Unity 系统学习|电子笔记

课题：3D RPG 项目学习

学习导师：杨哥

学习本体：（Bilibili）M_Studio

使用的引擎及其版本：Unity2020 LTS

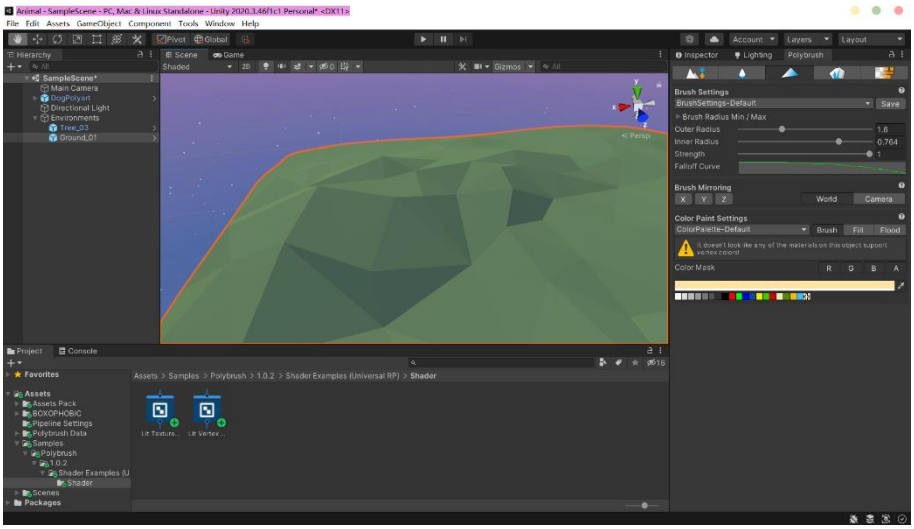
渲染管线：URP

·目录 Menu（重点 P 集前会标注 * 提示）

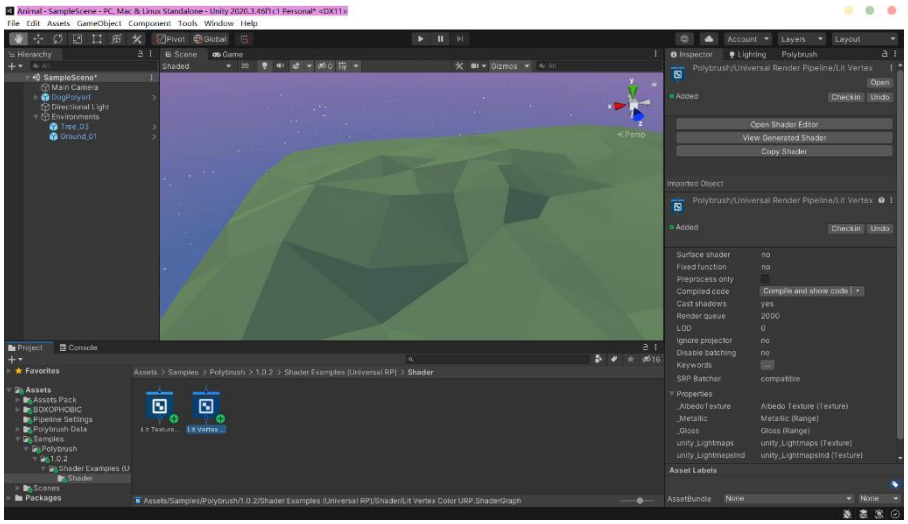
【注】由于教程中反复对 cs 脚本代码进行修改，截至 6P 的 cs 最终版代码请见 5-附录中（P18），其他页码上的 cs 代码（PlayerController.cs 和 MouseManager.cs 均为紧跟教程的非完整版本）

