scene
Set as Default Parent
Create Empty Ctrl+Shift+N
2D Object
3D Object
Effects
Light
Audio
Video
UI
AI
UI Toolkit
Rendering
Volume
Camera
Visual Scripting Scene Variables
2.「Sphere」を選択
Cube
Sphere
Capsule
Cylinder
Plane
Quad
Text - TextMeshPro
Legacy
Ragdoll...
Terrain
Tree
Wind Zone
Project
All Models
All Prefabs
Assets
Scenes
Settings
Packages
Burst
Collections
結果

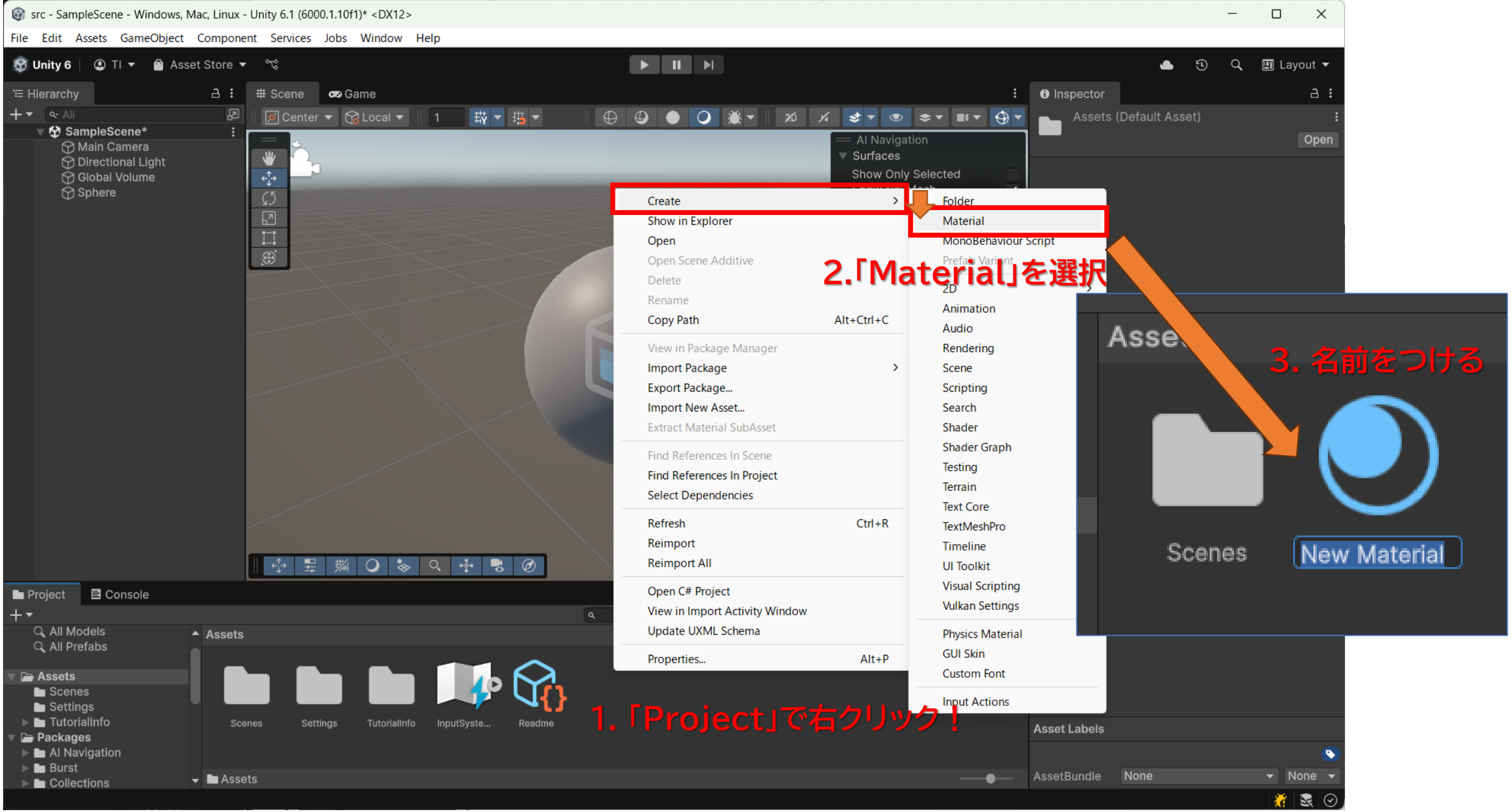
ステップ

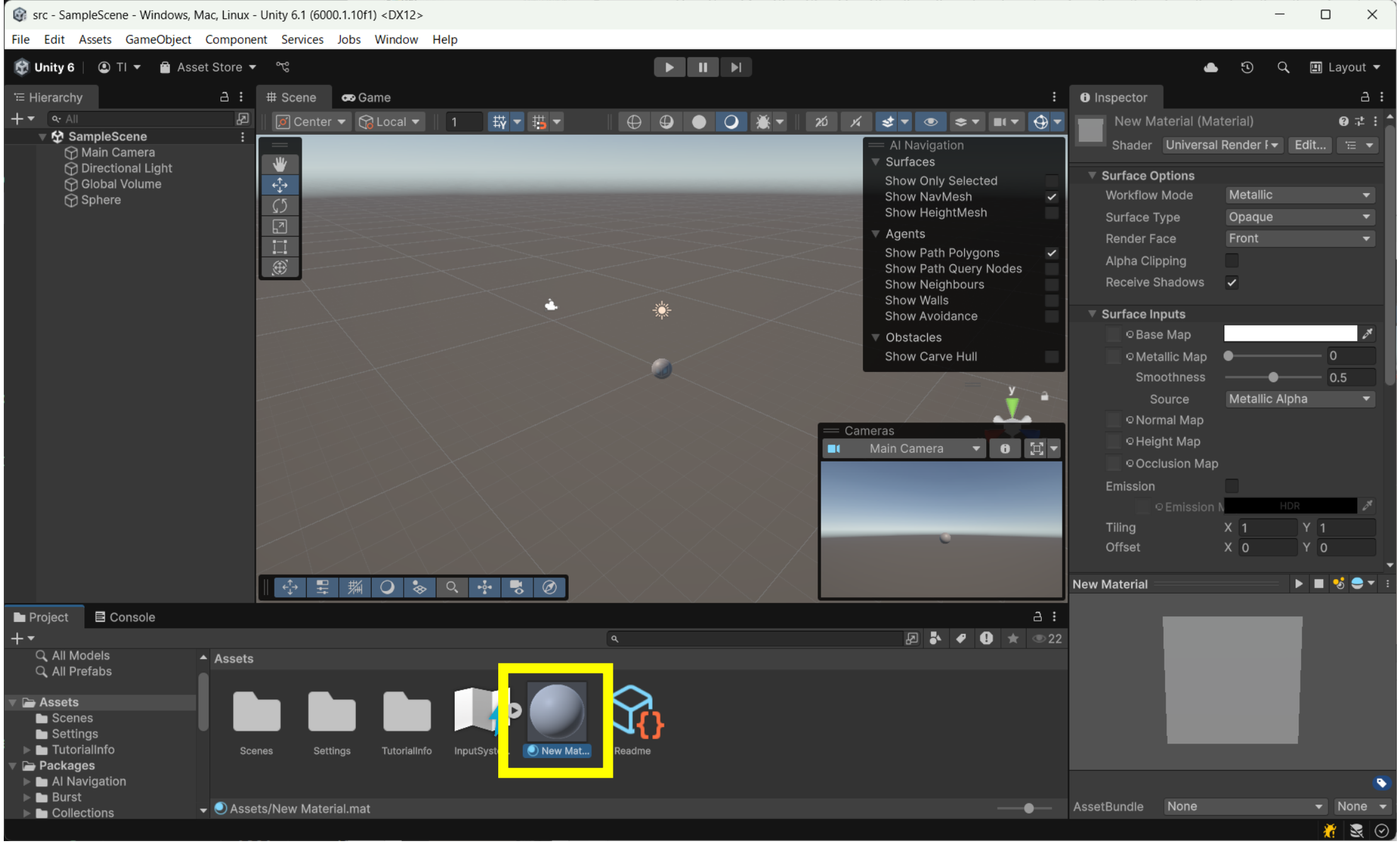
- GitHubリポジトリのクローン
- Unityを立ち上げる
- オブジェクトを作る
- シェーダを作る
- オブジェクトにシェーダを関連付ける
- テクスチャを貼る

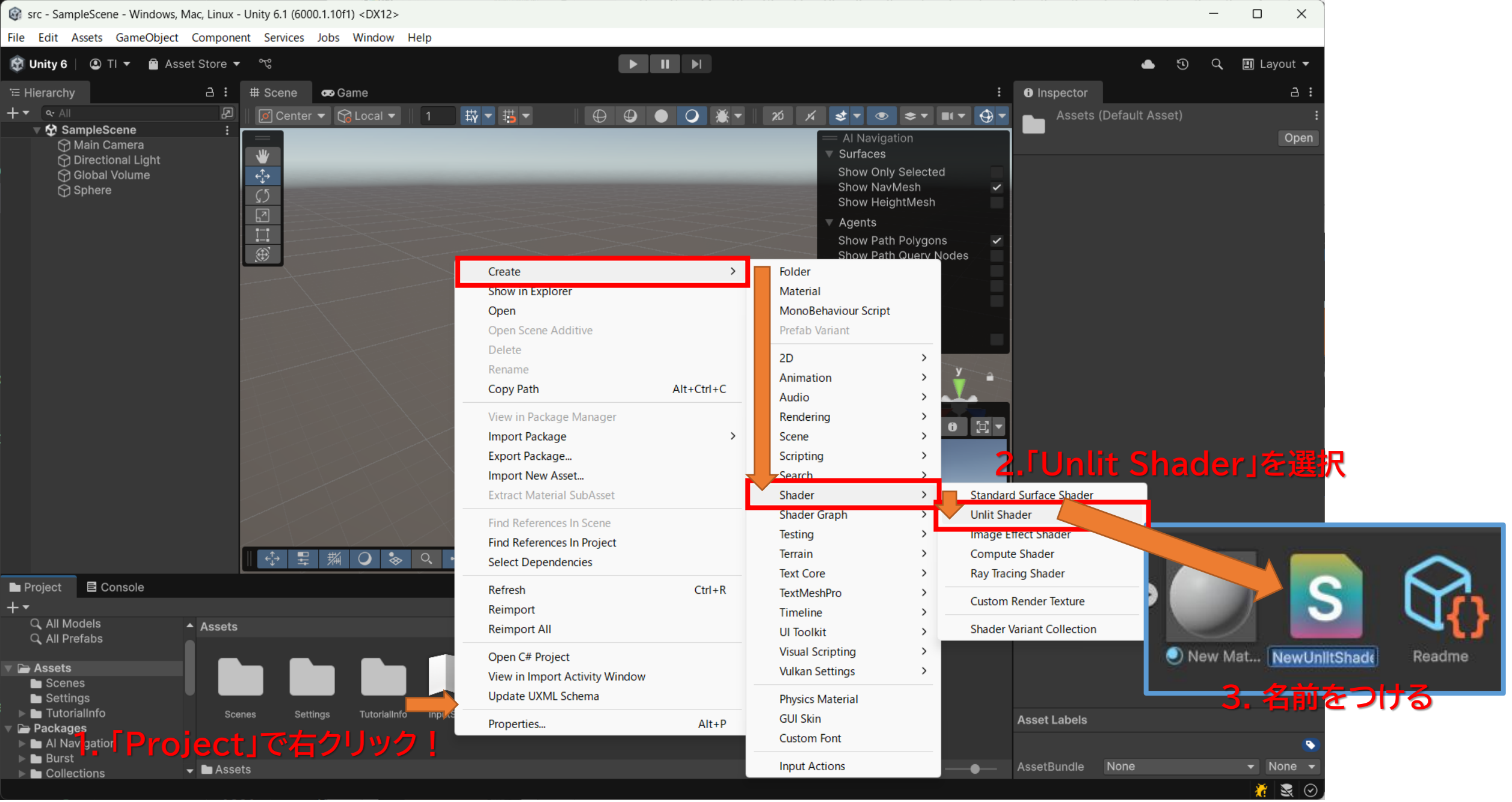
マテリアル

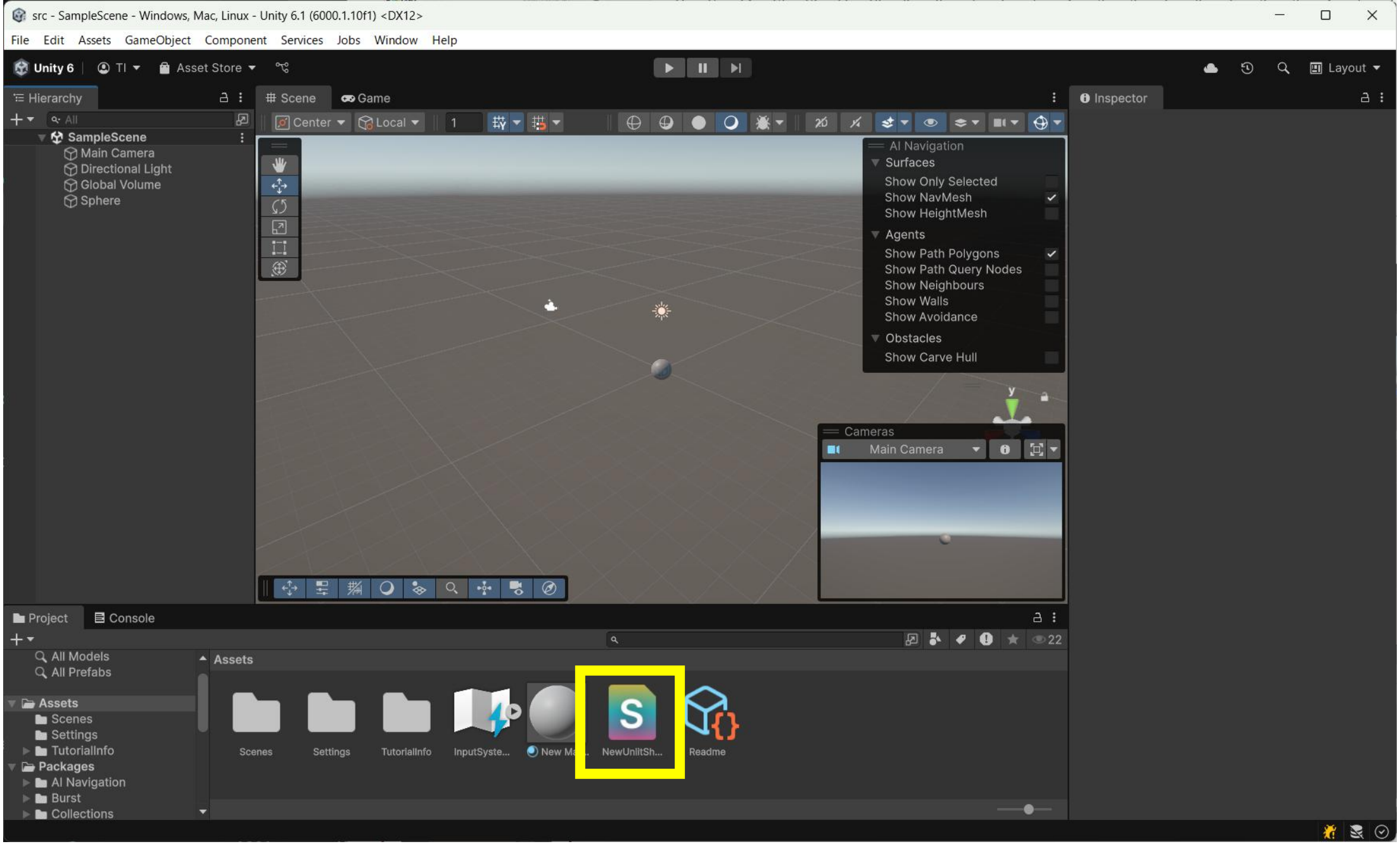
- Unityでは、マテリアルごとに見た目を変えることができる
- マテリアルの中にシェーダが指定されていて、シェーダが実質的な計算を行う
- パラメータはマテリアルごとに指定され、マテリアルからシェーダに渡される




