

プログラム ワークショップⅣ

(10) インスタンスング・ジオメトリシェーダ

今日やること

芝生を生やそう

- インスタンスング
- ジオメトリシェーダ



今日やること

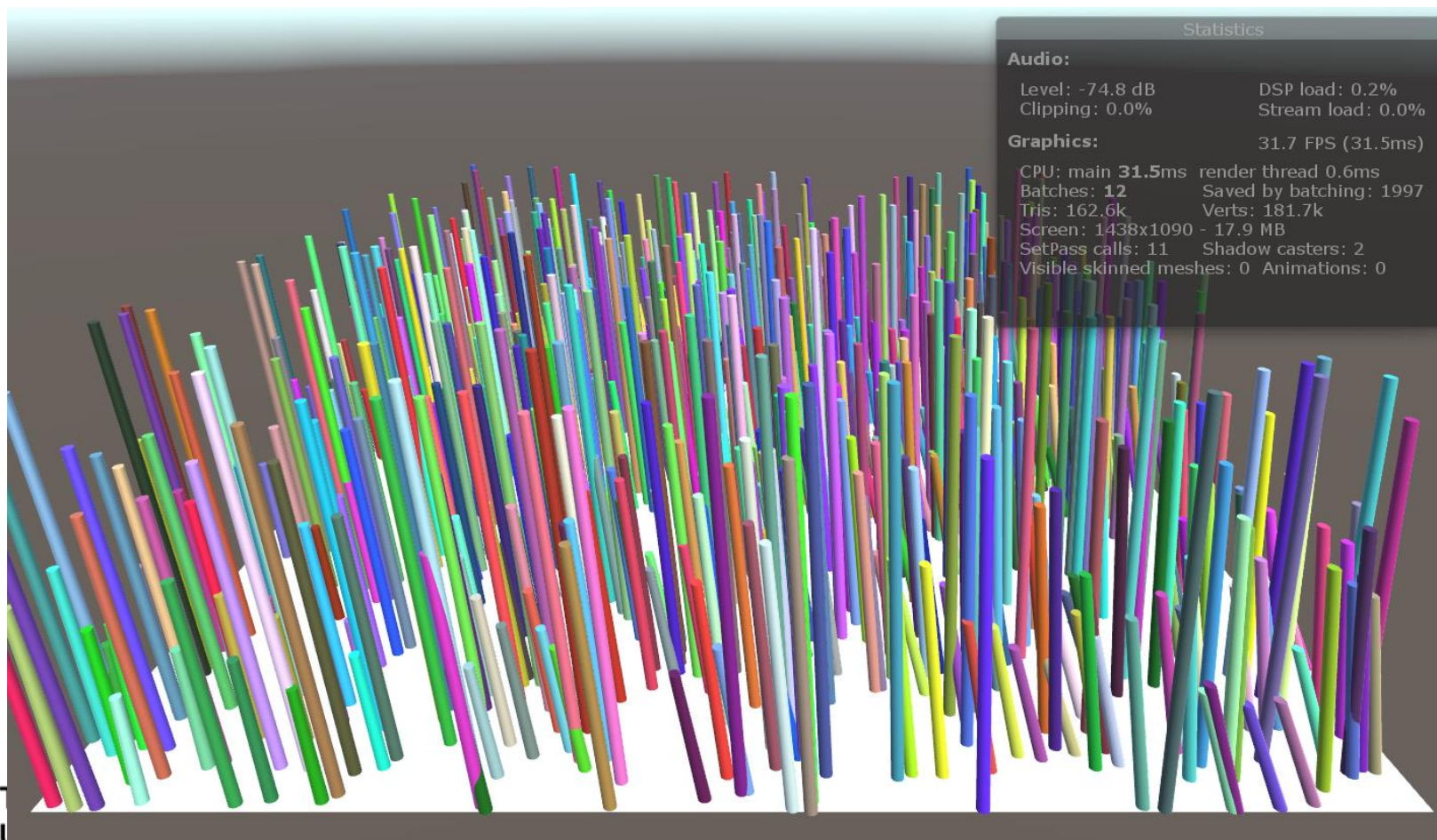
芝生を生やそう

- インスタンスング
- ジオメトリシェーダ



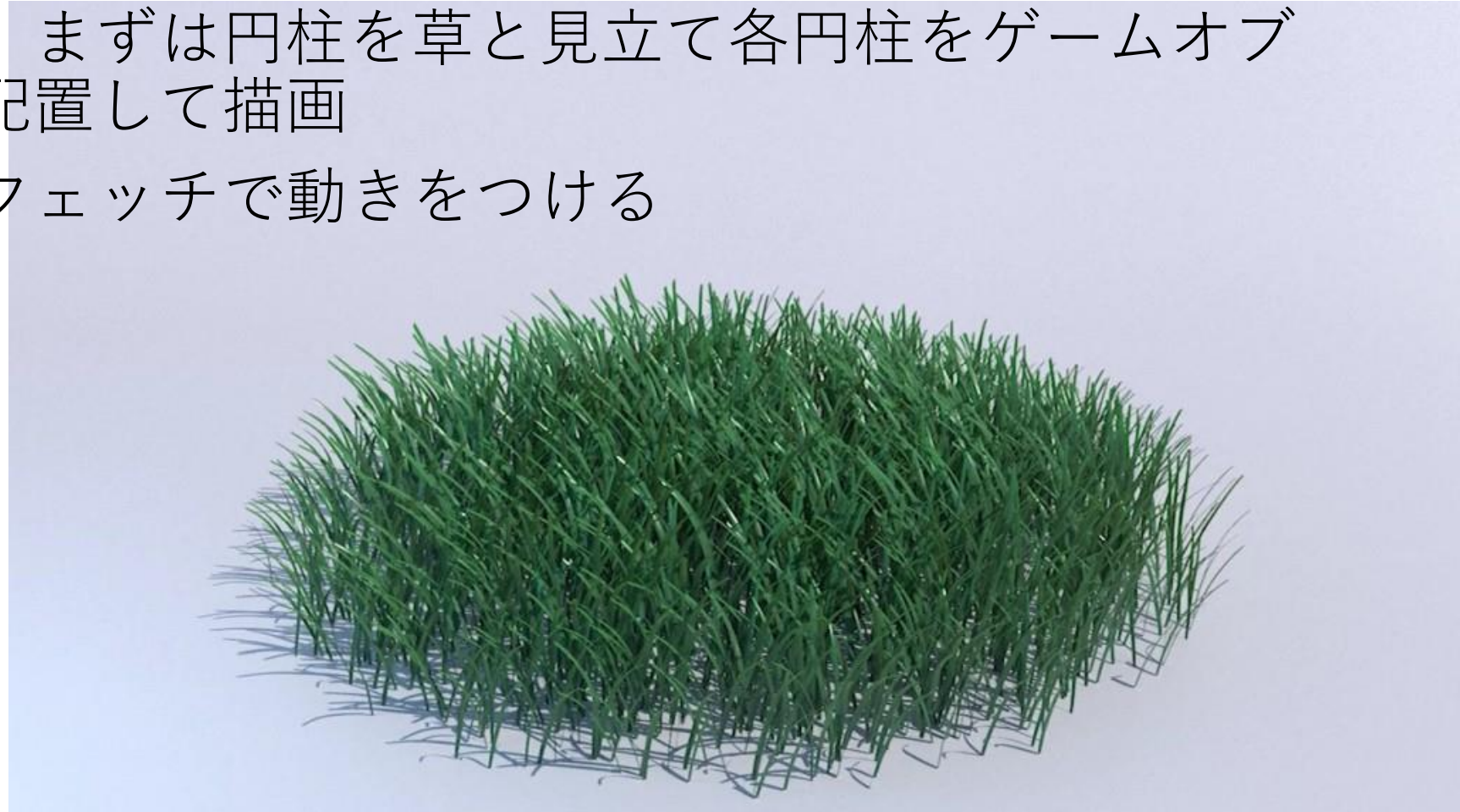
今回の結果

- 円柱を揺らす...