*3P.PolyBrush（低多边形地编）

（1）PolyBrush 地形上色

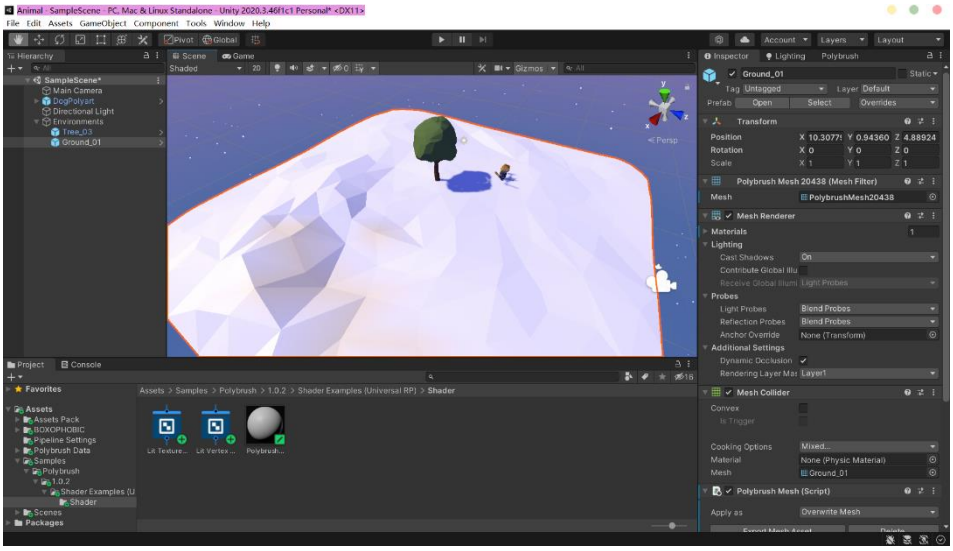


1.找到文件：Assets/……/ Polybrush/1.0.2/Shader Examples (Universal RP)/Shader/Lit Vertex Color URP.ShaderGraph



鼠标右键-Create-Material

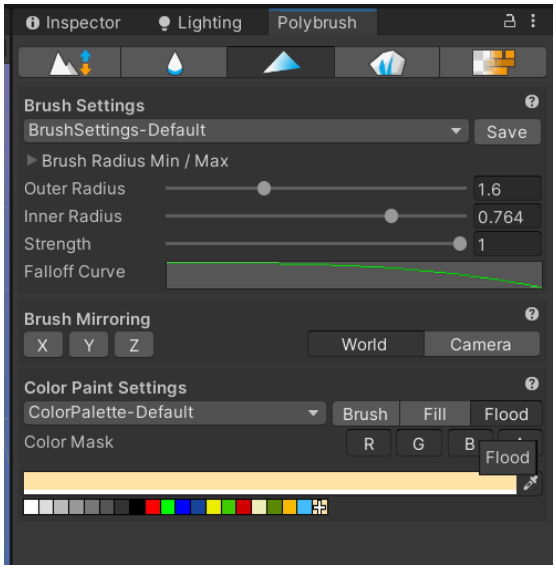
将新建的 Material 球拖入场景 Scene 即可，会发现地面变为白色（如图↓）