File Edit Assets GameObject Component Services Jobs Window Help
Hierarchy
SampleScene*
Main Camera
Directional Light
Global Volume
Sphere
Scene
Game
Inspector
Assets (Default Asset)
Open
AI Navigation
Surfaces
Show Only Selected
Create
Show in Explorer
Open
Open Scene Additive
Delete
Rename
Copy Path Alt+Ctrl+C
View in Package Manager
Import Package
Export Package...
Import New Asset...
Extract Material SubAsset
Find References In Scene
Find References In Project
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Open C# Project
View in Import Activity Window
Update UXML Schema
Properties... Alt+P
Folder
Material
MonoBehaviour Script
2D
Animation
Audio
Rendering
Scene
Scripting
Search
Shader
Shader Graph
Testing
Terrain
Text Core
TextMeshPro
Timeline
UI Toolkit
Visual Scripting
Vulkan Settings
Physics Material
GUI Skin
Custom Font
Input Actions
2.「Material」を選択
3. 名前をつける
Scenes
New Material
Project
Console
All Models
All Prefabs
Assets
Scenes
Settings
TutorialInfo
Packages
AI Navigation
Burst
Collections
Readme
1.「Project」で右クリック！
Asset Labels
AssetBundle
None


src - SampleScene - Windows, Mac, Linux - Unity 6.1 (6000.1.10f1) <DX12>
File Edit Assets GameObject Component Services Jobs Window Help
Unity 6
Asset Store
Layout
Hierarchy
SampleScene
Main Camera
Directional Light
Global Volume
Sphere
Scene
Game
Center
Local
AI Navigation
Surfaces
Show Only Selected
Show NavMesh
Show HeightMesh
Agents
Show Path Polygons
Show Path Query Nodes
Show Neighbours
Show Walls
Show Avoidance
Obstacles
Show Carve Hull
Cameras
Main Camera
Inspector
New Material (Material)
Shader
Universal Render P
Edit...
Surface Options
Workflow Mode
Metallic
Surface Type
Opaque
Render Face
Front
Alpha Clipping
Receive Shadows
Surface Inputs
Base Map
Metallic Map
Smoothness
0.5
Source
Metallic Alpha
Normal Map
Height Map
Occlusion Map
Emission
Tiling
Offset
New Material
Project
Console
All Models
All Prefabs
Assets
Scenes
Settings
TutorialInfo
Packages
AI Navigation
Burst
Collections
Readme
Assets/New Material.mat
AssetBundle
None

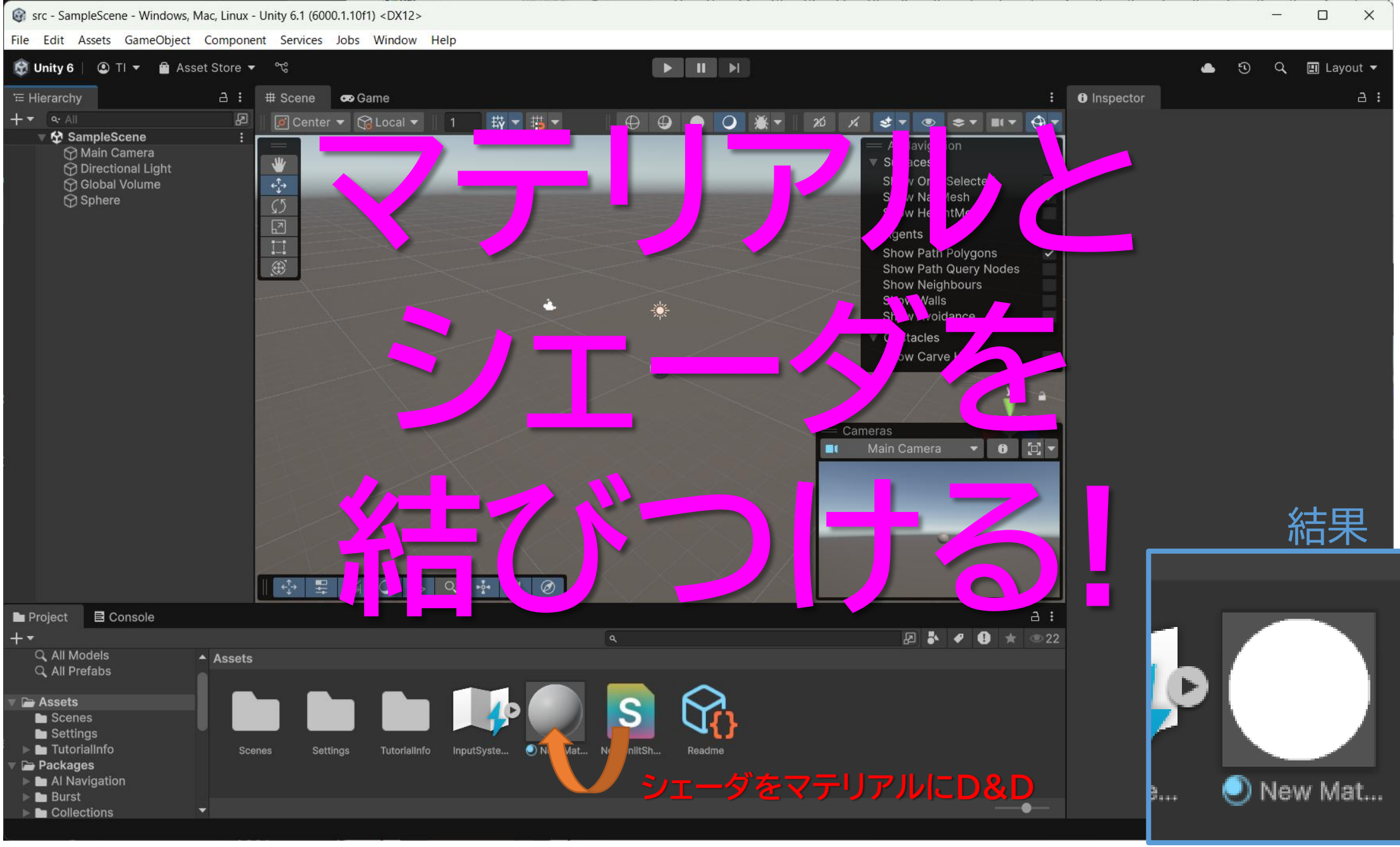

src - SampleScene - Windows, Mac, Linux - Unity 6.1 (6000.1.10f1) <DX12>
File Edit Assets GameObject Component Services Jobs Window Help
Unity 6
Asset Store
Layout
Hierarchy
SampleScene
Main Camera
Directional Light
Global Volume
Sphere
Scene
Game
Inspector
Assets (Default Asset)
Open
AI Navigation
Surfaces
Show Only Selected
Show NavMesh
Show HeightMesh
Agents
Show Path Polygons
Show Path Query Nodes
Create
Show in Explorer
Open
Open Scene Additive
Delete
Rename
Copy Path Alt+Ctrl+C
View in Package Manager
Import Package
Export Package...
Import New Asset...
Extract Material SubAsset
Find References In Scene
Find References In Project
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Open C# Project
View in Import Activity Window
Update UXML Schema
Properties... Alt+P
Folder
Material
MonoBehaviour Script
Prefab Variant
2D
Animation
Audio
Rendering
Scene
Scripting
Search
Shader
Shader Graph
Testing
Terrain
Text Core
TextMeshPro
Timeline
UI Toolkit
Visual Scripting
Vulkan Settings
Physics Material
GUI Skin
Custom Font
Input Actions
Standard Surface Shader
Unlit Shader
Image Effect Shader
Compute Shader
Ray Tracing Shader
Custom Render Texture
Shader Variant Collection
2.「Unlit Shader」を選択
New Mat...
NewUnlitShade
Readme
3. 名前をつける
Project
Console
All Models
All Prefabs
Assets
Scenes
Settings
TutorialInfo
Packages
AI Navigation
Burst
Collections
Assets
Scenes
Settings
TutorialInfo
1.「Project」で右クリック！
Asset Labels
AssetBundle
None
None


src - SampleScene - Windows, Mac, Linux - Unity 6.1 (6000.1.10f1) <DX12>
File Edit Assets GameObject Component Services Jobs Window Help
Unity 6
Asset Store
Layout
Hierarchy
SampleScene
Main Camera
Directional Light
Global Volume
Sphere
Scene
Game
Center
Local
AI Navigation
Surfaces
Show Only Selected
Show NavMesh
Show HeightMesh
Agents
Show Path Polygons
Show Path Query Nodes
Show Neighbours
Show Walls
Show Avoidance
Obstacles
Show Carve Hull
Cameras
Main Camera
Inspector
Project
Console
All Models
All Prefabs
Assets
Scenes
Settings
TutorialInfo
Packages
AI Navigation
Burst
Collections
Scenes
Settings
TutorialInfo
InputSyste...
NewUnlitSh...
Readme

ステップ

- GitHubリポジトリのクローン
- Unityを立ち上げる
- オブジェクトを作る
- シェーダを作る
- オブジェクトにシェーダを関連付ける
- テクスチャを貼る

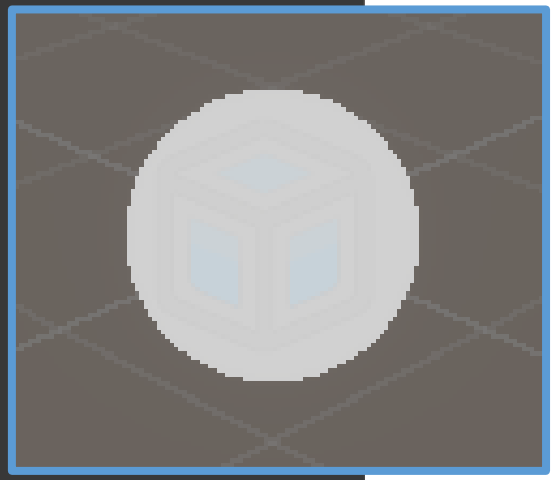
マテリアルとシェーダを結びつける!



結果

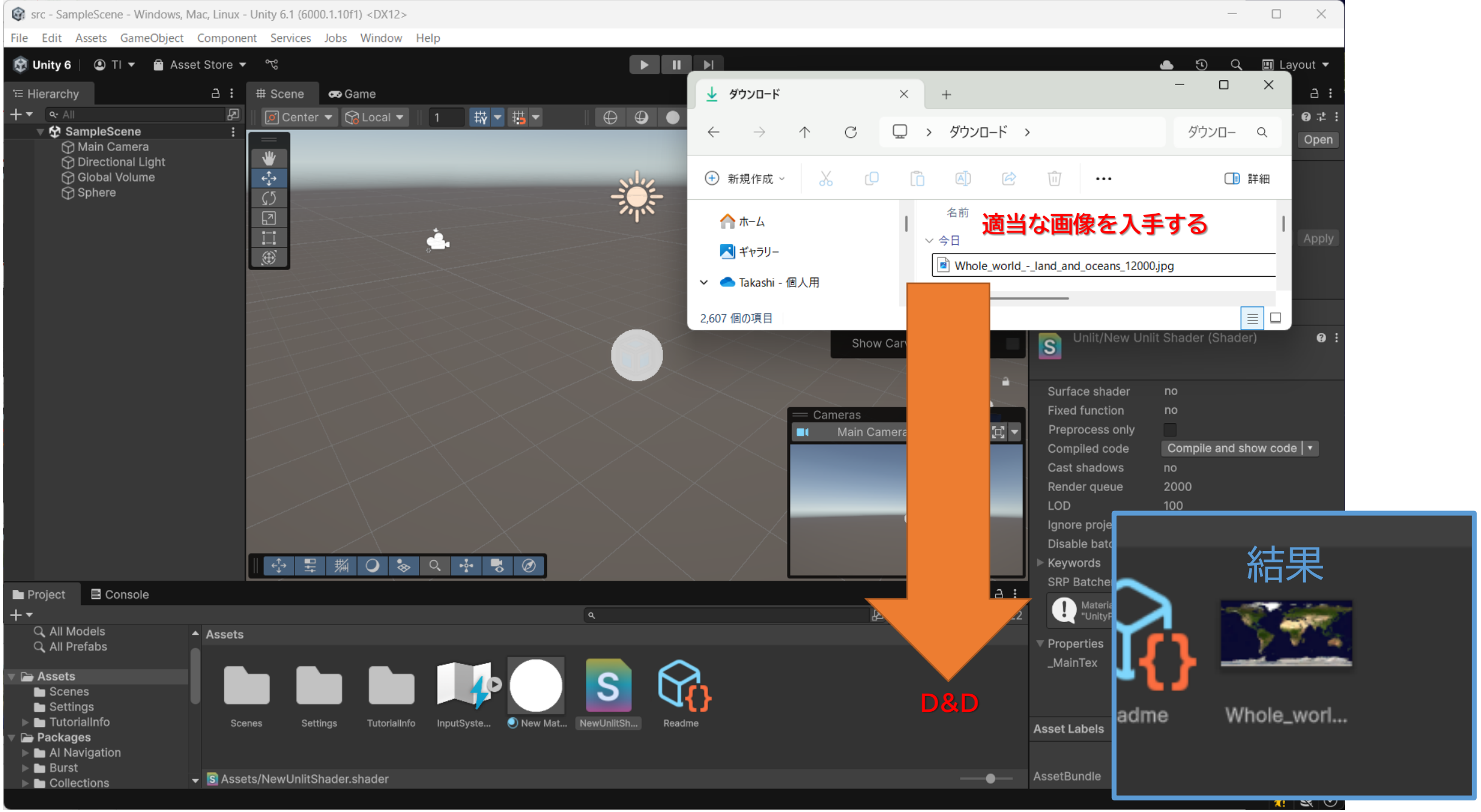
「マテリアル」を「Sphere」にD&D

結果



ステップ

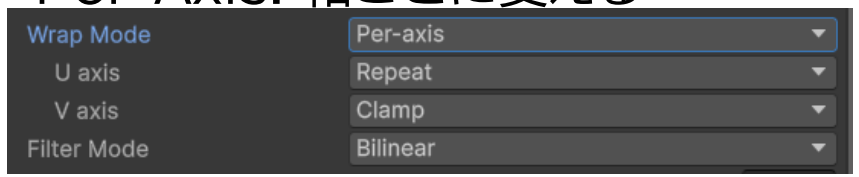
- GitHubリポジトリのクローン
- Unityを立ち上げる
- オブジェクトを作る
- シェーダを作る
- オブジェクトにシェーダを関連付ける
- テクスチャを貼る