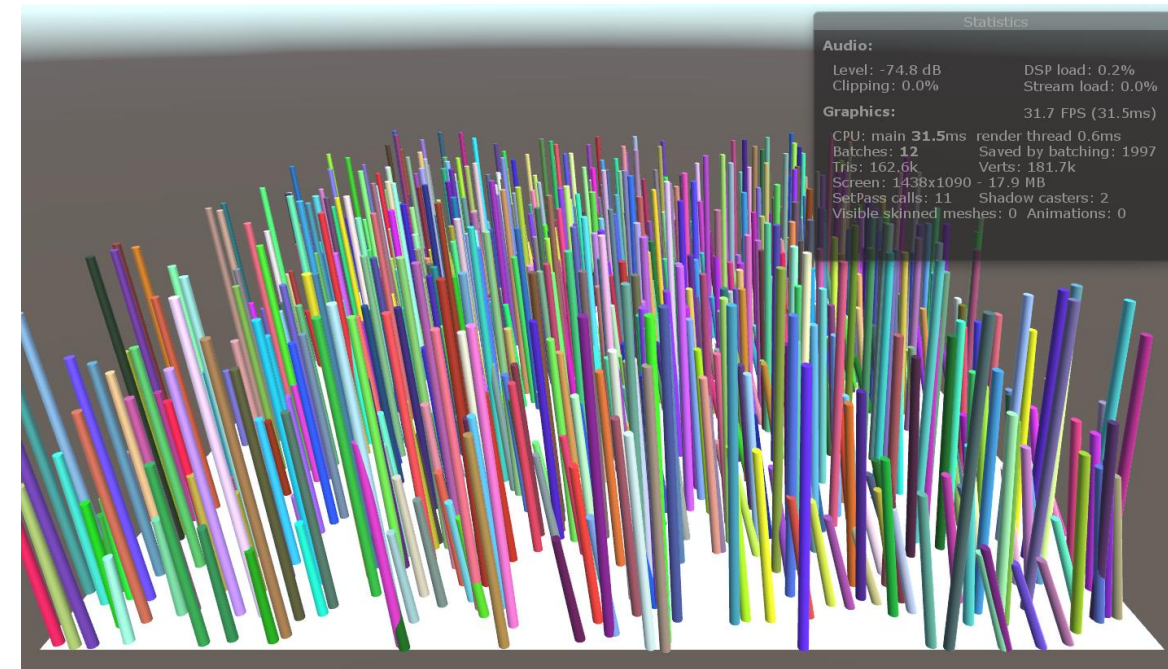
芝生をはやす方法

- いくつかあるが、まずは円柱を草と見立て各円柱をゲームオブジェクトとして配置して描画
- 頂点テクスチャフェッチで動きをつける



沢山のオブジェクトを出す

- 1000本の円柱を描画する
 - 沢山の描画命令を発行する必要がある？
 - 描画命令を増やすとCPU負荷が上がる
 - DX12やVulkanの登場の話になるが、今回はその話はしない
- GPUインスタンス
 - あるプレハブをまとめて大量に描画するのであれば、CPUの負荷を下げて描画できる



インスタンスング

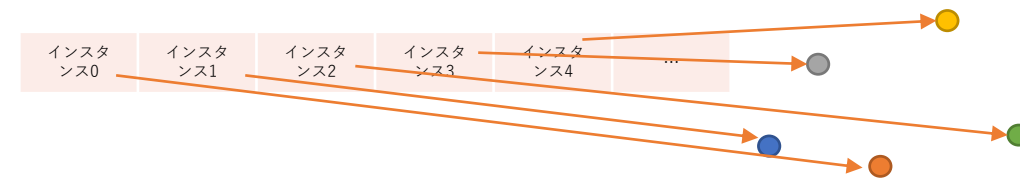
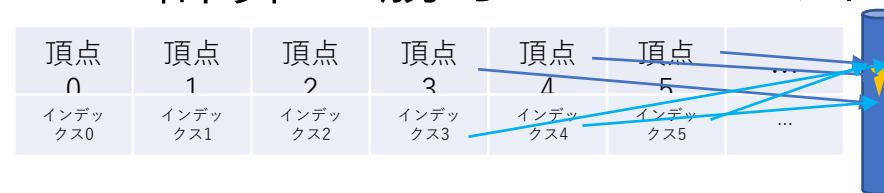
- 2種類のデータ配列を渡すと2重ループ計算を勝手にしてくれる

1. モデルデータ

- 頂点バッファ、インデックスバッファ

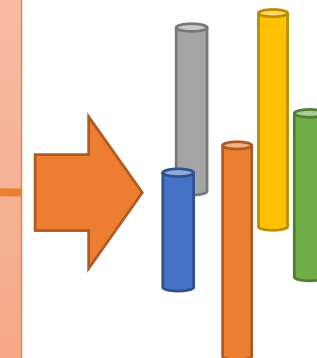
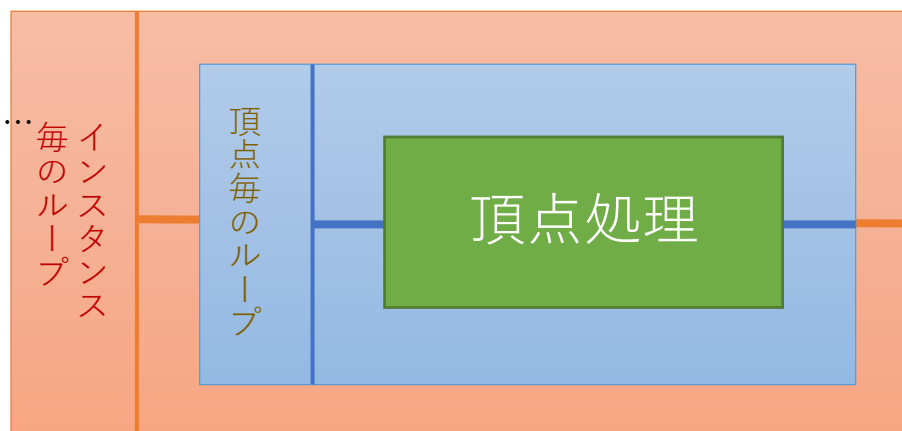
2. インスタンスデータ

- インスタンスの色、長さ
 - 今回はインスタンスの固有IDから算出
- インスタンスの位置



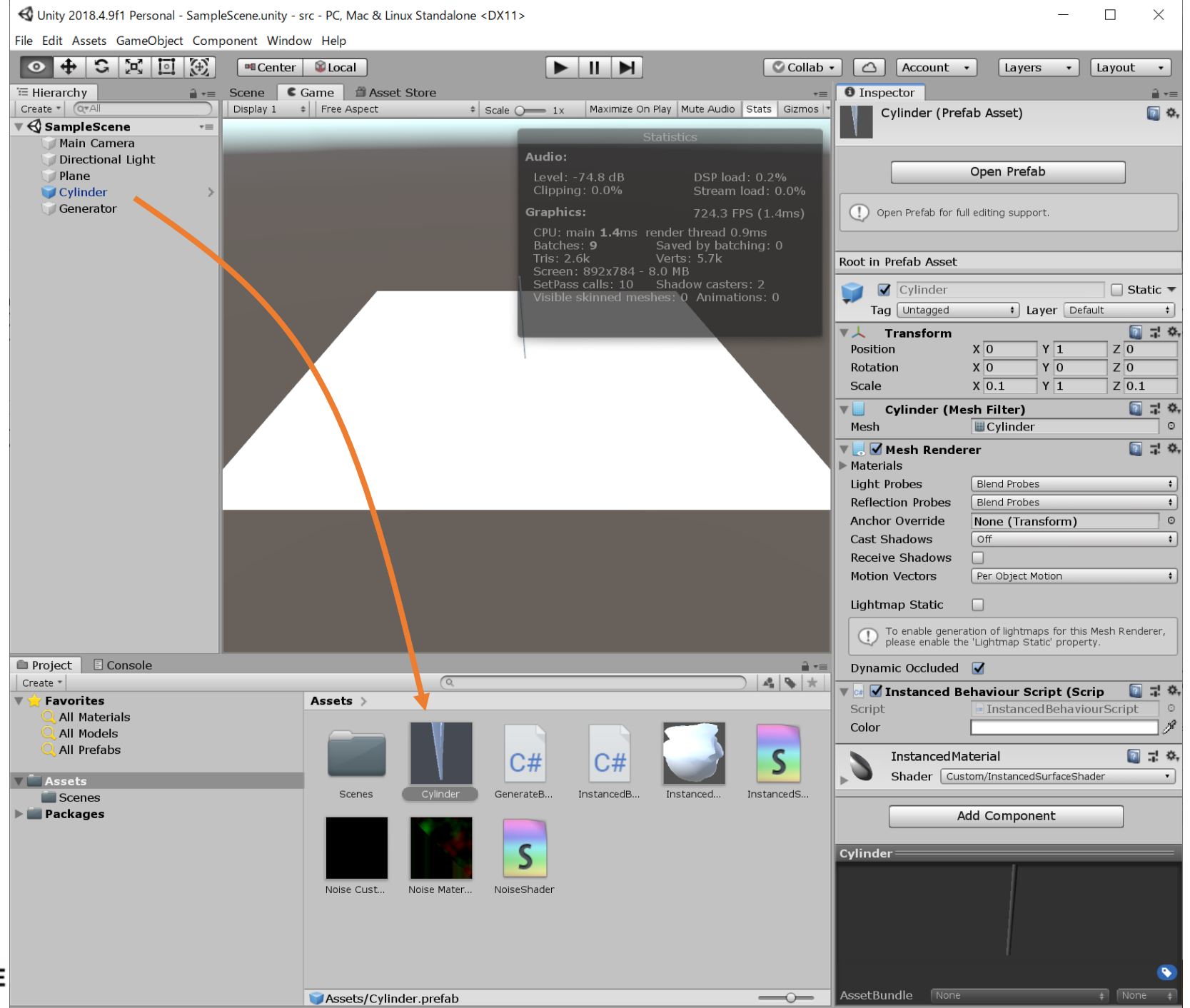
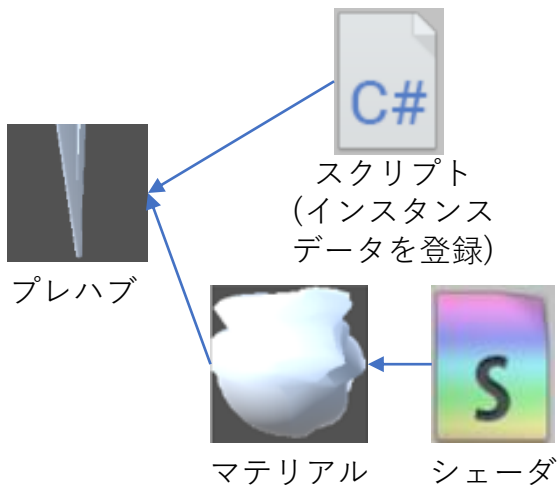
頂点 0	頂点 1	頂点 2	頂点 3	頂点 4	頂点 5	...
インデックス0	インデックス1	インデックス2	インデックス3	インデックス4	インデックス5	...
インスタンス0	インスタンス1	インスタンス2	インスタンス3	インスタンス4	...	