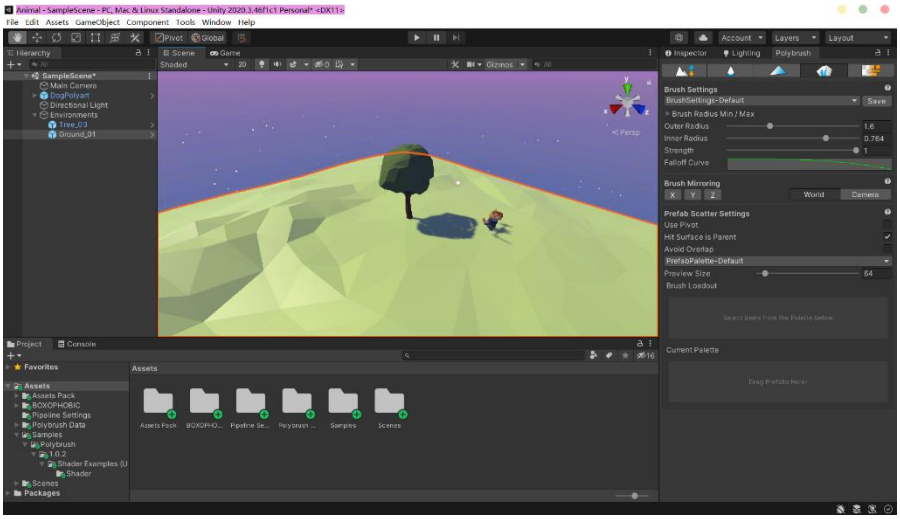
然后可以自主上色

【补充】Polybrush-Color Paint Settings

Flood：填充，可将所选颜色应用于整个 poly 地形

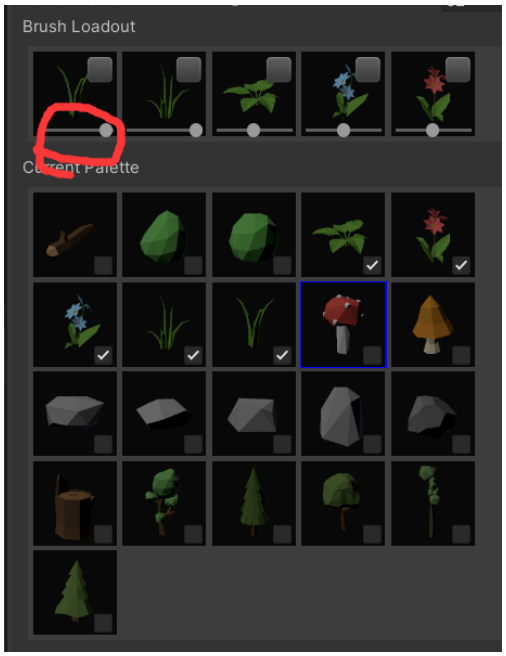


（2）PolyBrush 刷地形物件（草、树木等）

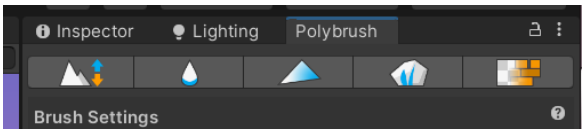

File Edit Assets GameObject Component Tools Window Help
Hierarchy
SampleScene*
Main Camera
DogPolyart
Directional Light
Environments
Tree_03
Ground_01
Scene
Game
Shaded
Gizmos
Persp
Inspector
Lighting
Polybrush
Brush Settings
BrushSettings-Default
Save
Brush Radius Min / Max
Outer Radius
Inner Radius
Strength
Falloff Curve
Brush Mirroring
World
Camera
Prefab Scatter Settings
Use Pivot
Hit Surface is Parent
Avoid Overlap
PrefabPalette-Default
Preview Size
Brush Loadout
Current Palette
Project
Console
Favorites
Assets
Assets Pack
BOXOPHOBIC
Pipeline Settings
Polybrush Data
Samples
Polybrush
Scenes
Packages