適当な画像を入手する
Whole_world_-_land_and_oceans_12000.jpg
D&D
結果

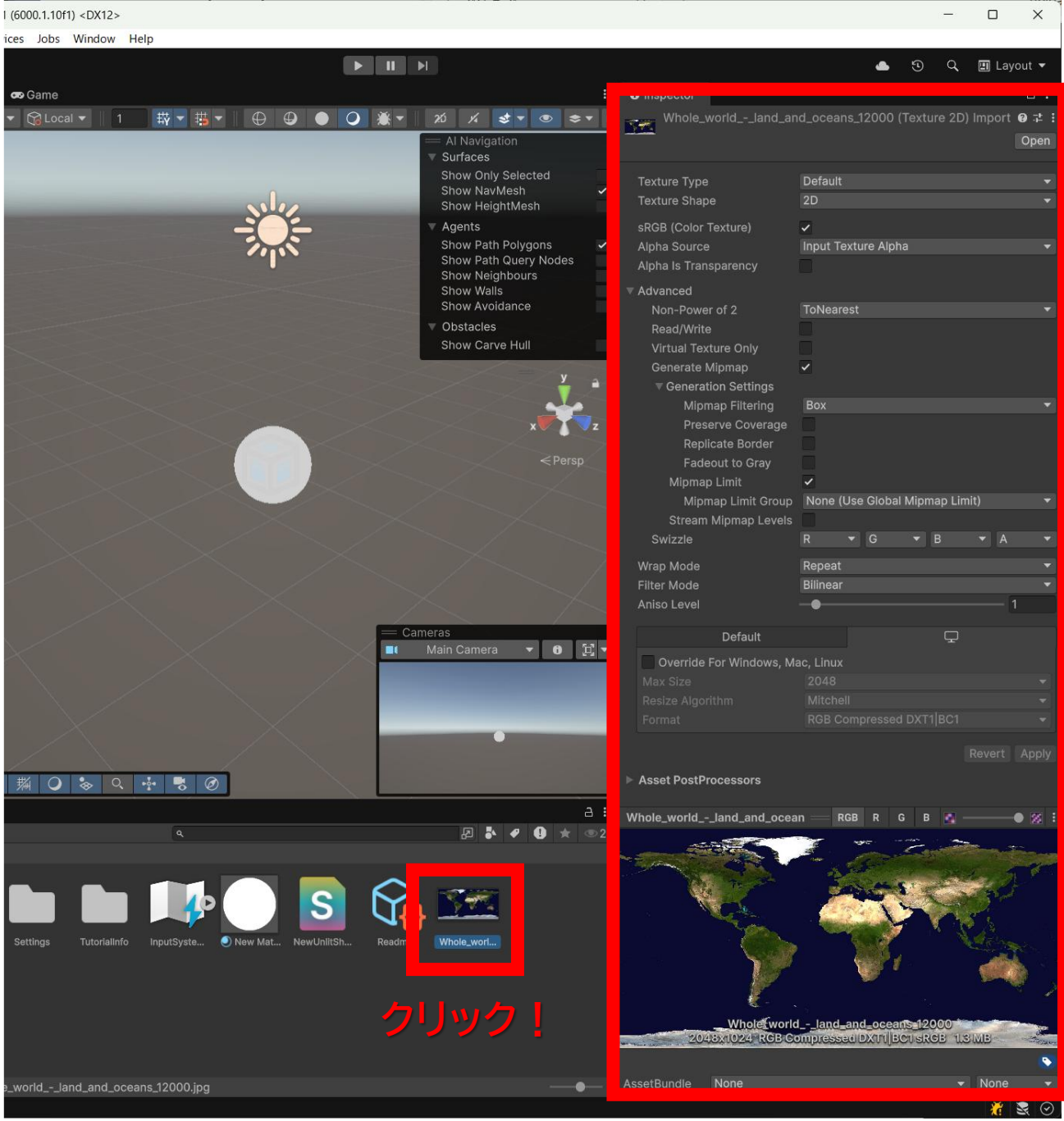
Inspectorで確認

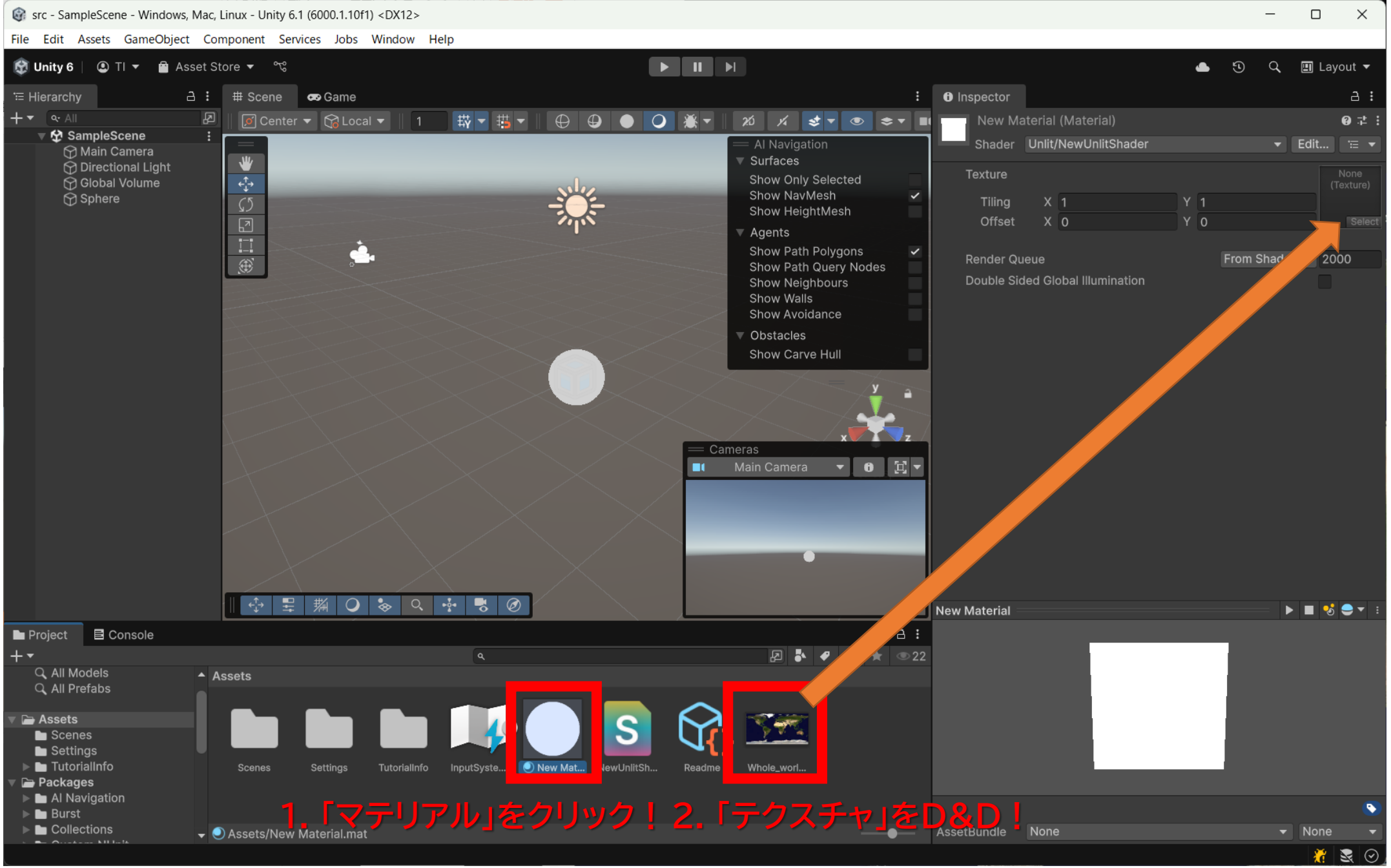
- ここでWrap ModeやFilter Modeを設定する
  - Wrap Mode：画像の外側の模様をどのようにするのか
    - Repeat：周期的に繰り返し
    - Clamp：最外周の色を使う
    - Mirror:反転
    - Mirror Once: 一度だけ反転
    - Per-Axis: 軸ごとに変える

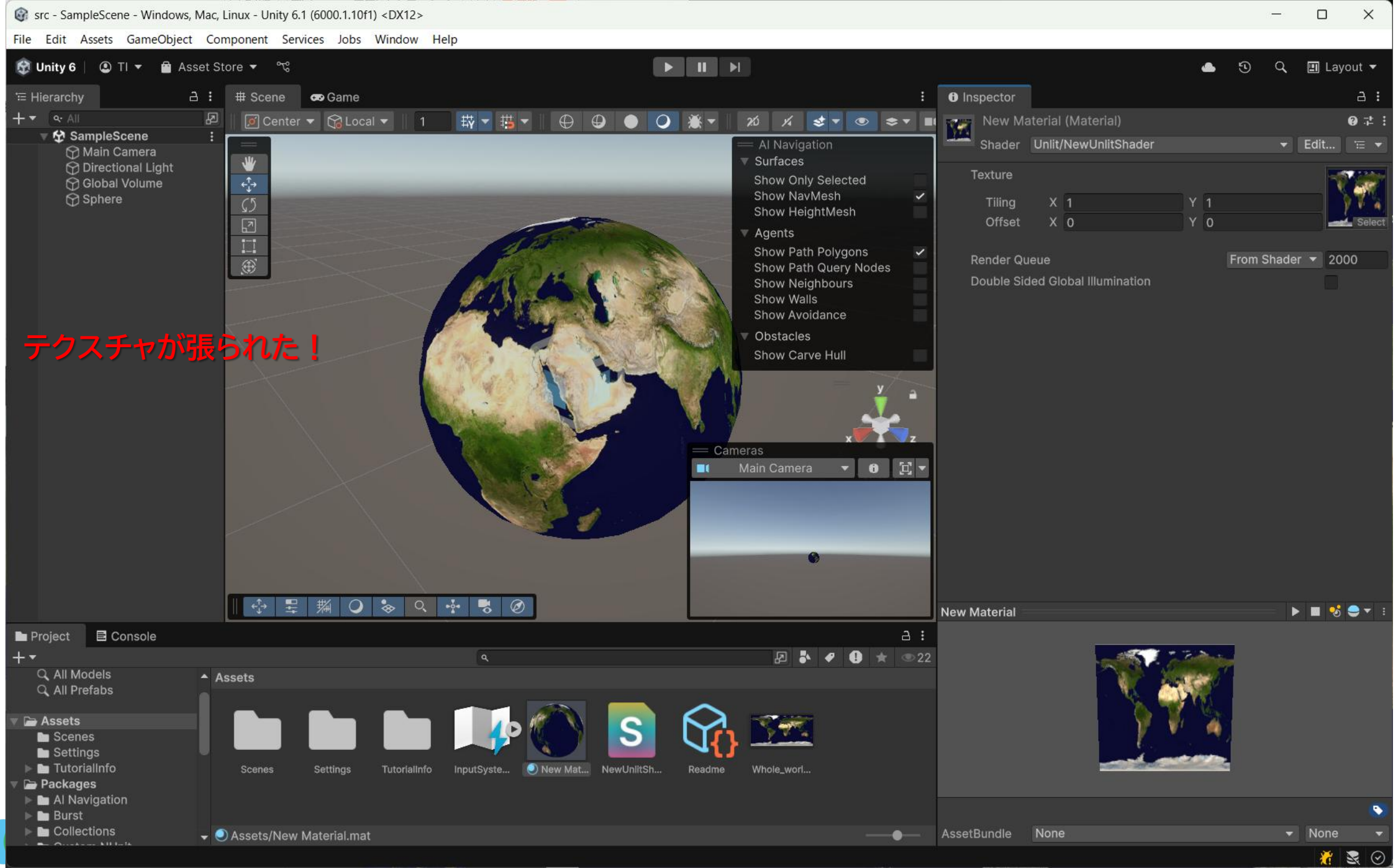
例えば、地球のテクスチャに良い設定はこれ



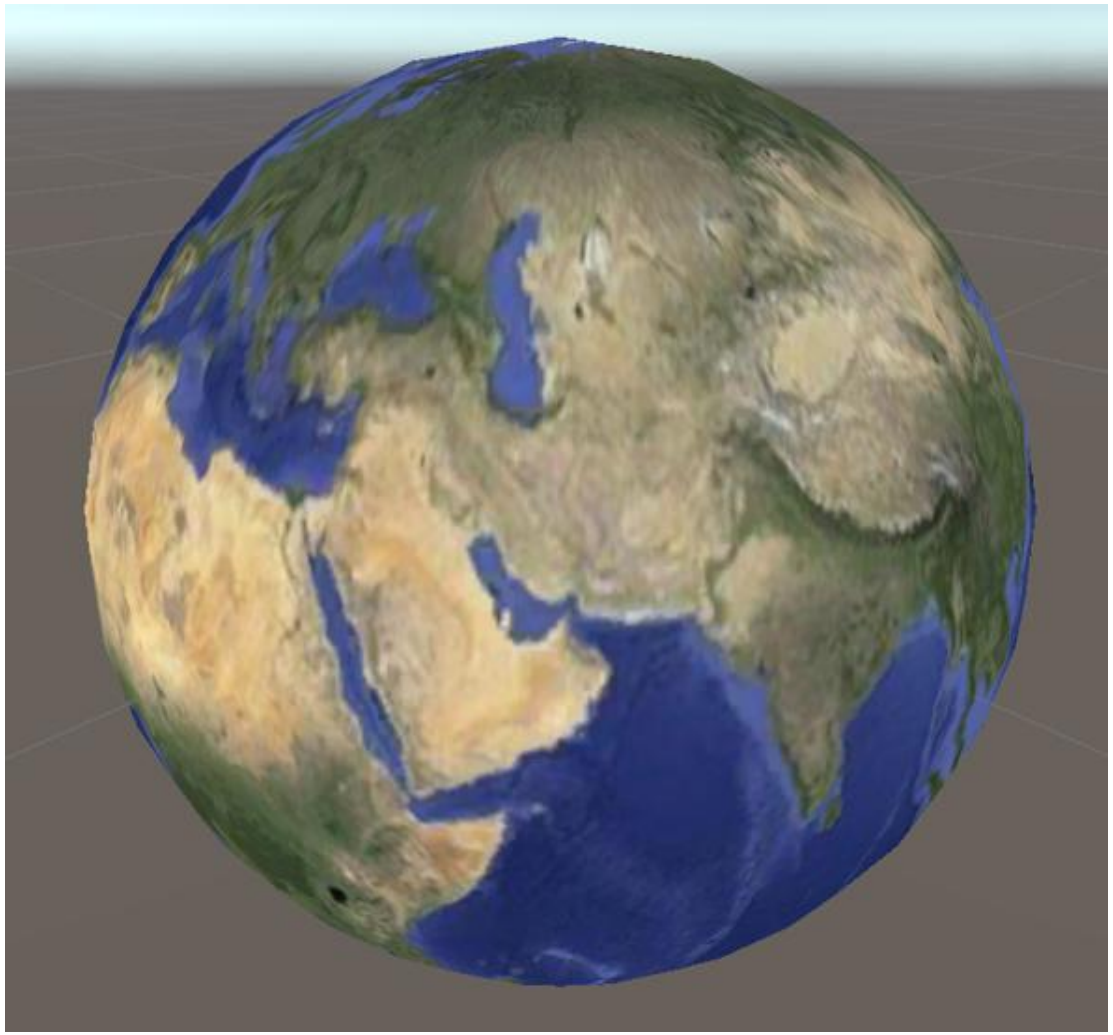
  - Filter Mode：画像をどのようにぼかすのか
    - Point: 一番近い画素の色を使う
    - Bilinear: 線形補間でぼかす
    - Trilinear: LODを考慮した線形補間