DrawIndexed...



プレハブ

- オブジェクトを作って
Projectに
D&D



シェーダ

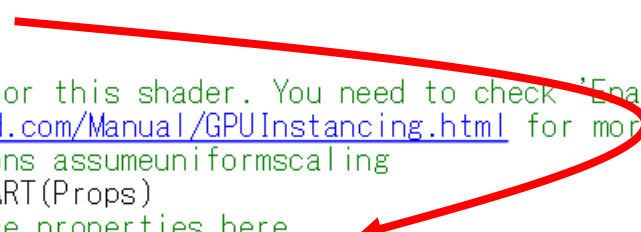
- 今回は、サーフェスシェーダを利用

```
1 Shader "Custom/InstancedSurfaceShader"
2 {
3     Properties
4     {
5         _Color ("Color", Color) = (1,1,1,1)
6         _MainTex ("Albedo (RGB)", 2D) = "white" {}
7         _Glossiness ("Smoothness", Range(0,1)) = 0.5
8         _Metallic ("Metallic", Range(0,1)) = 0.0
9         _DisplacementMap ("Displacement (RG)", 2D) = "white" {} // ☆追加
10    }
11    SubShader
12    {
13        Tags { "RenderType"="Opaque" }
14        LOD 200
15
16        CGPROGRAM
17        // Physically based Standard lighting model, and enable shadows on all light types
18        #pragma surface surf Standard fullforwardshadows
19        #pragma surface surf Standard fullforwardshadows vertex:vert // ☆頂点シェーダを変更する
20
21        // Use shader model 3.0 target, to get nicer looking lighting
22        #pragma target 3.0
23
24        sampler2D _MainTex;
25        sampler2D _DisplacementMap; // ☆追加
26
27        struct Input
28        {
29            float2 uv_MainTex;
30        };
31
32        half _Glossiness;
33        half _Metallic;
34        // fixed4 _Color; // ★削除
35
36        // Add instancing support for this shader. You need to check 'Enable Instancing' on materials that
37        // See https://docs.unity3d.com/Manual/GPUInstancing.html for more information about instancing.
38        // #pragma instancing_options assumeuniformscaling
39        UNITY_INSTANCING_BUFFER_START(Props)
40        // put more per-instance properties here
41        UNITY_DEFINE_INSTANCED_PROP(fixed4, _Color) // ★追加
42        UNITY_INSTANCING_BUFFER_END(Props)
43
44        // ☆頂点シェーダを追加
45        void vert(inout appdata full v) {
46            float2 d = tex2Dlod(_DisplacementMap, float4(v.vertex.xy, 0, 0)).xy; // 2次元ノイズの読み込み
47            v.vertex.xz += (v.vertex.y + 1.0) * d.xy * 10; // 高いほど大きく揺らす
48        }
49
50        void surf (Input IN, inout SurfaceOutputStandard o)
51        {
52            // Albedo comes from a texture tinted by color
53            fixed4 c = tex2D(_MainTex, IN.uv_MainTex) * _Color; // ★下記に変更
54            fixed4 c = tex2D(_MainTex, IN.uv_MainTex) * UNITY_ACCESS_INSTANCED_PROP(Props, _Color);
55            o.Albedo = c.rgb;
56            // Metallic and smoothness come from slider variables
57            o.Metallic = _Metallic;
58            o.Smoothness = _Glossiness;
59            o.Alpha = c.a;
60        }
61    } ENDCG
62    FallBack "Diffuse"
63 }
```

インスタンスリング用のシェーダの変更点

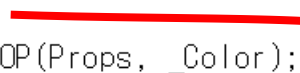
- マテリアルの色を変数からマクロで囲まれる変数へ
 - 変数定義

```
32 half _Glossiness;
33 half _Metallic;
34 // fixed4 _Color; // ★削除
35
36 // Add instancing support for this shader. You need to check 'Enable Instancing' on materials that use the shader.
37 // See https://docs.unity3d.com/Manual/GPUInstancing.html for more information about instancing.
38 // #pragma instancing_options assumeuniformscaling
39 UNITY_INSTANCING_BUFFER_START(Props)
40 // put more per-instance properties here
41 UNITY_DEFINE_INSTANCED_PROP(fixed4, _Color) // ★追加
42 UNITY_INSTANCING_BUFFER_END(Props)
```



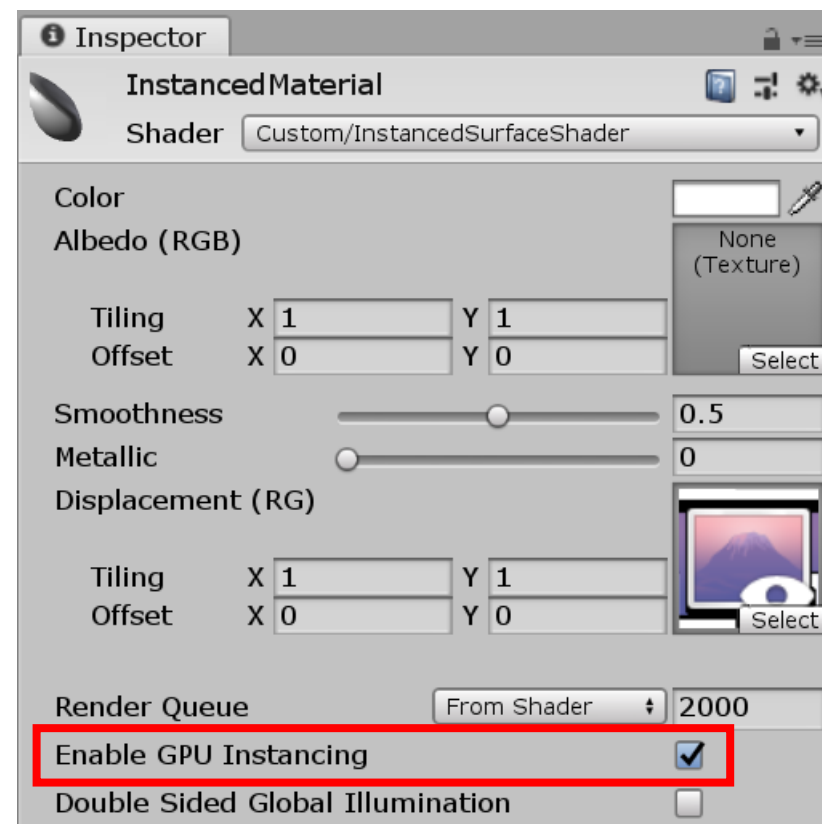
- シェーダ関数での利用

```
49 void surf (Input IN, inout SurfaceOutputStandard o)
50 {
51 // Albedo comes from a texture tinted by color
52 // fixed4 c = tex2D(_MainTex, IN.uv_MainTex) * _Color; // ★下記に変更
53 fixed4 c = tex2D(_MainTex, IN.uv_MainTex) * UNITY_ACCESS_INSTANCED_PROP(Props, _Color);
54 o.Albedo = c.rgb;
55 // Metallic and smoothness come from slider variables
56 o.Metallic = _Metallic;
57 o.Smoothness = _Glossiness;
58 o.Alpha = c.a;
59 }
```



マテリアル

- 「Enable GPU Instancing」を有効にする



スクリプト

- MaterialPropertyBlock (マテリアルのブロック要素)へパラメータを渡す
 - 今回は色は外部から設定
 - 生成時

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Assertions;
5
6  1 個の参照
7  public class InstancedBehaviourScript : MonoBehaviour
8  {
9      public Color color = Color.white;
10
11      0 個の参照
12      void Start()
13      {
14          var renderer = GetComponent<Renderer>();
15          Assert.IsNotNull(renderer);
16          MaterialPropertyBlock block = new MaterialPropertyBlock();
17          block.SetColor("_Color", color);
18          renderer.SetPropertyBlock(block);
19      }
20
21      0 個の参照
22      void Update()
23      {
24      }
25  }
```


揺らす

- 頂点テクスチャフェッチでテクスチャを読み込んで位置をずらす
 - テクスチャを定義