拖入需要用到的 prefab 文件（已绑定材质的模型文件）

拖入 Current Palette 中，并勾选需要用到的 prefab 文件



调节滑轮，可设置该 prefab 的出现概率

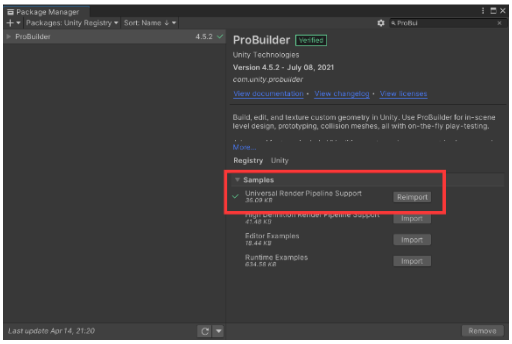


【Tips】当选择 PolyBrush 组件的任何一个选项时，你都会失去你的 Scene 坐标轴，请记得还原（如图↑）

如遇到绘制物件时悬空，请勾选 Use Pivot（如图↓）

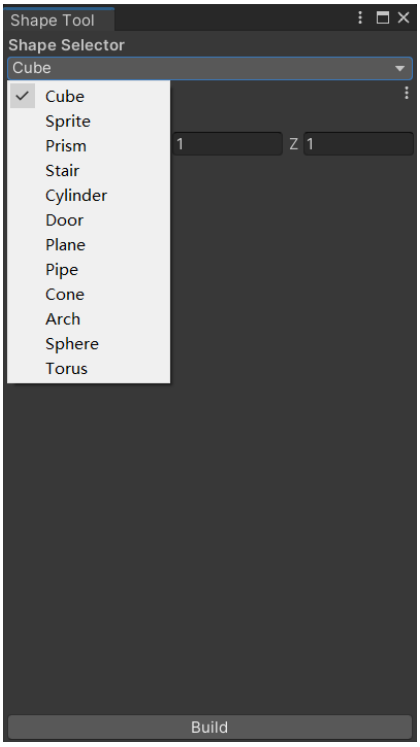
（3）低多边形地形大小修改（使用插件 ProBuilder）

【Tips】务必安装 ProBuilder 的 URP sample