File Edit Assets GameObject Component Services Jobs Window Help
Hierarchy
SampleScene
Main Camera
Directional Light
Global Volume
Sphere
Scene
Game
Inspector
New Material (Material)
Shader Unlit/NewUnlitShader
Edit...
Texture
Tiling X 1 Y 1
Offset X 0 Y 0
Render Queue
Double Sided Global Illumination
AI Navigation
Surfaces
Show Only Selected
Show NavMesh
Show HeightMesh
Agents
Show Path Polygons
Show Path Query Nodes
Show Neighbours
Show Walls
Show Avoidance
Obstacles
Show Carve Hull
Cameras
Main Camera
Project
Console
Assets
Scenes
Settings
TutorialInfo
Packages
AI Navigation
Burst
Collections
New Material
Assets/New Material.mat
1.「マテリアル」をクリック！ 2.「テクスチャ」をD&D！

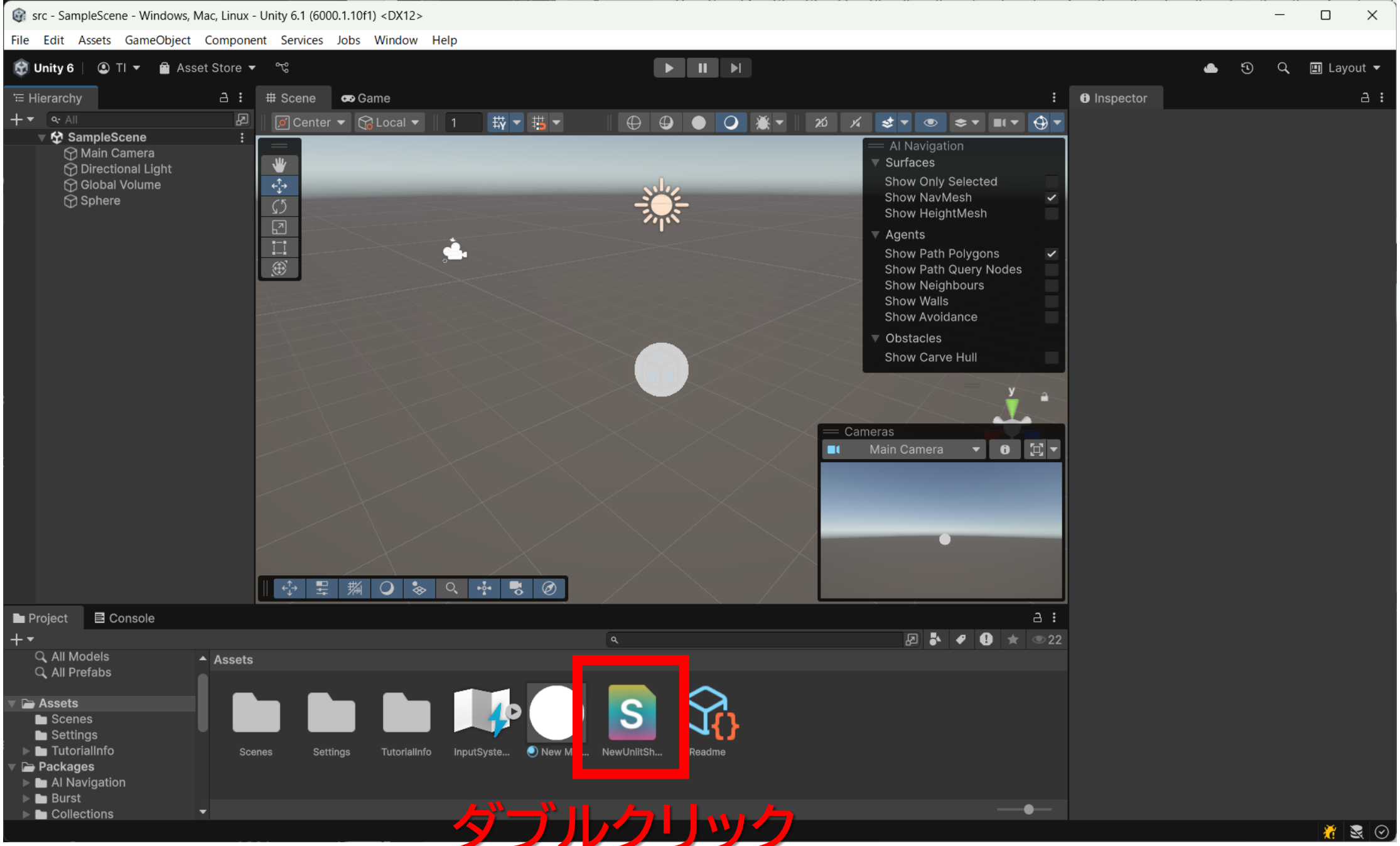

src - SampleScene - Windows, Mac, Linux - Unity 6.1 (6000.1.10f1) <DX12>
File Edit Assets GameObject Component Services Jobs Window Help
Unity 6
Asset Store
Layout
Hierarchy
SampleScene
Main Camera
Directional Light
Global Volume
Sphere
Scene
Game
Center
Local
AI Navigation
Surfaces
Show Only Selected
Show NavMesh
Show HeightMesh
Agents
Show Path Polygons
Show Path Query Nodes
Show Neighbours
Show Walls
Show Avoidance
Obstacles
Show Carve Hull
テクスチャが張られた！
Cameras
Main Camera
Inspector
New Material (Material)
Shader
Unlit/NewUnlitShader
Edit...
Texture
Tiling X 1 Y 1
Offset X 0 Y 0
Select
Render Queue
From Shader
2000
Double Sided Global Illumination
New Material
Project
Console
All Models
All Prefabs
Assets
Scenes
Settings
TutorialInfo
Packages
AI Navigation
Burst
Collections
Assets
Scenes
Settings
TutorialInfo
InputSyste...
New Mat...
NewUnlitSh...
Readme
Whole_worl...
Assets/New Material.mat
AssetBundle
None
None

やってみよう



目次

- シェーダ入門
- 簡単なシェーダ
- シェーダを読む
- HLSL入門


src - SampleScene - Windows, Mac, Linux - Unity 6.1 (6000.1.10f1) <DX12>
File Edit Assets GameObject Component Services Jobs Window Help
Unity 6
Asset Store
Layout
Hierarchy
SampleScene
Main Camera
Directional Light
Global Volume
Sphere
Scene
Game
Center
Local
AI Navigation
Surfaces
Show Only Selected
Show NavMesh
Show HeightMesh
Agents
Show Path Polygons
Show Path Query Nodes
Show Neighbours
Show Walls
Show Avoidance
Obstacles
Show Carve Hull
Cameras
Main Camera
Inspector
Project
Console
All Models
All Prefabs
Assets
Scenes
Settings
TutorialInfo
Packages
AI Navigation
Burst
Collections
Assets
Scenes
Settings
TutorialInfo
InputSyste...
NewUnlitSh...
Readme
ダブルクリック

VSでコードを確認・修正できる

- Unlit シェーダ
  - 照明計算を行わないシェーダ
    - 色や画像の模様をそのまま出力する

```
Shader "Unlit/NewUnlitShader"
{
    Properties
    {
        _MainTex ("Texture", 2D) = "white" {}
    }
    SubShader
    {
        Tags { "RenderType"="Opaque" }
        LOD 100

        Pass
        {
            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            // make fog work
            #pragma multi_compile_fog

            #include "UnityCG.cginc"

            struct appdata
            {
                float4 vertex : POSITION;
                float2 uv : TEXCOORD0;
            };

            struct v2f
            {
                float2 uv : TEXCOORD0;
                UNITY_FOG_COORDS(1)
                float4 vertex : SV_POSITION;
            };

            sampler2D _MainTex;
            float4 _MainTex_ST;

            v2f vert (appdata v)
            {
                v2f o;
                o.vertex = UnityObjectToClipPos(v.vertex);
                o.uv = TRANSFORM_TEX(v.uv, _MainTex);
                UNITY_TRANSFER_FOG(o,o.vertex);
                return o;
            }

            fixed4 frag (v2f i) : SV_Target
            {
                // sample the texture
                fixed4 col = tex2D(_MainTex, i.uv);
                // apply fog
                UNITY_APPLY_FOG(i.fogCoord, col);
                return col;
            }
            ENDCG
        }
    }
}
```

頂点・ピクセルシェーダ

- それぞれの処理を関数として書く
- 構造体でデータを受け渡す

```
38              v2f vert (appdata v)
39              {
40                  v2f o;
41                  o.vertex = UnityObjectToClipPos(v.vertex);
42                  o.uv = TRANSFORM_TEX(v.uv, _MainTex);
43                  UNITY_TRANSFER_FOG(o,o.vertex);
44                  return o;
45              }
46
47              fixed4 frag (v2f i) : SV_Target
48              {
49                  // sample the texture
50                  fixed4 col = tex2D(_MainTex, i.uv);
51                  // apply fog
52                  UNITY_APPLY_FOG(i.fogCoord, col);
53                  return col;
54              }
```

頂点シェーダ関数

返り値として出力を返す

頂点シェーダの出力はラスタライズされて
ピクセルシェーダに引き渡される

4成分固定小数点数

返り値として出力を返す

ピクセルシェーダ関数

入力

出力

入力

```
1   Shader "Unlit/NewUnlitShader"
2   {
3       Properties
4       {
5           _MainTex ("Texture", 2D) = "white" {}
6       }
7       SubShader
8       {
9           Tags { "RenderType"="Opaque" }
10          LOD 100
11
12          Pass
13          {
14              CGPROGRAM
15              #pragma vertex vert
16              #pragma fragment frag
17              // make fog work
18              #pragma multi_compile_fog
19
20              #include "UnityCG.cginc"
21
22              struct appdata
23              {
24                  float4 vertex : POSITION;
25                  float2 uv : TEXCOORD0;
26              };
27
28              struct v2f
29              {
30                  float2 uv : TEXCOORD0;
31                  UNITY_FOG_COORDS(1)
32                  float4 vertex : SV_POSITION;
33              };
34
35              sampler2D _MainTex;
36              float4 _MainTex_ST;
37
38              v2f vert (appdata v)
39              {
40                  v2f o;
41                  o.vertex = UnityObjectToClipPos(v.vertex);
42                  o.uv = TRANSFORM_TEX(v.uv, _MainTex);
43                  UNITY_TRANSFER_FOG(o,o.vertex);
44                  return o;
45              }
46
47              fixed4 frag (v2f i) : SV_Target
48              {
49                  // sample the texture
50                  fixed4 col = tex2D(_MainTex, i.uv);
51                  // apply fog
52                  UNITY_APPLY_FOG(i.fogCoord, col);
53                  return col;
54              }
55              ENDCG
56          }
57      }
58  }
```

テクスチャ

- tex*** という関数で読み込む

```
38          v2f vert (appdata v)
39          {
40              v2f o;                                  頂点シェーダ関数
41              o.vertex = UnityObjectToClipPos(v.vertex);
42              o.uv = TRANSFORM_TEX(v.uv, _MainTex);
43              UNITY_TRANSFER_FOG(o,o.vertex);
44              return o;
45          }
46
47  4成分固定小数点数  fixed4 frag (v2f i) : SV_Target
48          {
49              // sample the texture
50              fixed4 col = tex2D(_MainTex, i.uv);
51              // apply fog
52              UNITY_APPLY_FOG(i.fogCoord, col);
53              return col;
54          }                                           ピクセルシェーダ関数
```

```
1   Shader "Unlit/NewUnlitShader"
2   {
3       Properties
4       {
5           _MainTex ("Texture", 2D) = "white" {}
6       }
7       SubShader
8       {
9           Tags { "RenderType"="Opaque" }
10          LOD 100
11
12          Pass
13          {
14              CGPROGRAM
15              #pragma vertex vert
16              #pragma fragment frag
17              // make fog work
18              #pragma multi_compile_fog
19
20              #include "UnityCG.cginc"
21
22              struct appdata
23              {
24                  float4 vertex : POSITION;
25                  float2 uv : TEXCOORD0;
26              };
27
28              struct v2f
29              {
30                  float2 uv : TEXCOORD0;
31                  UNITY_FOG_COORDS(1)
32                  float4 vertex : SV_POSITION;
33              };
34
35              sampler2D _MainTex;
36              float4 _MainTex_ST;
37
38              v2f vert (appdata v)
39              {
40                  v2f o;
41                  o.vertex = UnityObjectToClipPos(v.vertex);
42                  o.uv = TRANSFORM_TEX(v.uv, _MainTex);
43                  UNITY_TRANSFER_FOG(o,o.vertex);
44                  return o;
45              }
46
47              fixed4 frag (v2f i) : SV_Target
48              {
49                  // sample the texture
50                  fixed4 col = tex2D(_MainTex, i.uv);
51                  // apply fog
52                  UNITY_APPLY_FOG(i.fogCoord, col);
53                  return col;
54              }
55              ENDCG
56          }
57      }
58  }
```

テクスチャ

- tex*** という関数で読み込む
  - UV座標:頂点データから流されてくる

```
38              v2f vert (appdata v)
39              {
40                  v2f o;                          頂点シェーダ関数
41                  o.vertex = UnityObjectToClipPos(v.vertex);
42                  o.uv = TRANSFORM_TEX(v.uv, _MainTex);
43                  UNITY_TRANSFER_FOG(o,o.vertex);
44                  return o;
45              }
46
47  4成分固定小数点数  fixed4 frag (v2f i) : SV_Target
48              {
49                  // sample the texture
50                  fixed4 col = tex2D(_MainTex, i.uv);
51                  // apply fog
52                  UNITY_APPLY_FOG(i.fogCoord, col);
53                  return col;
54              }                                   ピクセルシェーダ関数
```

```
1   Shader "Unlit/NewUnlitShader"
2   {
3       Properties
4       {
5           _MainTex ("Texture", 2D) = "white" {}
6       }
7       SubShader
8       {
9           Tags { "RenderType"="Opaque" }
10          LOD 100
11
12          Pass
13          {
14              CGPROGRAM
15              #pragma vertex vert
16              #pragma fragment frag
17              // make fog work
18              #pragma multi_compile_fog
19
20              #include "UnityCG.cginc"
21
22              struct appdata
23              {
24                  float4 vertex : POSITION;
25                  float2 uv : TEXCOORD0;
26              };
27
28              struct v2f
29              {
30                  float2 uv : TEXCOORD0;
31                  UNITY_FOG_COORDS(1)
32                  float4 vertex : SV_POSITION;
33              };
34
35              sampler2D _MainTex;
36              float4 _MainTex_ST;
37
38              v2f vert (appdata v)
39              {
40                  v2f o;
41                  o.vertex = UnityObjectToClipPos(v.vertex);
42                  o.uv = TRANSFORM_TEX(v.uv, _MainTex);
43                  UNITY_TRANSFER_FOG(o,o.vertex);
44                  return o;
45              }
46
47              fixed4 frag (v2f i) : SV_Target
48              {
49                  // sample the texture
50                  fixed4 col = tex2D(_MainTex, i.uv);
51                  // apply fog
52                  UNITY_APPLY_FOG(i.fogCoord, col);
53                  return col;
54              }
55              ENDCG
56          }
57      }
58  }
```