```
9  | | | _DisplacementMap("Displacement (RG)", 2D) = "white" {}// ☆追加
25  | | | sampler2D _DisplacementMap;// ☆追加
```

- 独自の頂点シェーダを使う宣言

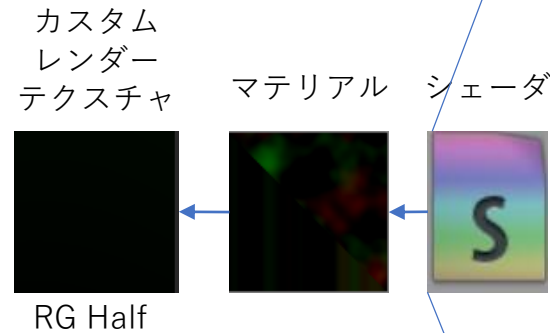
```
18  | | | // | #pragma surface surf Standard fullforwardshadows
19  | | | // | #pragma surface surf Standard fullforwardshadows vertex:vert// ☆頂点シェーダを変更する
```

- 頂点シェーダ

```
44  | | | // ☆頂点シェーダを追加
45  | | | void vert(inout appdata_full v) {
46  | | |     float4 vertex_w = mul(unity_ObjectToWorld, float4(0,v.vertex.y,0,1));// 円柱の中心軸の位置で読み込み
47  | | |     float2 d = tex2Dlod(_DisplacementMap, float4(vertex_w.xy, 0, 0)).xy;// 2次元ノイズの読み込み
48  | | |     v.vertex.xz += vertex_w.y * d.xy * 10;// 高いほど大きく揺らす
49  | | | }
```

揺らすテクスチャ

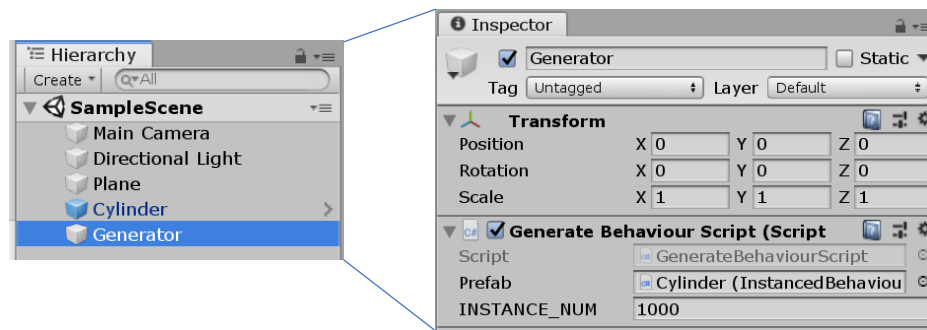
- 2次元の流れるperlin noise



```

1 Shader "Custom/NoiseShader"
2 {
3     Properties
4     {
5     }
6     SubShader
7     {
8         Cull Off
9         ZWrite Off
10        ZTest Always
11
12        Pass
13        {
14            CGPROGRAM
15            #include "UnityCustomRenderTexture.cginc"
16            #pragma vertex CustomRenderTextureVertexShader
17            #pragma fragment frag
18
19            #include "UnityCG.cginc"
20
21            fixed2 random2(float2 st) {
22                st = float2(dot(st, fixed2(127.1, 311.7)),
23                    dot(st, fixed2(269.5, 183.3)));
24                return -1.0 + 2.0 * frac(sin(st) * 43758.5453123);
25            }
26
27            float Noise(float2 st)
28            {
29                float2 p = floor(st);
30                float2 f = frac(st);
31                float2 u = f * f * (3.0 - 2.0 * f);
32
33                float2 v00 = random2(p + fixed2(0, 0));
34                float2 v10 = random2(p + fixed2(1, 0));
35                float2 v01 = random2(p + fixed2(0, 1));
36                float2 v11 = random2(p + fixed2(1, 1));
37
38                return lerp(
39                    lerp(dot(random2(p + float2(0.0, 0.0)), f - float2(0.0, 0.0)),
40                        dot(random2(p + float2(1.0, 0.0)), f - float2(1.0, 0.0)), u.x),
41                    lerp(dot(random2(p + float2(0.0, 1.0)), f - float2(0.0, 1.0)),
42                        dot(random2(p + float2(1.0, 1.0)), f - float2(1.0, 1.0)), u.x), u.y);
43            }
44
45            float Fbm(float2 texcoord)
46            {
47                float2 tc = texcoord * float2(.05, .05);
48                float time = _Time.y * 0.5;
49                float noise
50                    = Noise((tc + time) * 1.0)
51                    + Noise((tc + time) * 2.0) * 0.5
52                    + Noise((tc + time) * 4.0) * 0.25;
53                noise = noise / (1.0 + 0.5 + 0.25); // 正規化
54
55                return noise;
56            }
57
58            float2 frag (v2f_customrendertexture i) : SV_Target
59            {
60                return float2(
61                    Fbm(i.localTexcoord),
62                    Fbm(i.localTexcoord + float2(1000.0, 1000.0))); // 適当に遠い場所をサンプリング
63            }
64        }
65    }
66 }
67 
```


インスタンスの作成