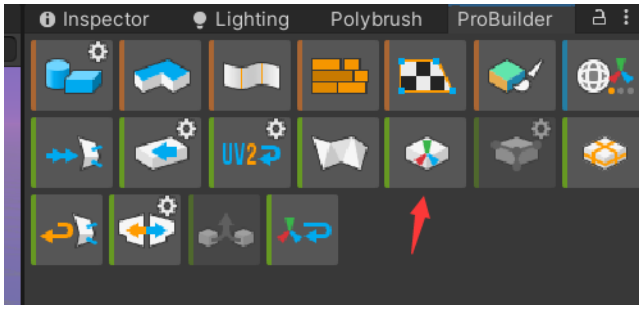


打开插件-Tools-ProBuilder-ProBuilder Window

←点击齿轮进行进一步设置（或 Alt+点击）



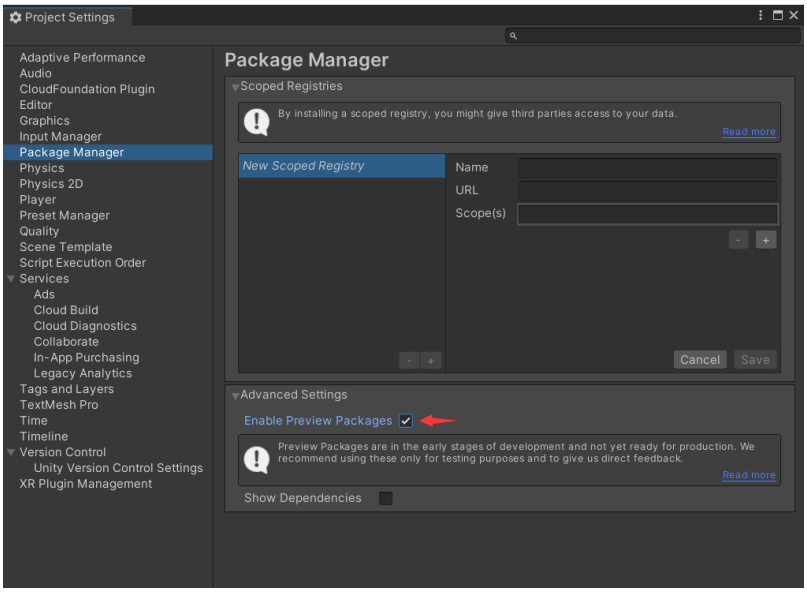
选 Plane→



将 Plane 中心点回至 Plane 中心

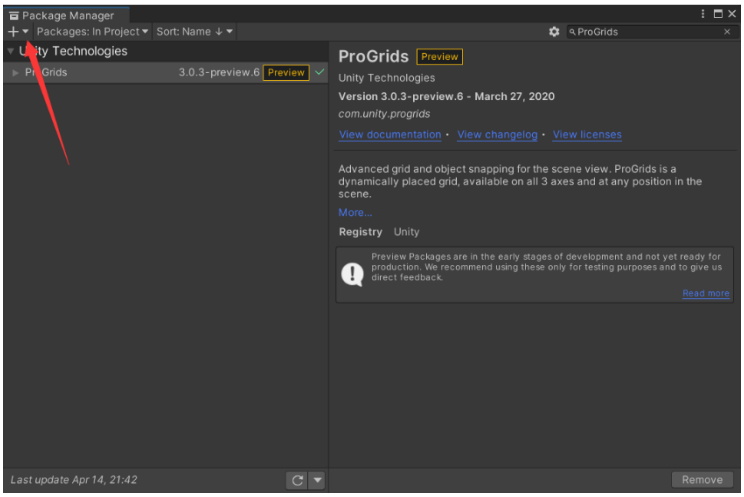
（3-1）搭配 ProBuilder 的插件-ProGrids

找到 Edit-Project Settings-Packet Manage-Enable Preview Packages（解锁此选项）

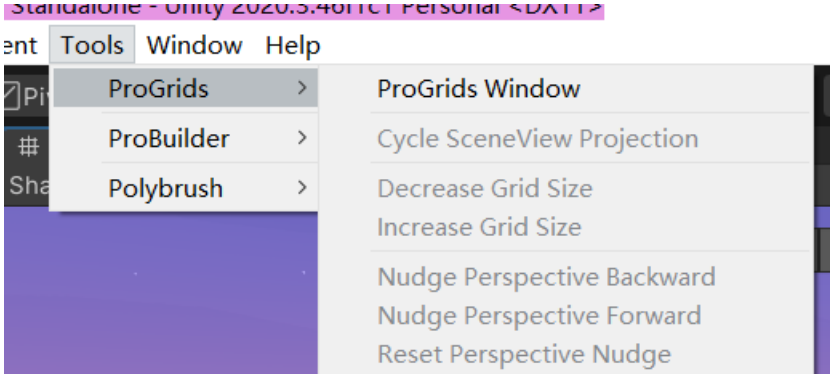