テクスチャ

- tex*** という関数で読み込む
  - UV座標:頂点データから流されてくる
  - サンプラーオブジェクト:画像データと読み込み方

```
38              v2f vert (appdata v)
39              {                                   頂点シェーダ関数
40                  v2f o;
41                  o.vertex = UnityObjectToClipPos(v.vertex);
42                  o.uv = TRANSFORM_TEX(v.uv, _MainTex);
43                  UNITY_TRANSFER_FOG(o,o.vertex);
44                  return o;
45              }
46
47  4成分固定小数点数  fixed4 frag (v2f i) : SV_Target
48              {
49                  // sample the texture
50                  fixed4 col = tex2D(_MainTex, i.uv);
51                  // apply fog
52                  UNITY_APPLY_FOG(i.fogCoord, col);
53                  return col;
54              }                                   ピクセルシェーダ関数
```

```
1   Shader "Unlit/NewUnlitShader"
2   {
3       Properties
4       {
5           _MainTex ("Texture", 2D) = "white" {}
6       }
7       SubShader
8       {
9           Tags { "RenderType"="Opaque" }
10          LOD 100
11
12          Pass
13          {
14              CGPROGRAM
15              #pragma vertex vert
16              #pragma fragment frag
17              // make fog work
18              #pragma multi_compile_fog
19
20              #include "UnityCG.cginc"
21
22              struct appdata
23              {
24                  float4 vertex : POSITION;
25                  float2 uv : TEXCOORD0;
26              };
27
28              struct v2f
29              {
30                  float2 uv : TEXCOORD0;
31                  UNITY_FOG_COORDS(1)
32                  float4 vertex : SV_POSITION;
33              };
34
35              sampler2D _MainTex;
36              float4 _MainTex_ST;
37
38              v2f vert (appdata v)
39              {
40                  v2f o;
41                  o.vertex = UnityObjectToClipPos(v.vertex);
42                  o.uv = TRANSFORM_TEX(v.uv, _MainTex);
43                  UNITY_TRANSFER_FOG(o,o.vertex);
44                  return o;
45              }
46
47              fixed4 frag (v2f i) : SV_Target
48              {
49                  // sample the texture
50                  fixed4 col = tex2D(_MainTex, i.uv);
51                  // apply fog
52                  UNITY_APPLY_FOG(i.fogCoord, col);
53                  return col;
54              }
55              ENDCG
56          }
57      }
58  }
```

ShaderLab

- CGPROGRAM / ENDCGで囲った中がHLSLの世界
  - Microsoftのドキュメントが役立つ
  - その外側はUnityの独自拡張
    - HLSLは、Unreal Engine でも役立つが、ShaderLabの知識は他の環境では役に立たない（似たようなものはあるかもしれない）

```
Shader "Unlit/NewUnlitShader"
{
    Properties
    {
        _MainTex ("Texture", 2D) = "white" {}
    }
    SubShader
    {
        Tags { "RenderType"="Opaque" }
        LOD 100

        Pass
        {
            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            // make fog work
            #pragma multi_compile_fog

            #include "UnityCG.cginc"

            struct appdata
            {...
            };

            struct v2f
            {...
            };

            sampler2D _MainTex;
            float4 _MainTex_ST;

            v2f vert (appdata v)
            {...
            }

            fixed4 frag (v2f i) : SV_Target
            {...
            }
            ENDCG
        }
    }
}
```

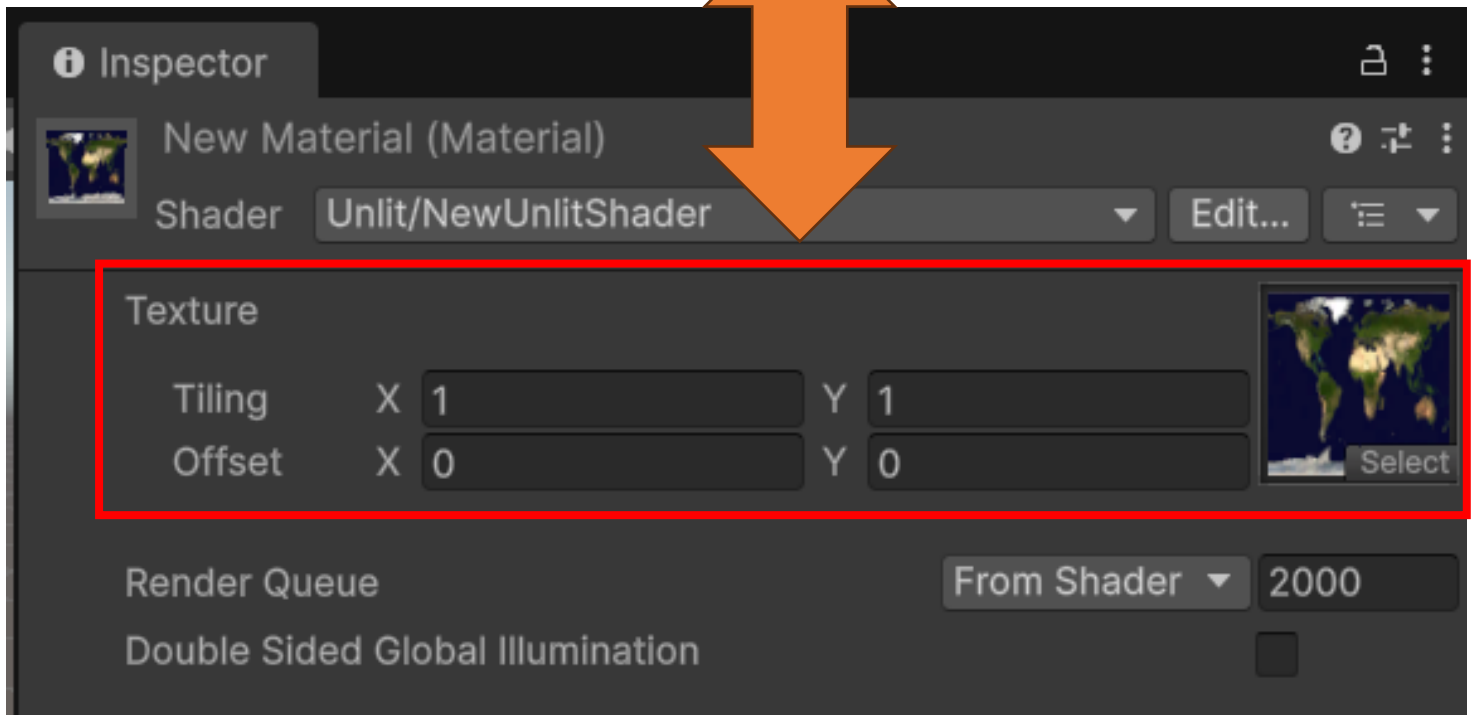
ShaderLab

HLSL

Properties

- インスペクターとの連携
  - 適切なフォーマットで
    コードを表記すると、
    設定項目がマテリアルの
    設定に現れる
  - テクスチャの様なシェーダの設定は、マテリアルごとに行われるので、基本的に同じシェーダでも
    違う画像を使うなら
    別のマテリアルが必要

```
1   Shader "Unlit/NewUnlitShader"
2   {
3       Properties
4       {
5           _MainTex ("Texture", 2D) = "white" {}
6       }
```



Propertiesの例

数字とスライダー

```
name ("display name", Range (min, max)) = number
name ("display name", Float) = number
name ("display name", Int) = number
```

カラーとベクトル

```
name ("display name", Color) = (number,number,number,number)
name ("display name", Vector) = (number,number,number,number)
```

テクスチャ

```
name ("display name", 2D) = "defaulttexture" {}
name ("display name", Cube) = "defaulttexture" {}
name ("display name", 3D) = "defaulttexture" {}
```

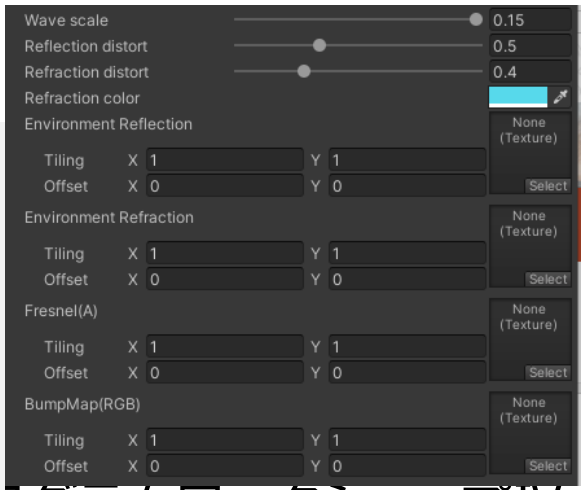
例

```
// water シェーダーのプロパティー
Properties
{
    _WaveScale ("Wave scale", Range (0.02,0.15)) = 0.07 // スライダー
    _ReflDistort ("Reflection distort", Range (0,1.5)) = 0.5
    _RefrDistort ("Refraction distort", Range (0,1.5)) = 0.4
    _RefrColor ("Refraction color", Color) = (.34, .85, .92, 1) // カラー
    _ReflectionTex ("Environment Reflection", 2D) = "" {} // テクスチャ
    _RefractionTex ("Environment Refraction", 2D) = "" {}
    _Fresnel ("Fresnel (A) ", 2D) = "" {}
    _BumpMap ("Bumpmap (RGB) ", 2D) = "" {}
}
```

変数名　エディタでの名前　型・入力範囲　初期値

シェーダ記述部分に
同名の変数が必要

```
float _WaveScale;
float _ReflDistort;
float _RefrDistort;
float4 _RefrColor;
sampler2D _ReflectionTex;
sampler2D _RefractionTex;
sampler2D _Fresnel;
sampler2D _BumpMap;
```



SubShader

- SubShader
  - オブジェクトを描画する際の一塊の処理（レンダリングパス）
  - タグを使ってタイミングを制御する
    - 半透明より不透明が先に描かれるように分かれている

- **Opaque**: ほとんどのシェーダー(Normal、Self Illuminated、Reflective、Terrain シェーダー)。
- **Transparent**: ほとんどの部分が透過なシェーダー(Transparent、パーティクル、フォント、Terrain 追加パスシェーダー)。
- **TransparentCutout**: マスキングされた透過シェーダー(Transparent Cutout、2 パス植生シェーダー)。
- **Background**: Skybox シェーダー。
- **Overlay**: ハロー、フレアシェーダー。
- **TreeOpaque**: Terrain エンジン Tree の樹皮。
- **TreeTransparentCutout**:Terrain エンジン Tree 葉っぱ。
- **TreeBillboard**: Terrain エンジンビルボードの 木。
- **Grass**: Terrain エンジンの 草。
- **GrassBillboard**: Terrain エンジンビルボードの 草。

```
Shader "Unlit/NewUnlitShader"
{
    Properties
    {...
    }
    SubShader
    {
        Tags { "RenderType"="Opaque" }
        LOD 100

        Pass
        {
            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            // make fog work
            #pragma multi_compile_fog

            #include "UnityCG.cginc"

            struct appdata
            {...
            };

            struct v2f
            {...
            };

            sampler2D _MainTex;
            float4 _MainTex_ST;

            v2f vert (appdata v)
            {...
            }

            fixed4 frag (v2f i) : SV_Target
            {...
            }
            ENDCG
        }
    }
}
```

Pass

- 一度の描画で用いる設定
- 複数の描画を続けて行うことができる
  - VRの両目の描画用に片目ずつ連続で描画したり
- 描画する際の状態の設定を行う

- AlphaToMask: alpha-to-coverage モードを設定します。
- Blend: アルファブレンディングを有効化と設定を行います。
- BlendOp: Blend コマンドで使用する操作を設定します。
- ColorMask: カラーチャンネルの書き込みマスクを設定します。
- Conservative: 慎重なラスタライズの有効化/無効化を行います。
- Cull: ポリゴンのカリングモードを設定します。
- Offset: ポリゴンの深度オフセットを設定します。
- Stencil: ステンシルテスト、およびステンシルバッファへの書き込み内容を設定します。
- ZClip: 深度クリップモードを設定します。
- ZTest: 深度テストのモードを設定します。
- ZWrite: 深度バッファの書き込みモードを設定します。

```
Shader "Unlit/NewUnlitShader"
{
    Properties
    {...
    }
    SubShader
    {
        Tags { "RenderType"="Opaque" }
        LOD 100

        Pass
        {
            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            // make fog work
            #pragma multi_compile_fog

            #include "UnityCG.cginc"

            struct appdata
            {...
            };

            struct v2f
            {...
            };

            sampler2D _MainTex;
            float4 _MainTex_ST;

            v2f vert (appdata v)
            {...
            }

            fixed4 frag (v2f i) : SV_Target
            {...
            }
            ENDCG
        }
    }
}
```

例:裏面の描画

```
12
13
14
15
16
17
```

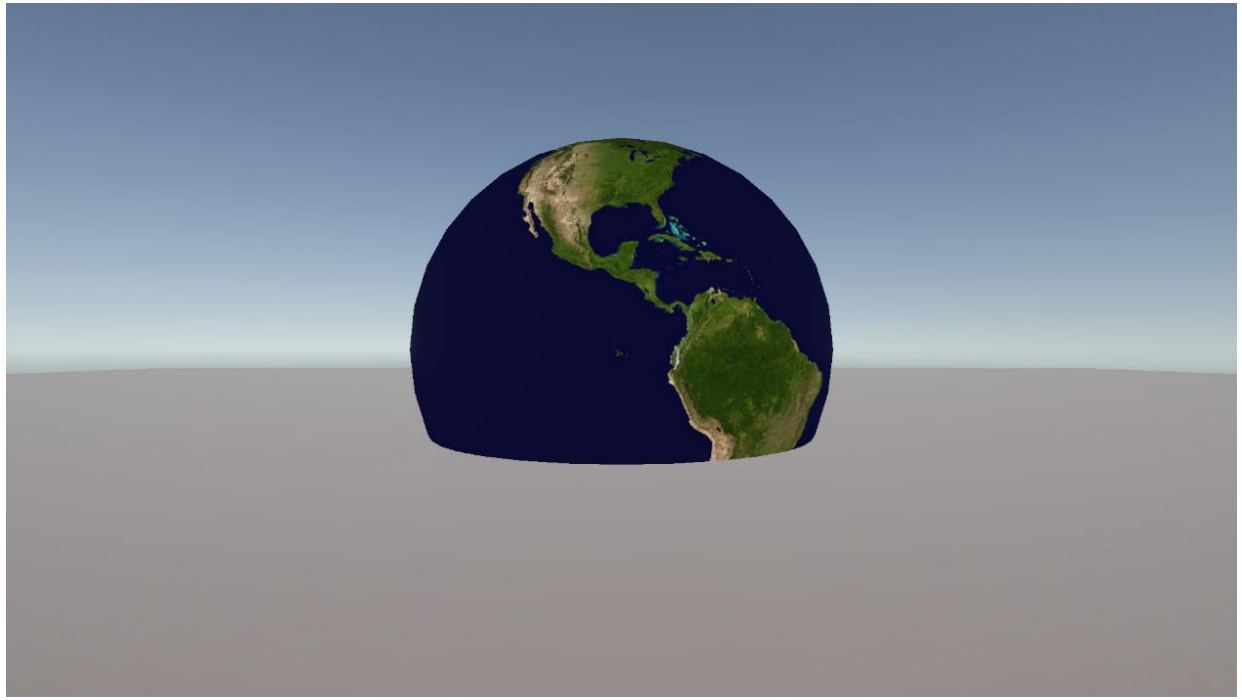
```
        Pass
        {
            Cull Back

            CGPROGRAM
            #pragma vertex vert
```