```
11 void generate()
12 {
13     for (int i = 0; i < INSTANCE_NUM; i++)
14     {
15         var inst = Instantiate(prefab);
16         検索しやすいようにこのオブジェクトの子とする
17         inst.transform.parent = gameObject.transform;
18         長さや位置をランダムに設定
19         // 長さ
20         float y = Random.Range(0.3f, 1.5f);
21         inst.transform.localScale =
22             new Vector3(.1f, y, .1f);
23
24         // 位置
25         inst.transform.position = new Vector3(
26             Random.Range(-5.0f, 5.0f),
27             y,
28             Random.Range(-5.0f, 5.0f));
29         色はバラバラにしてみる
30         // 色変え
31         inst.color.r = Random.Range(0.0f, 1.0f);
32         inst.color.g = Random.Range(0.0f, 1.0f);
33         inst.color.b = Random.Range(0.0f, 1.0f);
34     }
35 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 0 個の参照
6 public class GenerateBehaviourScript : MonoBehaviour
7 {
8     public InstancedBehaviourScript prefab;
9     public int INSTANCE_NUM = 1000;
10    int last_instance_num = 1000;
11
12    2 個の参照
13    void generate()...
14
15    0 個の参照
16    void Start()
17    {
18        generate();
19    }
20
21    0 個の参照
22    void Update()
23    {
24    }
25
26    0 個の参照
27    void OnValidate()
28    {
29        if (last_instance_num != INSTANCE_NUM)
30        {
31            last_instance_num = INSTANCE_NUM;
32
33            // 現在のオブジェクトを削除
34            for (int i = 0; i < gameObject.transform.childCount; i++)
35            {
36                Destroy(gameObject.transform.GetChild(i).gameObject);
37            }
38
39            // 生成
40            generate();
41        }
42    }
43
44    0 個の参照
45    void OnDestroy()
46    {
47    }
48
49    0 個の参照
50    void OnDisable()
51    {
52    }
53
54    0 個の参照
55    void OnEnable()
56    {
57    }
58
59    0 個の参照
60    void OnGUI()
61    {
62    }
63 }
```

エディタでインスタンス数に変更された際に
現在のオブジェクトを消して、新たに生成

やってみよう

- メッシュを差し替えて草原にしよう
 - 見た目が良くなかったらシェーダを書き替えてみよう
 - 伸びている部分を分割して、やさしく揺らしてみよう

今日やること

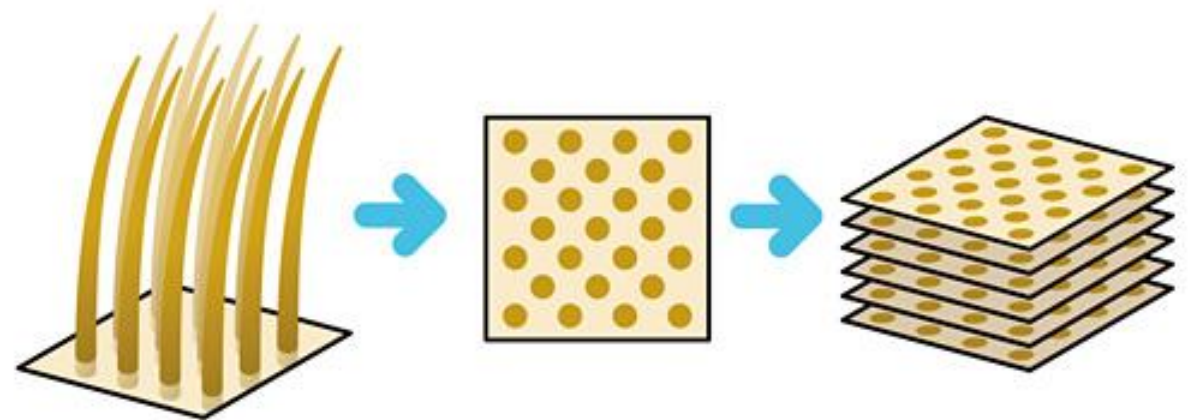
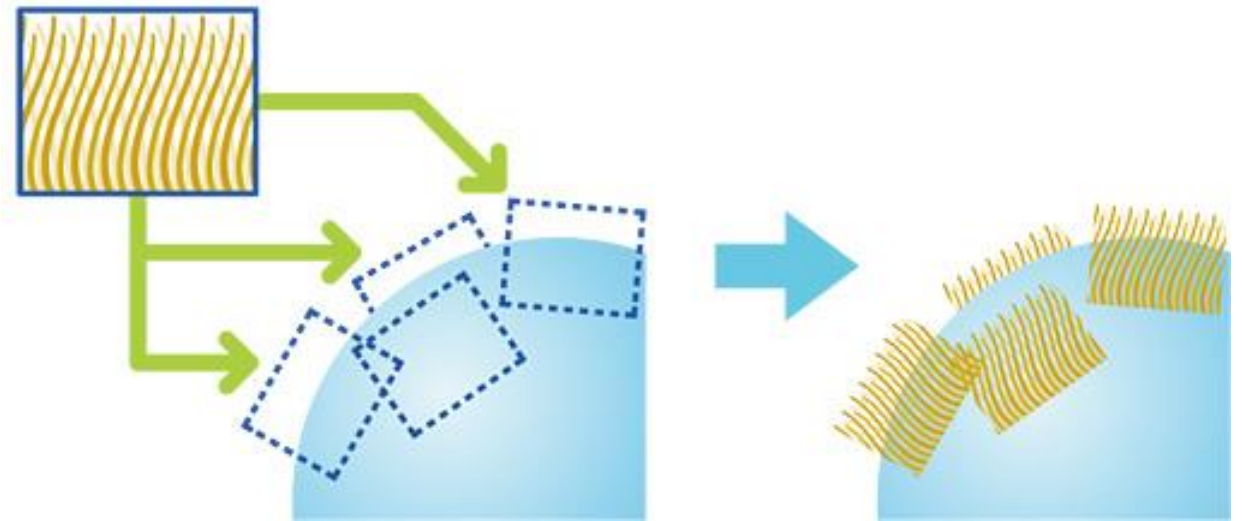
芝生を生やそう

- インスタンスング
- ジオメトリシェーダ



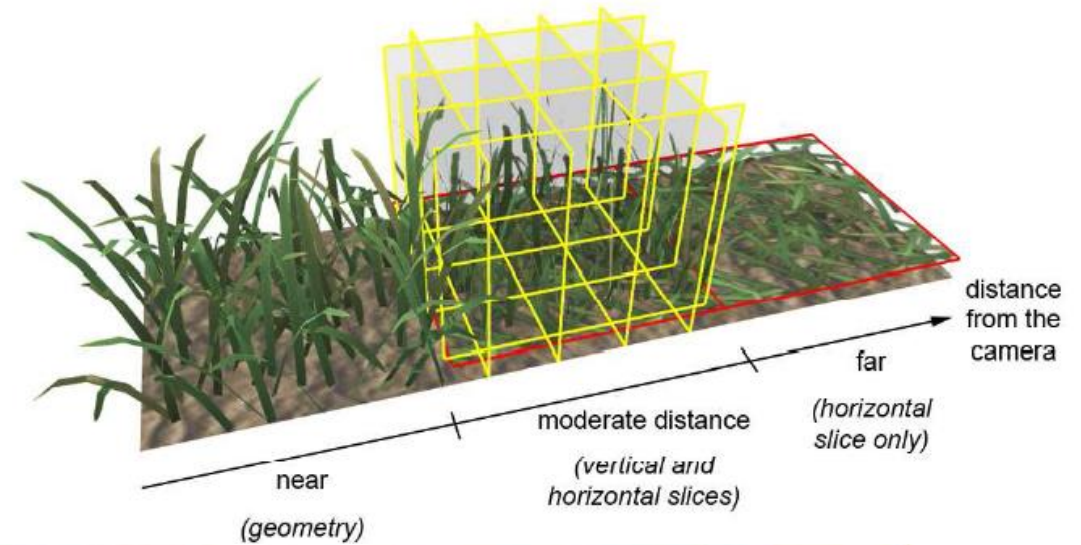
芝生のはやし方

- フィン法
 - オブジェクトに垂直にひれを立てる
 - 横から見たときに良い
 - 上から見たときに良くない
- シェル法
 - オブジェクトの上に膨らめたオブジェクトを何層もかぶせる
 - 上から見たときに良い
 - 横から見たときに良くない



Rendering Grass Terrains in Real-Time with Dynamic Lighting

- Kévin Boulanger, Sumanta Pattanaik, Kadi Bouatouch
- Siggraph 2006 sketch'



Horizontal
slices
Horizontal +
vertical slices
Horizontal +
vertical slices
Horizontal +
vertical slices
+ geometry
Geometry



よくある問題

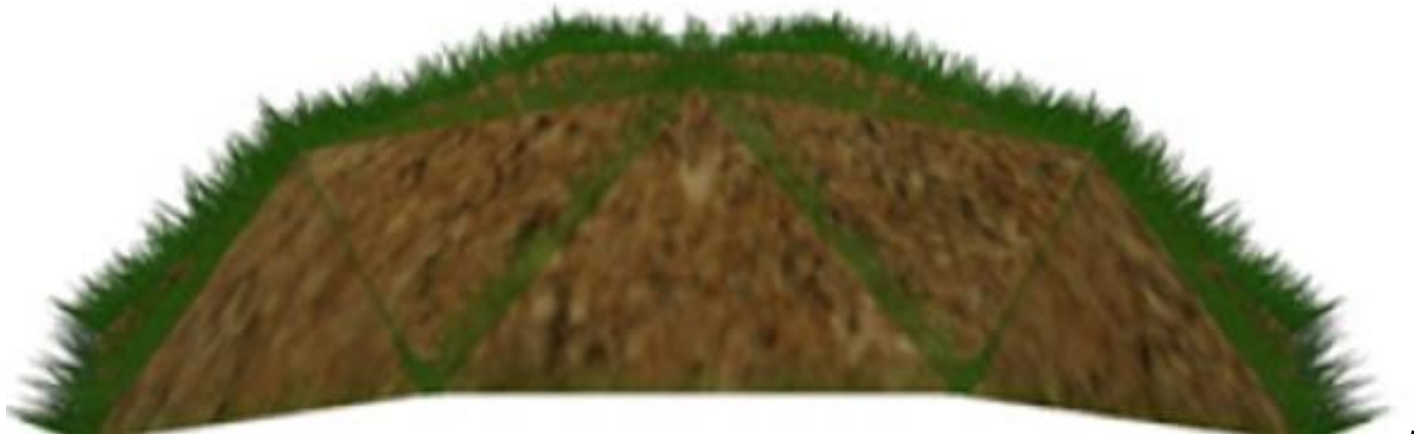
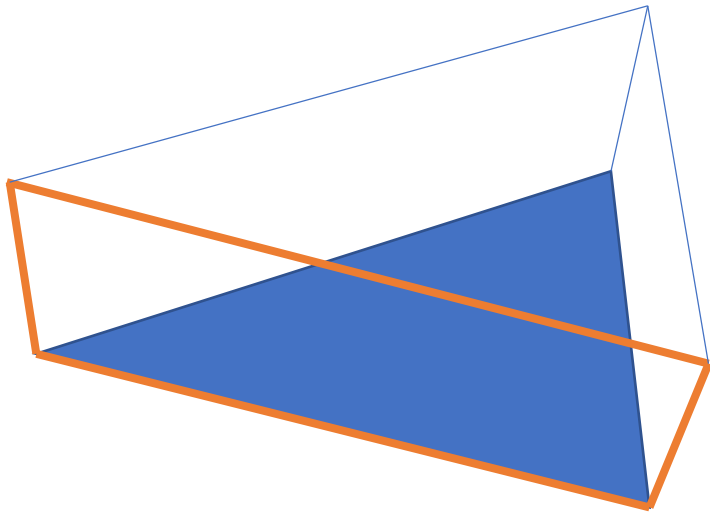
- デザイナーさんが面倒くさがって草をはやしてくれない
 - いい感じに生やしておいてよ…

よくある問題

- デザイナーさんが面倒くさがって草をはやしてくれない
 - いい感じに生やしておいてよ...
- 地形データからどのように草を生やせばよいか？

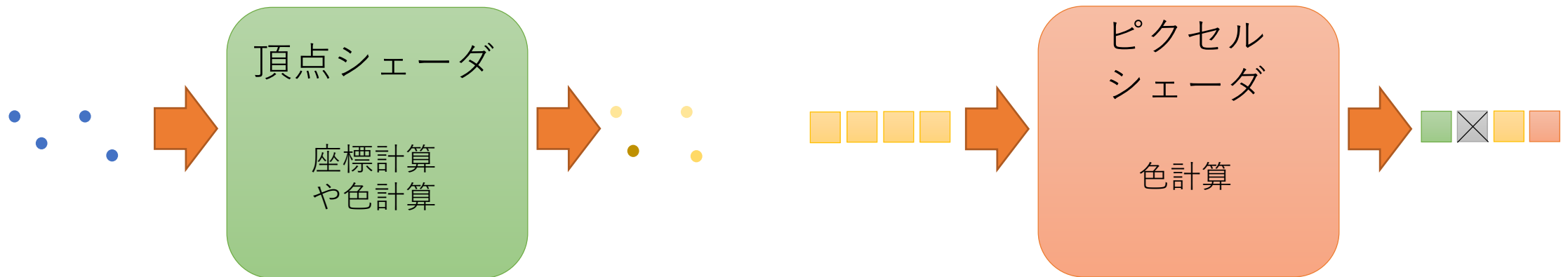
アイデア1: 稜線に草を生やす

- ポリゴンの輪郭：自動的に得られる線分の一つ
- 線分の場所に板を立ててフィンを加える



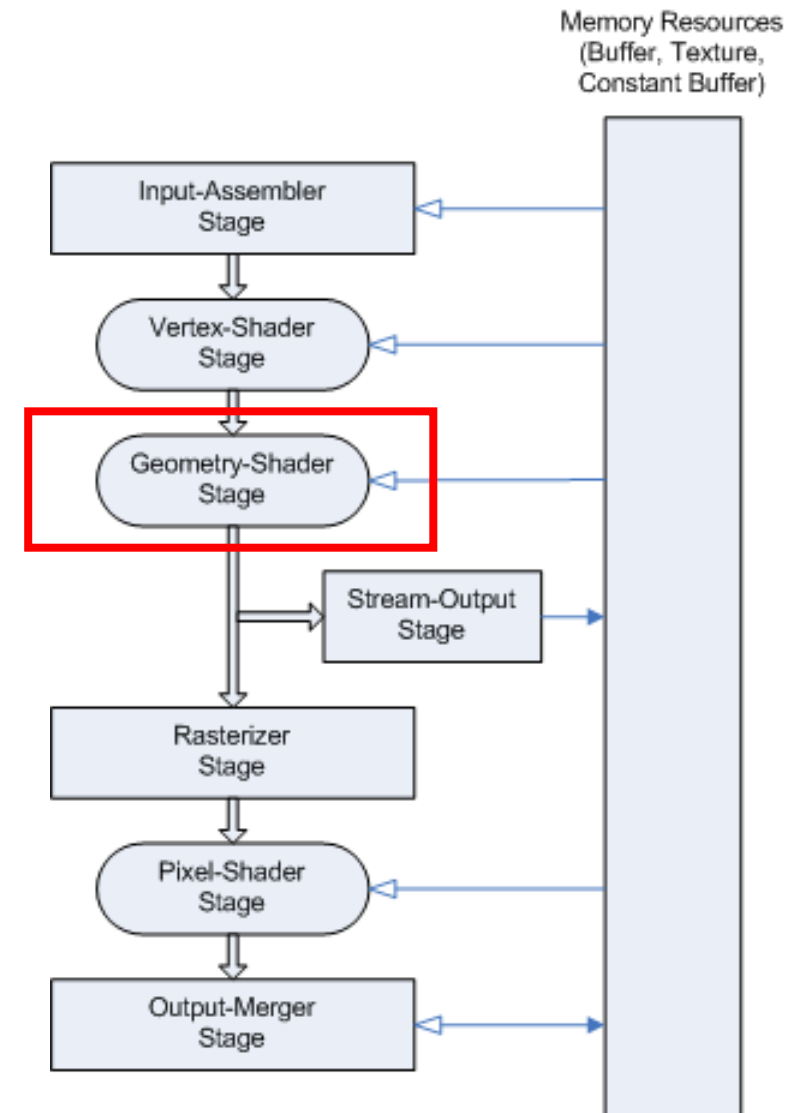
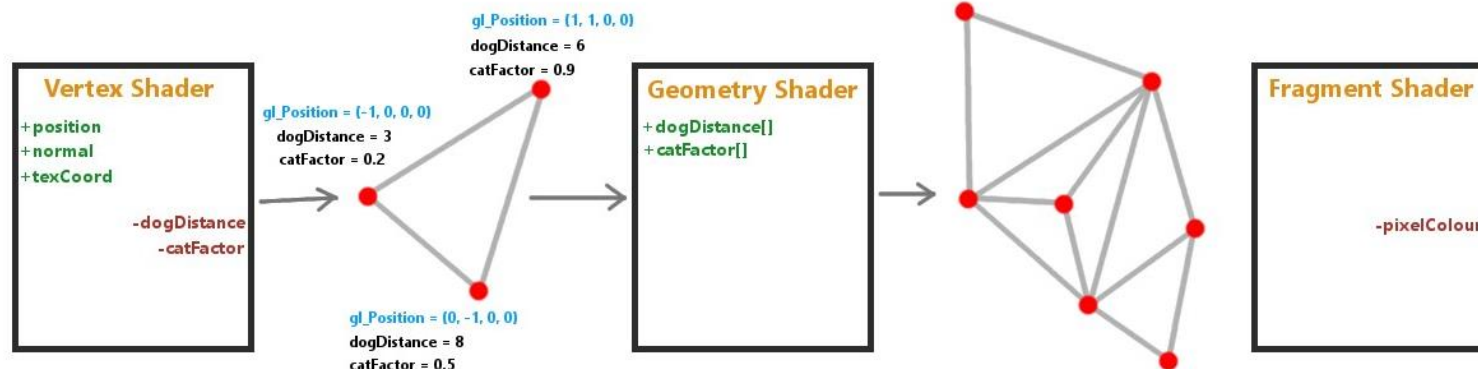
頂点シェーダやピクセルシェーダの欠点

- データを増やすことはできない
 - ピクセルシェーダのdiscard命令でデータを減らすことはできる



ジオメトリシェーダ

- 3番目の論理シェーダ
 - 頂点シェーダの出力から新たにポリゴン等を生成する
- 主な用途
 - キューブマップへの同時出力
 - VR画像の両眼同時描画
 - ポイントデータからビルボード生成
 - ポイントパーティクル
 - **ファー**



ジオメトリシェーダの使い方

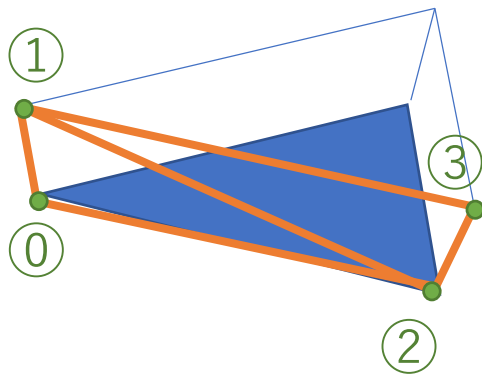
シェーダの書き替えだけ

- #pragma geometry
 - ジオメトリシェーダの関数名の指示
- 構造体
 - 頂点シェーダからピクセルシェーダへ受け渡す構造体を「頂点シェーダからジオメトリシェーダ」「ジオメトリシェーダからピクセルシェーダ」に分ける
- 関数
 - ジオメトリシェーダの関数を書く
- 頂点シェーダの出力を変更
- ピクセルシェーダの入力を変更