随后按照正常的 Packet Manage 安装 ProGrids

【备用方案】＋-Add packet from git URL（URL：com.unity.progrids）

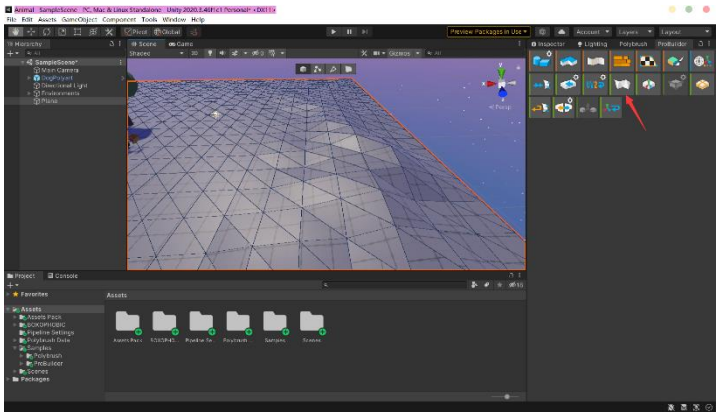


启动 ProGrids（↓）



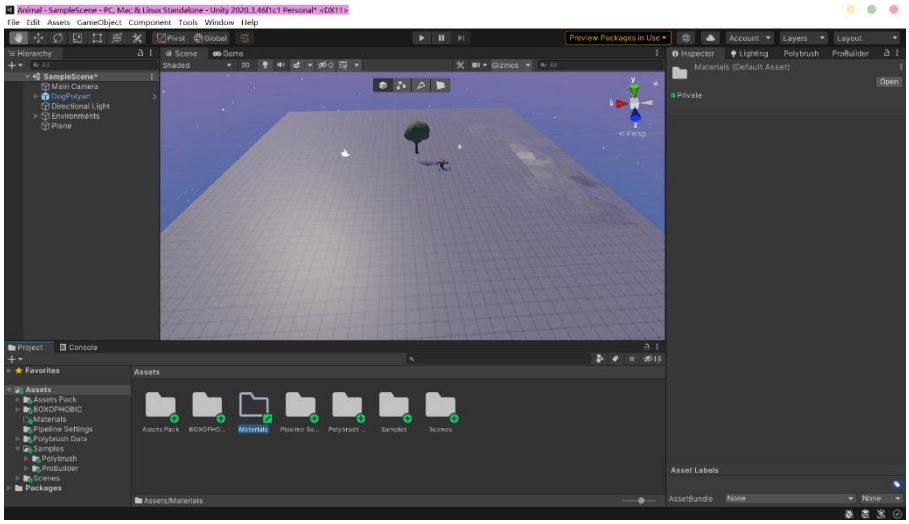
ProGrids 可不用

（3.2）继续通过 PolyBrush 和 ProBuilder 绘制地形



点击该按钮将地面的格点形式（原：方形）转换为三角形格点（Poly 风格）

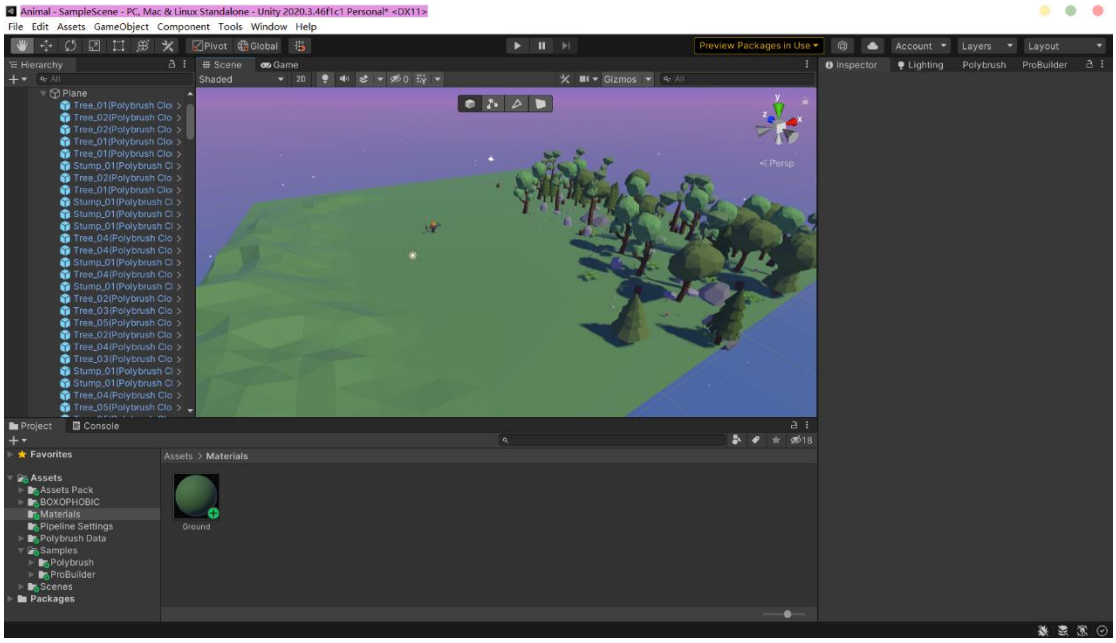