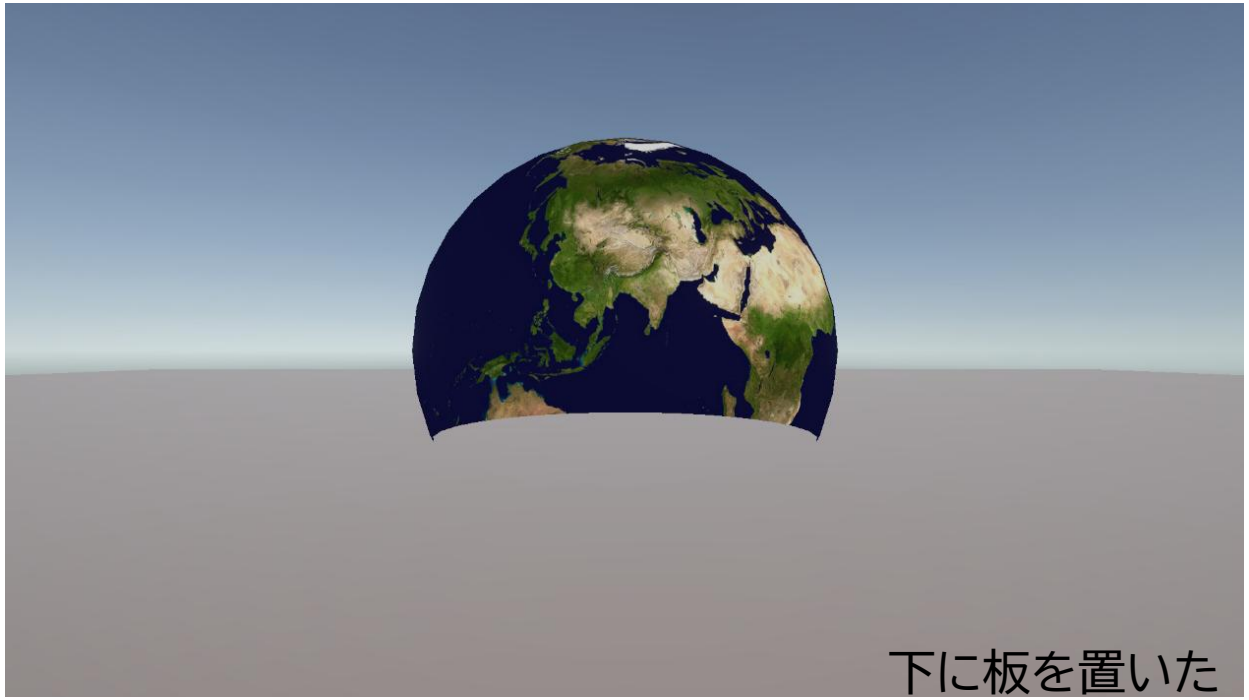
裏面を削除(デフォルト)

```
12
13
14
15
16
17
```

```
        Pass
        {
            Cull Front

            CGPROGRAM
            #pragma vertex vert
```

表面を削除





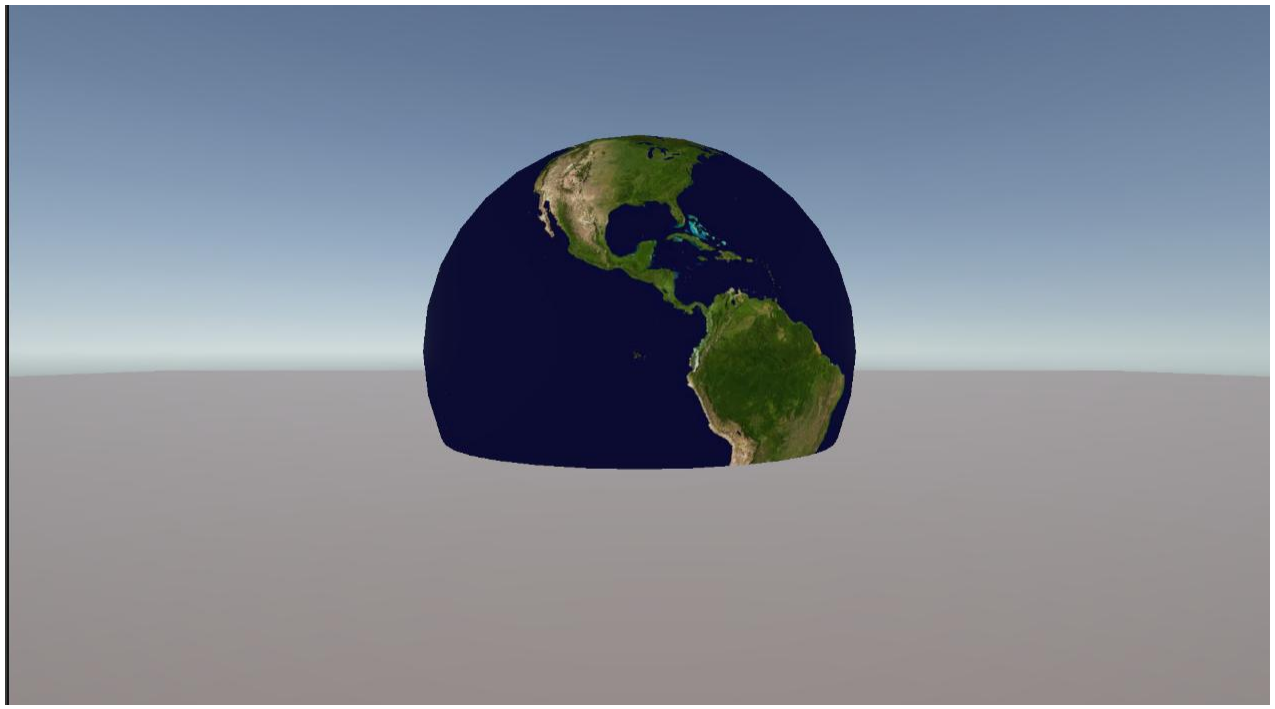
下に板を置いた

例:深度テストの反転

```
12
13
14
15
16
17
```

```
Pass
{
    ZTest LEqual

    CGPROGRAM
    #pragma vertex vert
```

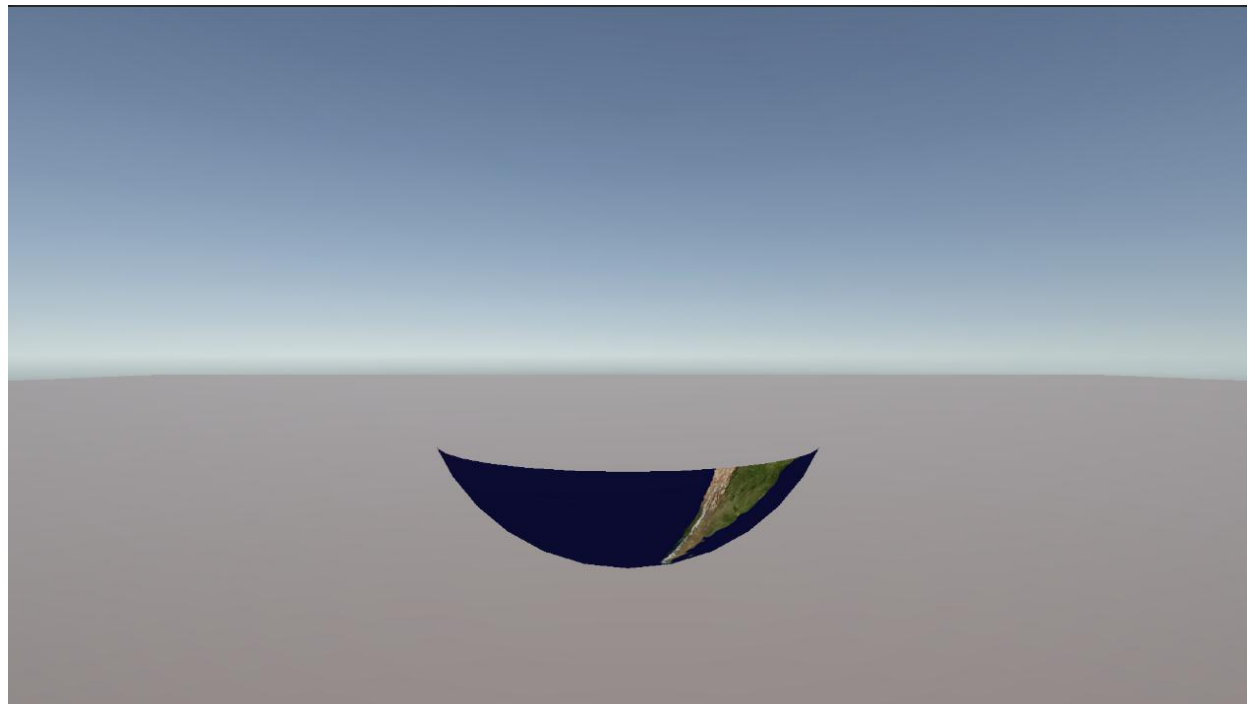
Less Equal（デフォルト）
手前にあるものを描画



```
12
13
14
15
16
17
```

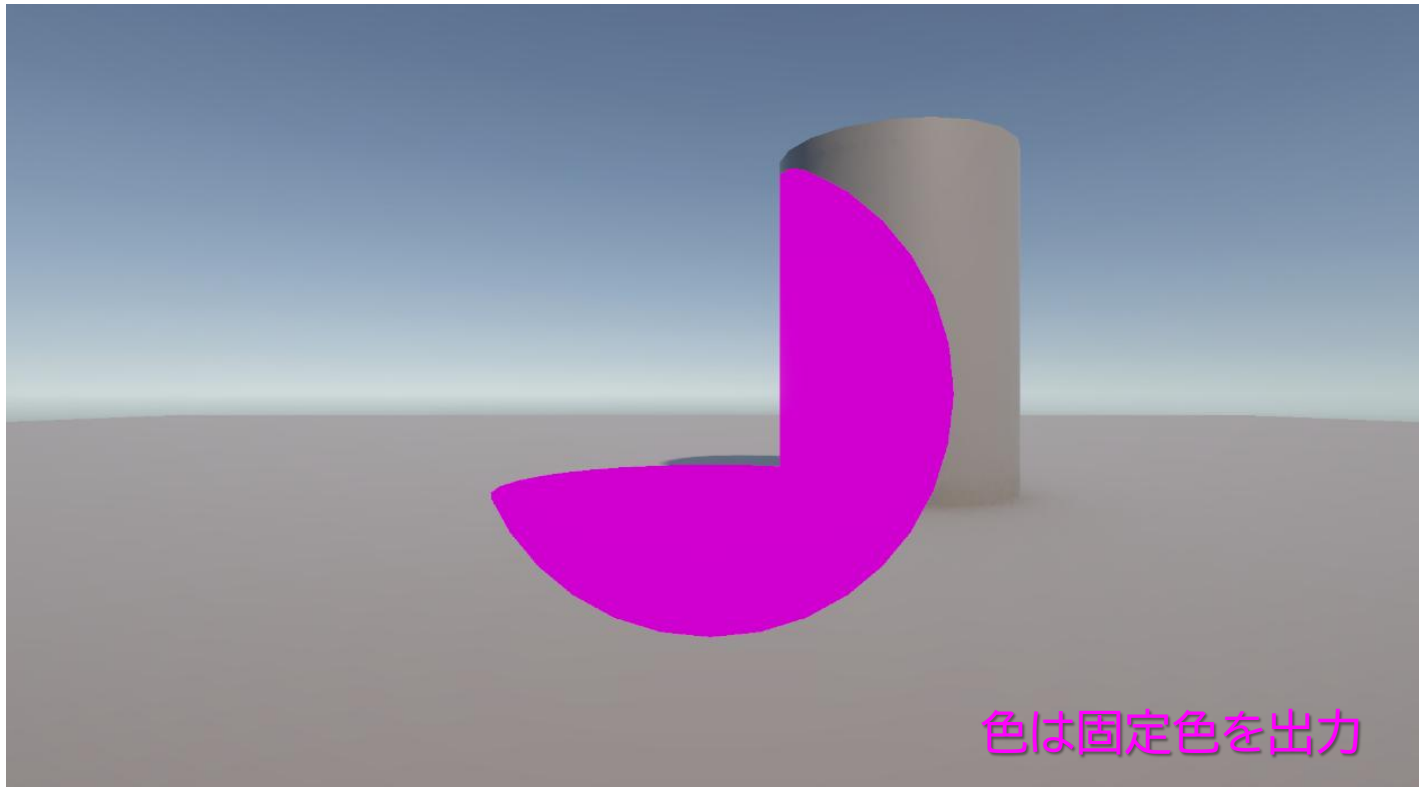
```
Pass
{
    ZTest Greater

    CGPROGRAM
    #pragma vertex vert
```

奥にあるものを描画



授業でいただいた質問：範囲に入った部分だけ色が変わる動的な変化の例が知りたいです

- 領域(球)に入った部分だけ色を変える
  - 「床は色を変えない」とかするにはもっと凝った処理が必要




```
12          Pass
13          {
14              ZTest Greater
15              Cull Front
16
17              CGPROGRAM
```

その他

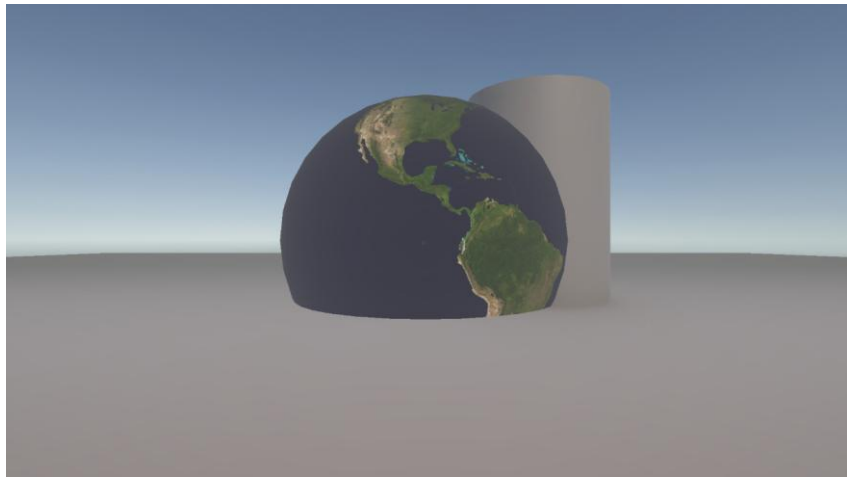
```
14          CGPROGRAM
15          #pragma vertex vert
16          #pragma fragment frag
17          // make fog work
18          #pragma multi_compile_fog
19
20          #include "UnityCG.cginc"
```

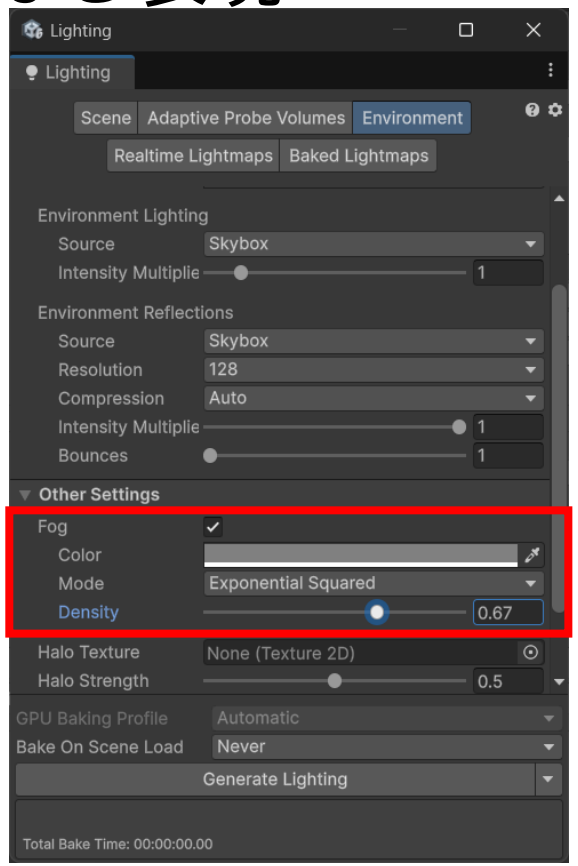
- #pragma: 色々設定
  - Vertex: 頂点シェーダの関数名
  - Fragment: ピクセルシェーダの関数名
  - multi_compile_fog: いろいろなフォグが使えるようにする