```
1 Shader "Geometry/EdgeGlassShader"
2 {
3     Properties
4     {
5         _GlassTex("Glass 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 geometry geom
17            #pragma fragment frag
18
19            #include "UnityCG.cginc"
20
21            struct appdata
22            {
23                float4 pos : POSITION;
24                float2 uv : TEXCOORD0;
25            };
26
27            struct v2g
28            {
29                float2 uv : TEXCOORD0;
30                float4 pos : SV_POSITION;
31            };
32
33            struct g2f
34            {
35                float2 uv : TEXCOORD0;
36                float4 pos : SV_POSITION;
37            };
38
39            sampler2D _GlassTex;
40            float4 _MainTex_ST;
41
42            v2g vert(appdata v) { ... }
43
44            [maxvertexcount(12)]
45            void geom(triangle v2g IN[3], inout TriangleStream<g2f> stream) { ... }
46
47            fixed4 frag(g2f i) : SV_Target { ... }
48            ENDCG
49        }
50    }
51 }
```

シェーダ関数

最大頂点数の指定(一時メモリの確保)



```
[maxvertexcount(12)]
void geom(triangle v2g IN[3], inout TriangleStream<g2f> stream)
{
    g2f o;
    // 頂点3つの三角形                                トライアングルストリップを出力
    // トライアングルストリップとして2ポリゴンの四角形を稜線に追加
    float h = 1.0;
    int n = 2;
    for (int i = 0; i < 3; i++) { ポリゴンの3つのエッジ分処理
        o.pos = UnityObjectToClipPos(IN[i].pos);
        o.uv = float2(0.0, 0.0);
        stream.Append(o); 頂点データの追加

        o.pos = UnityObjectToClipPos(IN[i].pos + float4(0, h, 0, 0));
        o.uv = float2(0.0, 1.0);
        stream.Append(o);