在 Assests 里新建一个文件夹，命名为 Materials



在文件夹内新建一个 Materials 并拖入 Scene

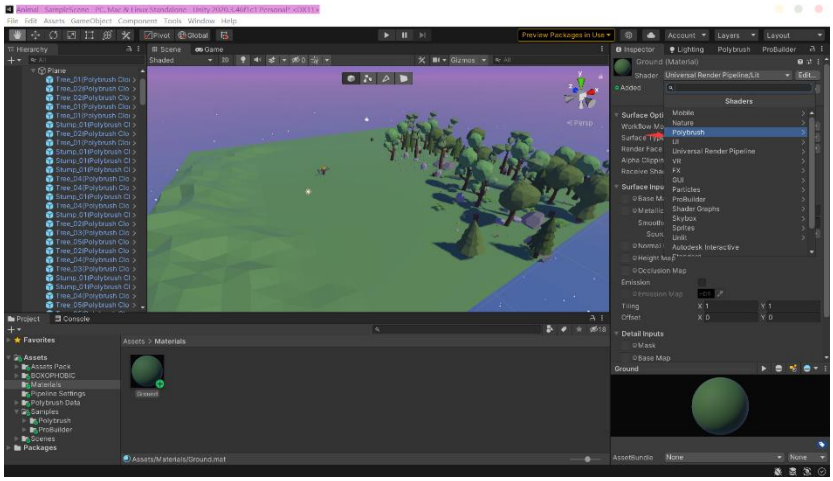
按照正常方式进行绘制

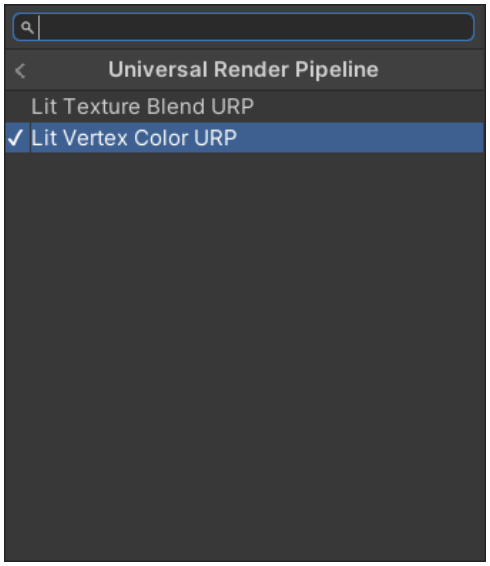
【3P 结束·Project 完成度展示】



*4P. Navigation（智能导航地图烘焙）

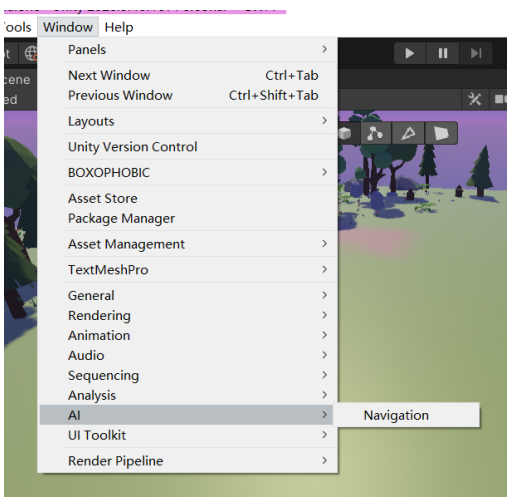
【接上 P】给地形刷上颜色



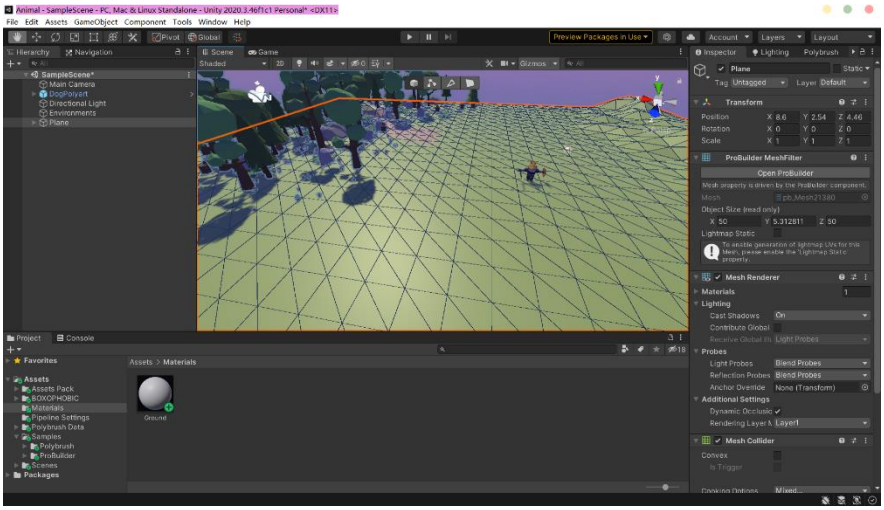


(1) Navigation

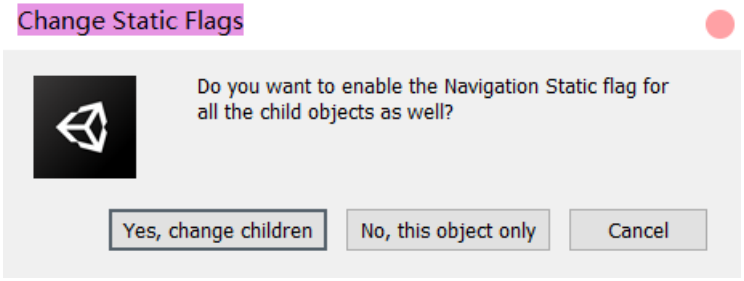