- #include “UnityCG.cginc”
  - いろいろな処理が書いてあってあまり詳しくなくても書ける
    - unity独自の関数・マクロの定義
  - 他にもいくつかのシェーダプログラムがある
    - Lighting.cginc: 照明計算に必要
    - TerrainEngine.cginc: 地形や植物用のシェーダのサポート

フォグ

- 遠ざかると白っぽくなる表現
  - お手軽空気遠近法
- 各種マクロは、**UnityCG.cginc** にある






```
15          #pragma vertex vert
16          #pragma fragment frag
17          // make fog work
18          #pragma multi_compile_fog
19
20          #include "UnityCG.cginc"
21
22          struct appdata
23          {
24              float4 vertex : POSITION;
25              float2 uv : TEXCOORD0;
26          };
27
28          struct v2f
29          {
30              float2 uv : TEXCOORD0;
31              UNITY_FOG_COORDS(1)
32              float4 vertex : SV_POSITION;
33          };
34
35          sampler2D _MainTex;
36          float4 _MainTex_ST;
37
38          v2f vert (appdata v)
39          {
40              v2f o;
41              o.vertex = UnityObjectToClipPos(v.vertex);
42              o.uv = TRANSFORM_TEX(v.uv, _MainTex);
43              UNITY_TRANSFER_FOG(o,o.vertex);
44              return o;
45          }
46
47          fixed4 frag (v2f i) : SV_Target
48          {
49              // sample the texture
50              fixed4 col = tex2D(_MainTex, i.uv);
51              // apply fog
52              UNITY_APPLY_FOG(i.fogCoord, col);
53              return col;
54          }
55          ENDCG
```

[Window]-[Rendering]-[Lighting]

目次

- シェーダ入門
- 簡単なシェーダ
- シェーダを読む
- HLSL入門

シェーダ言語の基本

- 入力と引数を構造体で受け取る
  - 一変数であれば変数でも可
- 出力は返り値として返す
  - return！
  - 返すべき出力はシェーダごとに決まっている
    ＋自分で選べる（構造体のメンバーとして）
    - 頂点シェーダ：射影座標系での位置
      - ピクセルシェーダで使う頂点毎のデータを渡す
    - ピクセルシェーダ：色
      - 透明度は色のα成分
      - 深度を書き換えることもできる
- グローバル変数は読み込みのみ
  - 定数バッファとしてCPUからの全体パラメータを受け取る

```
// 頂点シェーダーへの入力頂点構造体
struct VSInput
{
    float4 pos : POSITION;
};

// 頂点シェーダーの出力
struct VSOutput
{
    float4 pos : SV_POSITION;
};

// 頂点シェーダー
// 1. 引数は変換前の頂点情報
// 2. 戻り値は変換後の頂点情報
VSOutput VSMain(VSInput In)
{
    VSOutput vsOut = (VSOutput)0;

    // 入力された頂点座標を出力データに代入する
    vsOut.pos = In.pos;

    return vsOut;
}

// ピクセルシェーダー
float4 PSMain(VSOutput vsOut) : SV_Target0
{
    // 赤色を出力している
    return float4(1.0f, 0.0f , 0.0f, 1.0f);
}
```

組み込み関数

- 関数一覧
  - https://docs.microsoft.com/ja-jp/windows/win32/direct3dhlsl/dx-graphics-hlsl-intrinsic-functions
    - 「abs」が「絶対」とか、名前が勝手に日本語化されるので注意
- 種類：

  シェーダグラフでの基本的なノード
  - 簡単な操作：絶対値、クランプ、小数部...
  - 算術関数：sqrt, exp, dot, cross, …
  - 勾配命令（偏微分）：ddx, ddy
  - ノイズ関数: noise
  - テクスチャ読み込み: tex2D, …

組み込み関数

2021/09/15 •

次の表に、HLSL で使用できる組み込み関数を示します。 各関数には簡単な説明と、入力引数と戻り値の型に関する詳細が含まれている参照ページへのリンクがあります。

名前	説明	最小シェーダーモデル
取り消し	実行中の現在の描画またはディスパッチ呼び出しを終了します。	4
絶対	絶対値 (コンポーネントごと)。	1[1]
acos	X の各要素のアークコサインを返します。	1[1]
すべての	X のすべてのコンポーネントが0以外であるかどうかをテストします。	1[1]
AllMemoryBarrier	すべてのメモリアクセスが完了するまで、グループ内のすべてのスレッドの実行をブロックします。	5
AllMemoryBarrierWithGroupSync	すべてのメモリアクセスが完了し、グループ内のすべてのスレッドがこの呼び出しに到達するまで、グループ内のすべてのスレッドの実行をブロックします。	5
いつ	X のコンポーネントが0以外であるかどうかをテストします。	1[1]
asdouble	キャスト値を double に再解釈します。	5
asfloat	入力の型を float に変換します。	4
サイン	X の各コンポーネントのアークサインを返します。	1[1]
asint	入力の型を整数に変換します。	4
asuint	64ビット型のビットパターンを uint に再解釈します。	5
asuint	入力の型を符号なし整数に変換します。	4
atan	X のアークタンジェントを返します。	1[1]
atan2	2つの値 (x, y) のアークタンジェントを返します。	1[1]
ceil	X 以上の最小の整数を返します。	1[1]

型

- 基本型　精度が十分で小さなサイズ(速い)がよい
  - float(32bit)
  - half(16bit)
  - fixed(-2～2, 11bit)
  - int
- ベクトル
  - float4, float4x4等
    - .xyzw, .rgba で成分指定
- サンプラー(テクスチャ)
  - sampler2D
  - samplerCUBE
  - sampler3D

- セマンティック
  - シェーダの入出力のデータの意味を与える

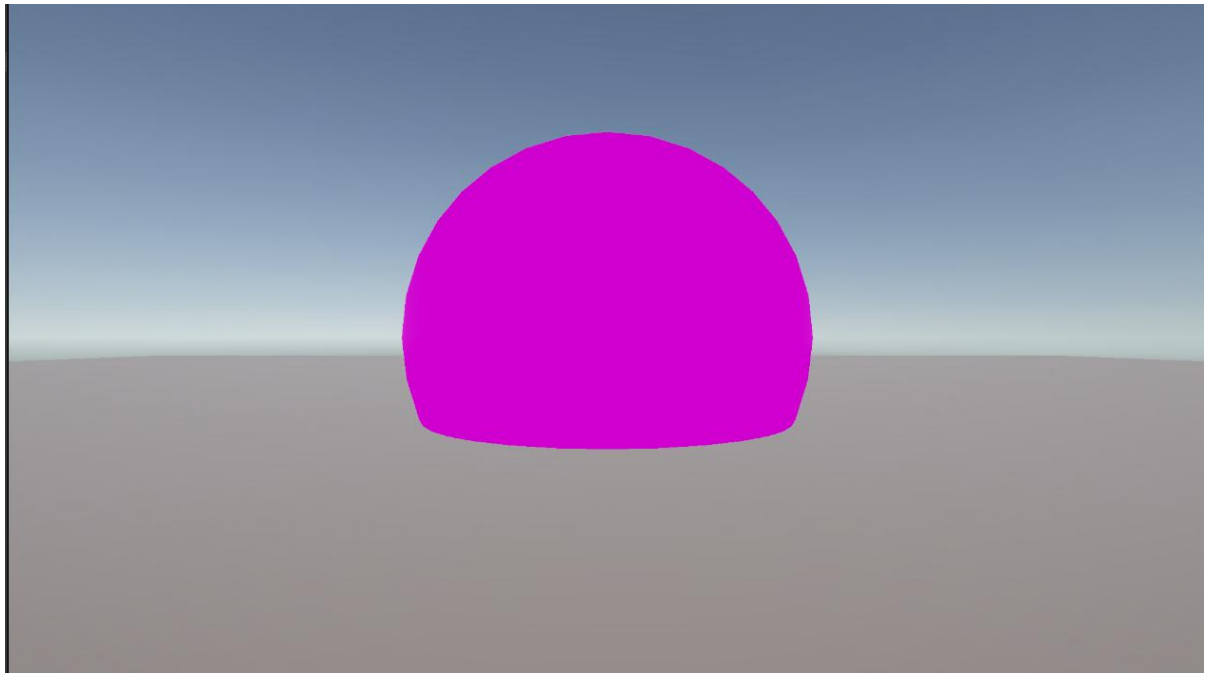
- POSITION は頂点位置、一般的には float3 か float4 です。
- NORMAL は通常の頂点で、一般的には float3 です。
- TEXCOORD0 は、第 1 の UV 座標で、一般的に、float2,float3,float4 です。
- TEXCOORD1,TEXCOORD2,TEXCOORD3 は、それぞれ第 2、第 3、第 4 の UV 座標です。
- TANGENT は、（ノーマルマッピングで使用される）接線ベクトルで、一般的には、 float4 です。
- COLOR は、頂点ごとの色で、一般的には、 float4 です。

```
// テクスチャをサンプリング
fixed4 col = tex2D(_MainTex, i.uv);
```

サンプラ　テクスチャ座標

シェーダの例:固定色

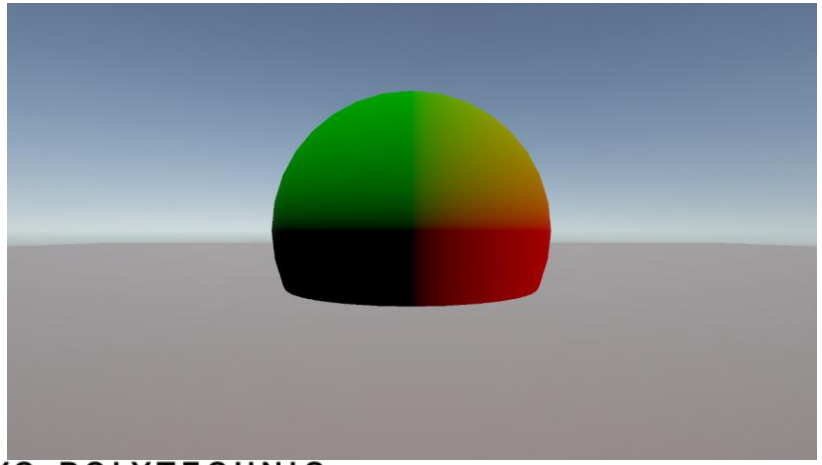
- 定数として出力値を与える



```
50          fixed4 frag (v2f i) : SV_Target
51          {
52              // 固定色
53              fixed4 col = fixed4(1, 0, 1, 1);// 赤、緑、青、不透明度→ピンク
54              // apply fog
55              UNITY_APPLY_FOG(i.fogCoord, col);
56              return col;
57          }
```

シェーダの例:位置の可視化

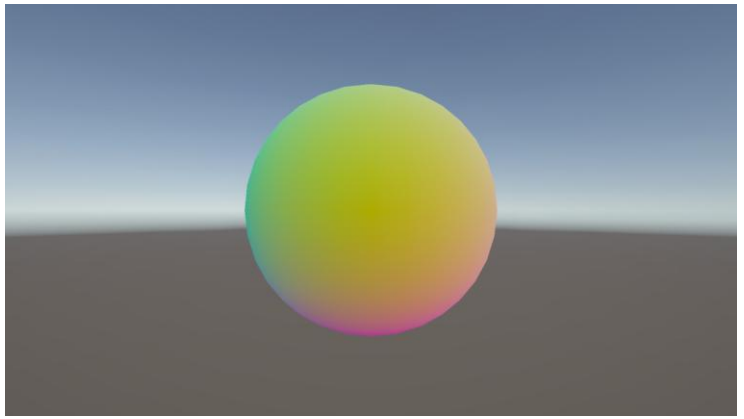
- モデル空間での位置について、(x, y, z) = (赤、緑、青)で可視化
  - 負の値は0(黒)
- 一般的な変数はテクスチャ座標として頂点シェーダからピクセルシェーダに渡す



```
28          struct v2f
29          {
30              float2 uv : TEXCOORD0;
31              UNITY_FOG_COORDS(1)
32              float4 vertex : SV_POSITION;
33              float3 pos : TEXCOORD1;
34          };
35
36          sampler2D _MainTex;
37          float4 _MainTex_ST;
38
39          v2f vert (appdata v)
40          {
41              v2f o;
42              o.vertex = UnityObjectToClipPos(v.vertex);
43              o.uv = TRANSFORM_TEX(v.uv, _MainTex);
44              o.pos = v.vertex.xyz;
45              UNITY_TRANSFER_FOG(o,o.vertex);
46              return o;
47          }
48
49          fixed4 frag (v2f i) : SV_Target
50          {
51              // モデルの位置を色に変換
52              fixed4 col = fixed4(i.pos.xyz, 1);
53              // apply fog
54              UNITY_APPLY_FOG(i.fogCoord, col);
55              return col;
56          }
```

シェーダの例：法線の可視化

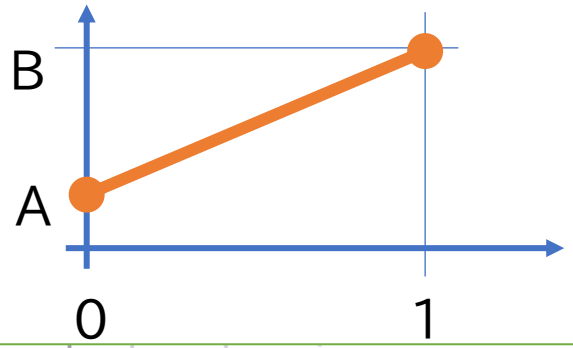
- 法線の[-1, +1]の範囲を[0,1]にして色にして可視化
- アプリケーションから、「NORMAL」という名前で頂点の法線情報が取れる
- 頂点シェーダからピクセルシェーダに渡す変数に「NORMAL」は予約済み（使える）