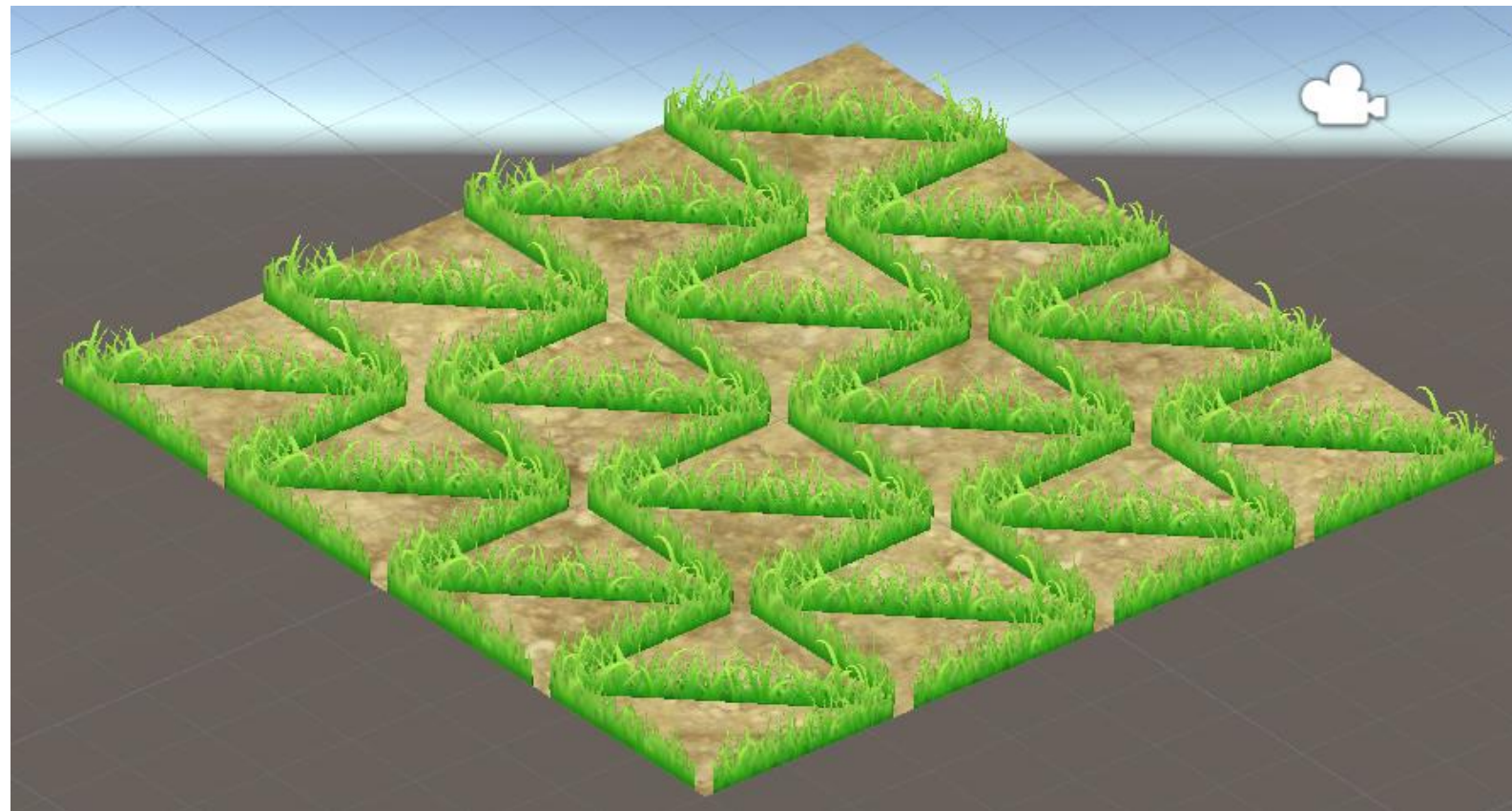
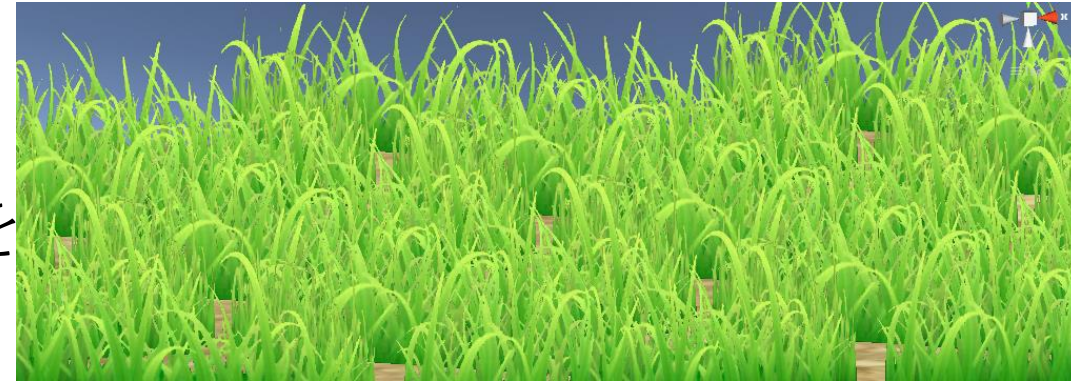
        o.pos = UnityObjectToClipPos(IN[n].pos);
        o.uv = float2(1.0, 0.0);
        stream.Append(o);

        o.pos = UnityObjectToClipPos(IN[n].pos + float4(0, h, 0, 0));
        o.uv = float2(1.0, 1.0);
        stream.Append(o);

        stream.RestartStrip(); // ストリップを切る
        n = i; // 次の相手方の頂点を指定
    }
}
```

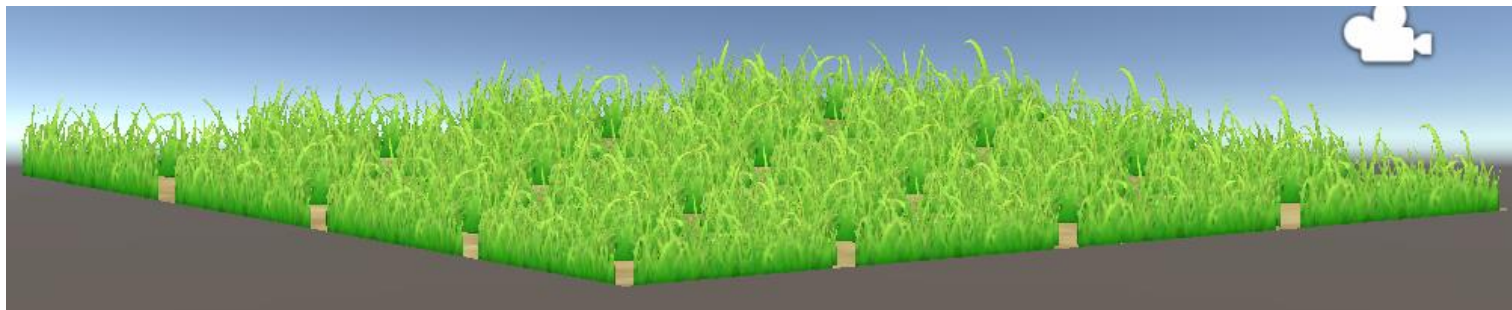
結果

- ProBuilder で作成したメッシュを2種類のシェーダーで描画
 - テクスチャ
 - ジオメトリシェーダーでの描画
- 表裏カリングにより1度しか描かれない
- フラグメントシェーダーは、カットオフ

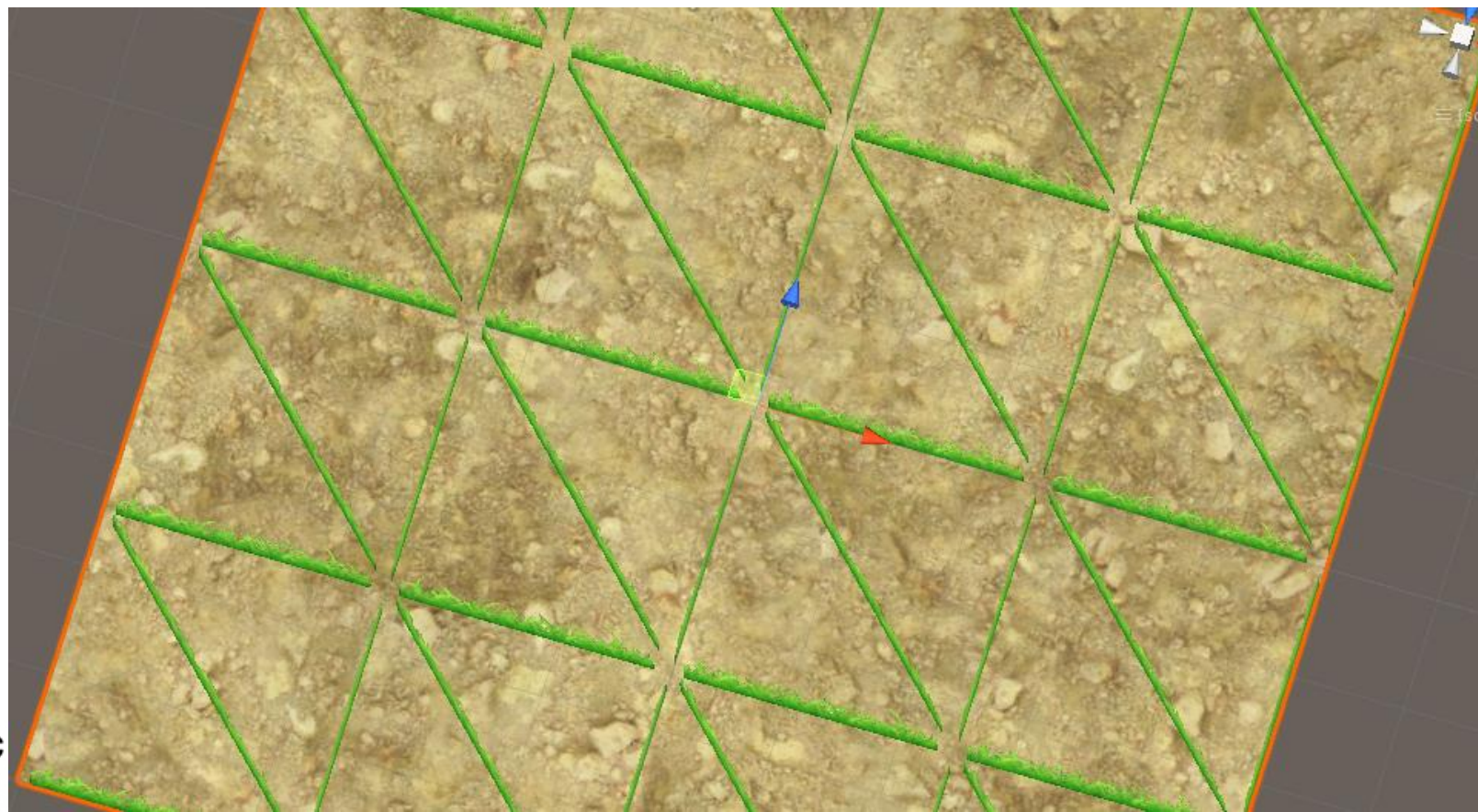