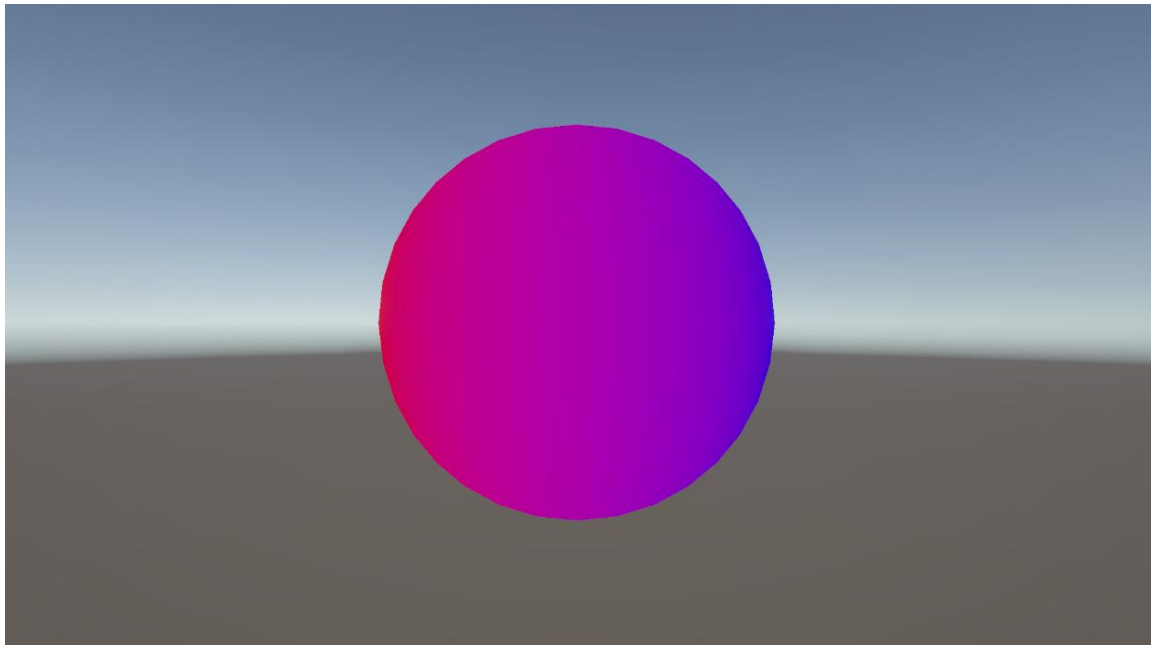


```
22          struct appdata
23          {
24              float4 vertex : POSITION;
25              float2 uv : TEXCOORD0;
26              float3 normal: NORMAL;
27          };
28
29          struct v2f
30          {
31              float2 uv : TEXCOORD0;
32              UNITY_FOG_COORDS(1)
33              float4 vertex : SV_POSITION;
34              float3 normal : NORMAL;
35          };
36
37          sampler2D _MainTex;
38          float4 _MainTex_ST;
39
40          v2f vert (appdata v)
41          {
42              v2f o;
43              o.vertex = UnityObjectToClipPos(v.vertex);
44              o.uv = TRANSFORM_TEX(v.uv, _MainTex);
45              o.normal = v.normal.xyz;
46              UNITY_TRANSFER_FOG(o,o.vertex);
47              return o;
48          }
49
50          fixed4 frag (v2f i) : SV_Target
51          {
52              // 法線を正規化して色に変換
53              fixed4 col = fixed4(i.normal.xyz * 0.5 + 0.5, 1);
54              // apply fog
55              UNITY_APPLY_FOG(i.fogCoord, col);
56              return col;
57          }
```

シェーダの例:グラデーション

- lerp(A, B, t): 内挿
  - lerp(A, B, t) = (1-t) * A + t * B





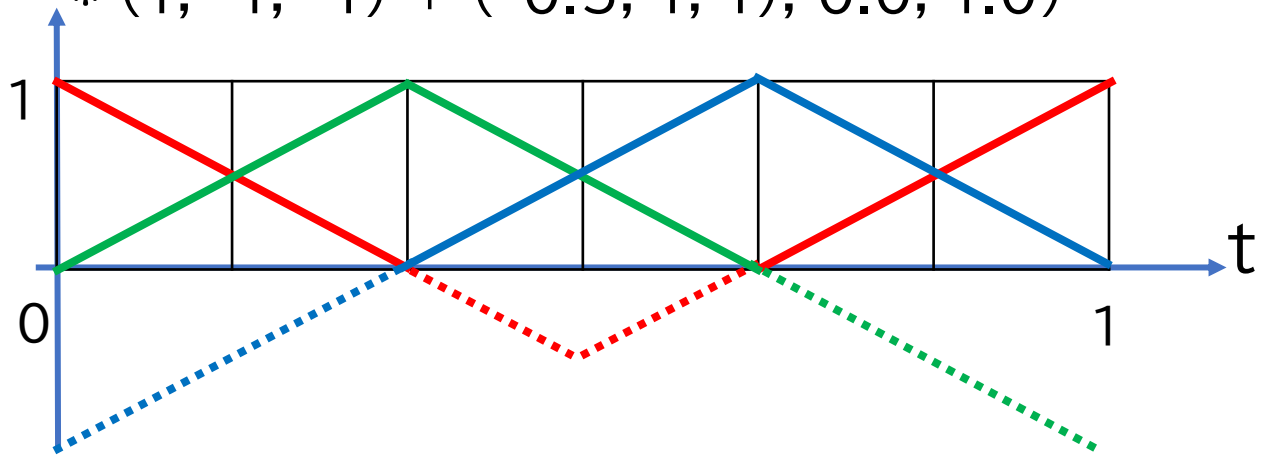
```
50          fixed4 frag (v2f i) : SV_Target
51          {
52              // 赤から青へのグラデーション
53              fixed4 col = lerp(fixed4(1,0,0,1), fixed4(0,0,1,1), i.normal.x*0.5+0.5);
54              // apply fog
55              UNITY_APPLY_FOG(i.fogCoord, col);
56              return col;
57          }
```

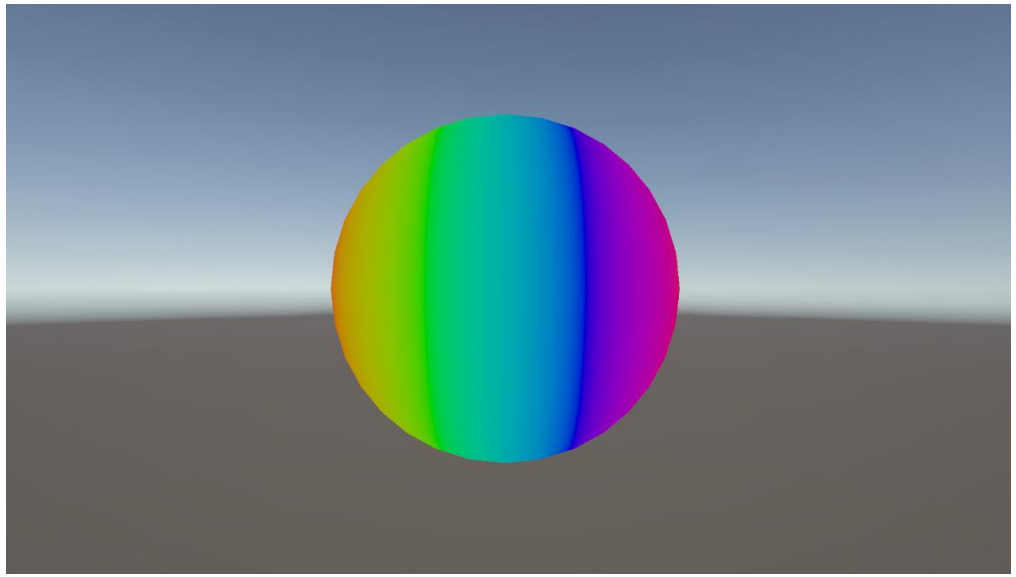
赤　　青

前頁のプログラムを修正

シェーダの例:虹の模様

- clamp(abs(3.0 * t + (-1.5, -1, -2)) * (1, -1, -1) + (-0.5, 1, 1), 0.0, 1.0)





```
50                  fixed4 frag (v2f i) : SV_Target
51                  {
52                      // 虹のグラデーション
53                      float t = i.normal.x*0.5+0.5;
54                      fixed4 col = fixed4(saturate(abs(3*t+fixed3(-1.5,-1,-2))*fixed3(1,-1,-1)+fixed3(-0.5,1,1)), 1);
55                      // apply fog
56                      UNITY_APPLY_FOG(i.fogCoord, col);
57                      return col;
58                  }
```

どのような命令があるか把握する

- MicrosoftのドキュメントのHLSLの「組み込み関数」を読む
  - https://learn.microsoft.com/ja-jp/windows/win32/direct3dhlsl/dx-graphics-hlsl-intrinsic-functions
  - 先ずは、最小シェーダモデル「1」の関数を読む

シェーダモデル1の関数

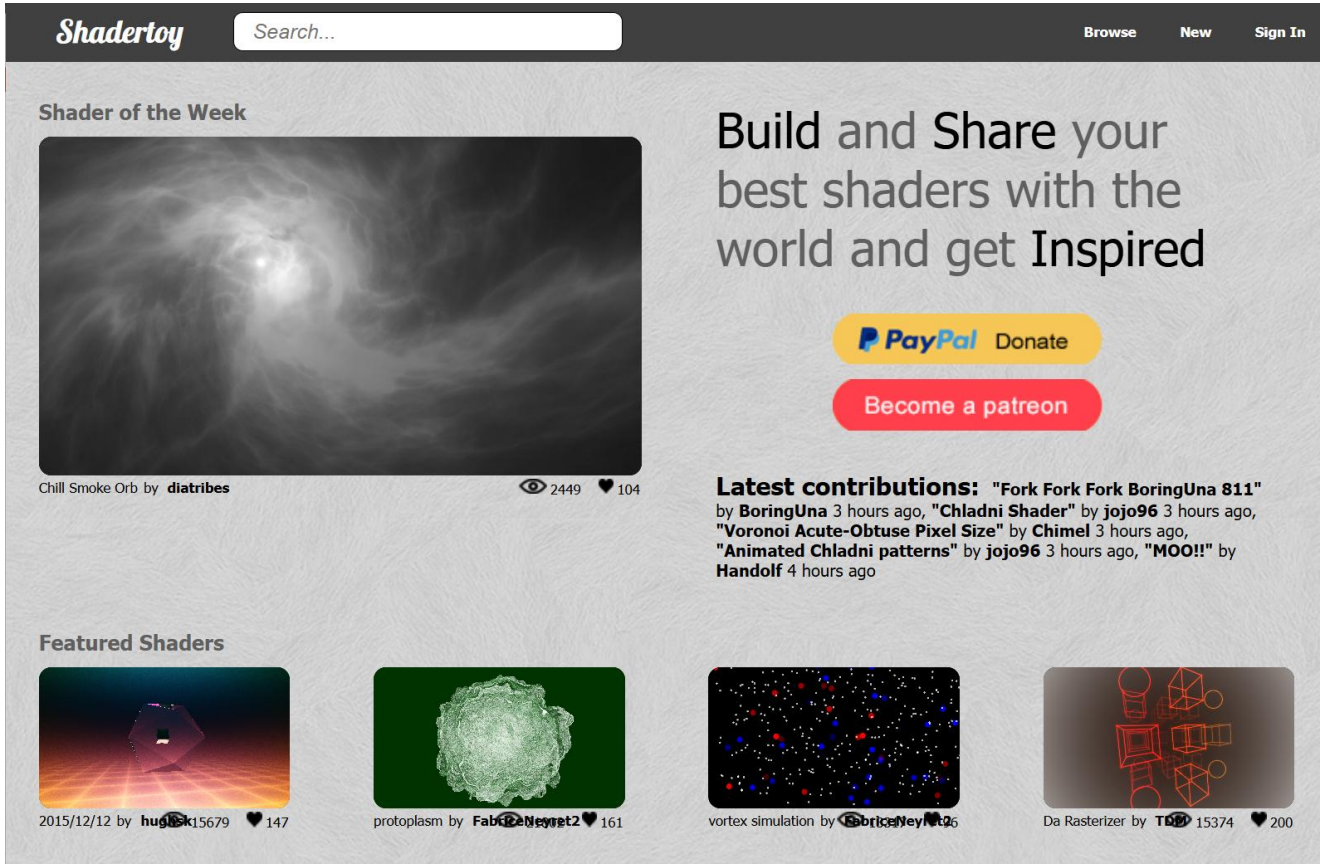
abs	絶対値（コンポーネントごと）。
acos	x の各コンポーネントのアークコサインを返します。
all	x のすべてのコンポーネントが 0 以外であるかどうかをテストします。
any	x のいずれかのコンポーネントが 0 以外であるかどうかをテストします。
asin	x の各コンポーネントのアークサインを返します。
atan	x のアークタンジェントを返します。
atan2	2 つの値 (x,y) のアークタンジェントを返します。
ceil	x 以上の最小の整数を返します。
clamp	x を範囲 [min, max] にクランプします。
clip	x のいずれかのコンポーネントが 0 未満の場合、現在のピクセルを破棄します。
cos	x のコサインを返します。
cosh	x の双曲線コサインを返します。
cross	2 つの 3D ベクトルの外積を返します。
D3DCOLORtoUBYTE4	一部のハードウェアでの UBYTE4 サポートの欠如を補うために、4D ベクトル x のコンポーネントのスウィズルおよびスケーリングを行います。
degrees	x をラジアンから度に変換します。
determinant	平方行列 m の行列式を返します。
distance	2 点間の距離を返します。
dot	2 つのベクトルのドット積を返します。
exp	底が e の指数を返します。
exp2	底が 2 の指数（コンポーネントごと）。
faceforward	-n * sign(dot(i, ng)) を返します。
floor	x 以下の最大の整数を返します。
fmod	x/y の浮動小数点の剰余を返します。
frac	x の小数部を返します。
isfinite	x が有限の場合は true、それ以外の場合は false を返します。
isinf	x が +INF または -INF の場合は true、それ以外の場合は false を返します。
isnan	x が NAN または QNAN の場合は true、それ以外の場合は false を返します。
ldexp	x * 2exp を返します
length	ベクトル v の長さを返します。
lerp	x + s(y - x) を返します。

lit	照明ベクトル（アンビエント、拡散、反射、1）を返します。
log	x の e を底とする対数を返します。
log10	x の 10 を底とする対数を返します。
log2	x の 2 を底とする対数を返します。
max	x と y の大きい方を選択します。
min	x と y の小さい方を選択します。
modf	値 x を小数部と整数部に分割します。
mul	x と y を使用して行列の乗算を実行します。
noise	パーリン ノイズ アルゴリズムを使用してランダムな値を生成します。
normalize	正規化されたベクトルを返します。
pow	x^y を返します。
radians	x を度からラジアンに変換します。
reflect	反射ベクトルを返します。
refract	屈折ベクトルを返します。
round	x を最も近い整数に丸めます
rsqrt	1 / sqrt(x) を返します
saturate	x を範囲 [0, 1] にクランプします
sign	x の符号を計算します。
sin	x のサインを返します
sincos	x のサインとコサインを返します。
sinh	x の双曲サインを返します
smoothstep	0 から 1 の間の滑らかなエルミート補間を返します。
sqrt	平方根（コンポーネントごと）
step	(x >= a) ? 1 : 0 を返します
tan	x のタンジェントを返します
tanh	x の双曲タンジェントを返します
tex1D(s, t)	1D テクスチャ参照。
tex2D(s, t)	2D テクスチャ参照。
tex3D(s, t)	3D テクスチャ参照。
texCUBE(s, t)	キューブ テクスチャ参照。
transpose	行列 m の転置を返します。
trunc	浮動小数点値を切り捨てて整数値にします。

何ができるか知るために

他の人がしていることを調べる

- Shadertoy
  - https://www.shadertoy.com/
  - ピクセルシェーダでできる表現
  - 各画素のスクリーンの位置だけからいろいろな事を実現する



まとめ

- シェーダ入門
  - シェーダについての概要を知る
- 簡単なシェーダ
  - オブジェクトを Unlit Shader で一から表示する
  - テクスチャを貼る
- シェーダを読む
  - デフォルトのコードを読んで理解する
- HLSL入門
  - シェーダで色を変えてみる