```
80 fixed4 frag(g2f i) : SV_Target{  
81     fixed4 col = tex2D(_GlassTex, i.uv);  
82     if (col.a <= 0.5) discard;  
83     return col;  
84 }
```


欠点

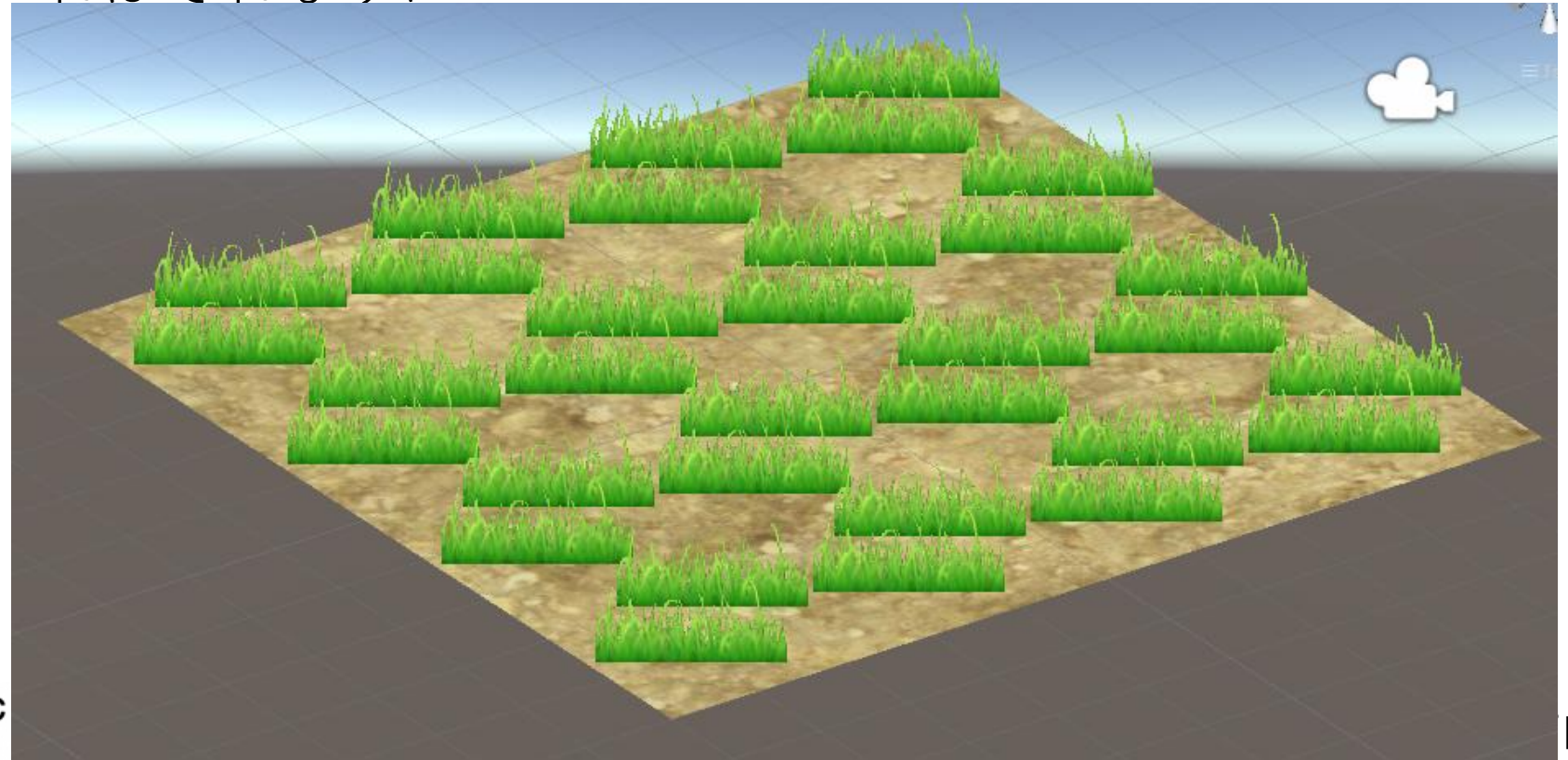
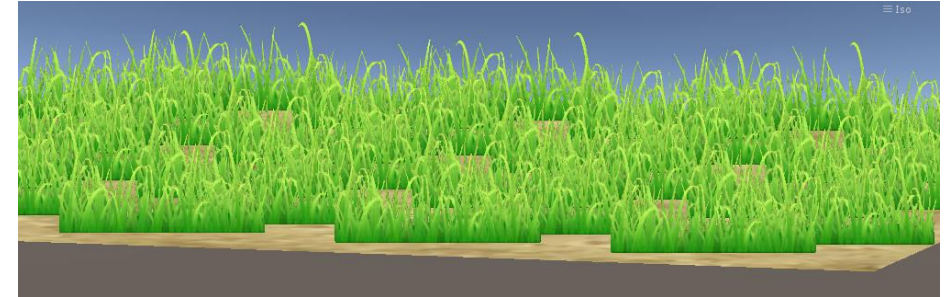


- 見た方向に応じて見た目が変わる
 - よこから見るケースに適している



草のはやし方その2

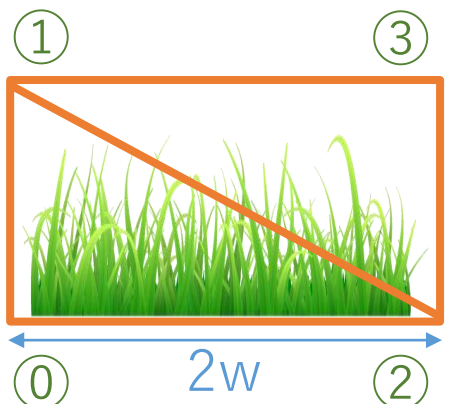
- ポリゴンの中心にビルボードを立てる
 - 等間隔のメッシュだと奇妙だが、地形がゆがんでいればそれなりに
 - 向きを変えても草が細く見えない



シェーダ関数(の変更だけでよい)

```
50 [maxvertexcount(4)]
51 void geom(triangle v2g IN[3], inout TriangleStream<g2f> stream)
52 {
53     // ポリゴンの中心を求める
54     float4 pos = float4(          ポリゴンの重心を求め、ビュー空間に変換する
55         UnityObjectToViewPos((IN[0].pos.xyz + IN[1].pos.xyz + IN[2].pos.xyz) / 3.0),
56         1.0);
57
58     // posを中央下に、ビュー空間での四角形の頂点を求め、射影空間に変換する
59     float w = 1.;
60     float h = 1.;
61
62     g2f o;
63     // トライアングルストリップとして2ポリゴンを出力
64     o.pos = mul(UNITY_MATRIX_P, pos + float4(-w, 0, 0, 0));
65     o.uv = float2(0.0, 0.0);
66     stream.Append(o);
67
68     o.pos = mul(UNITY_MATRIX_P, pos + float4(-w, h, 0, 0));
69     o.uv = float2(0.0, 1.0);
70     stream.Append(o);
71
72     o.pos = mul(UNITY_MATRIX_P, pos + float4(+w, 0, 0, 0));
73     o.uv = float2(1.0, 0.0);
74     stream.Append(o);
75
76     o.pos = mul(UNITY_MATRIX_P, pos + float4(+w, h, 0, 0));
77     o.uv = float2(1.0, 1.0);
78     stream.Append(o);
79 }
```

ビュー空間でz軸と垂直に板を置くとカメラの正面を向かせることができる



① ③

② ④

2w

h

欠点



- 真上からの見た目がおかしい



やってみよう

- シェル法の草も追加してみよう