打开 Navigation



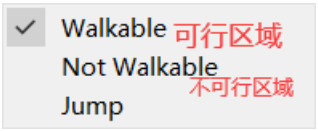
选择地面（Plane），点开 Inspector，勾选 Static-NavigationStatic



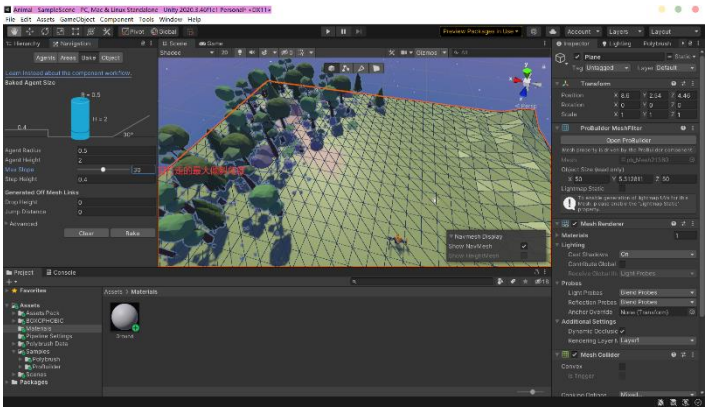
因为这次我们其他的 Prefabs（诸如树木之类）我们不希望让角色穿过，所以我们选用“No, this object only”



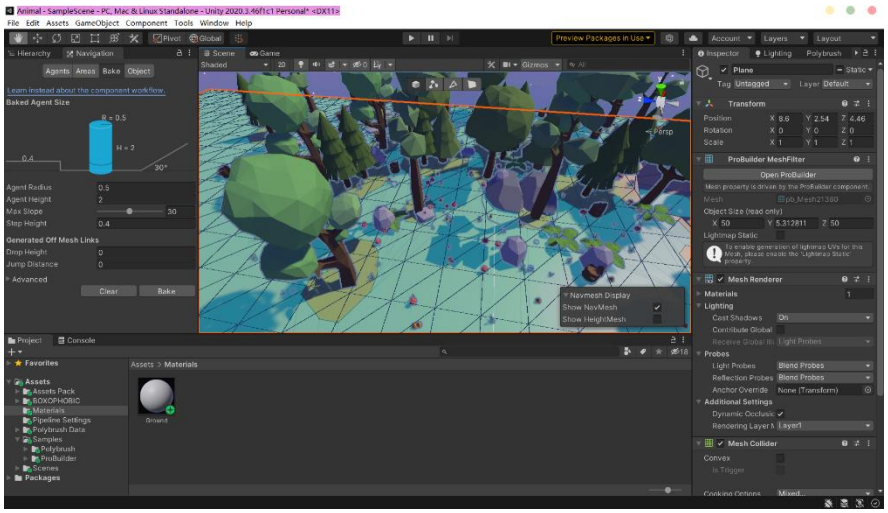
设置完毕后回到 Navigation-Object，找到 Navigation Area，设置为 Walkable（可行）



在 Bake 中 Bake 场景

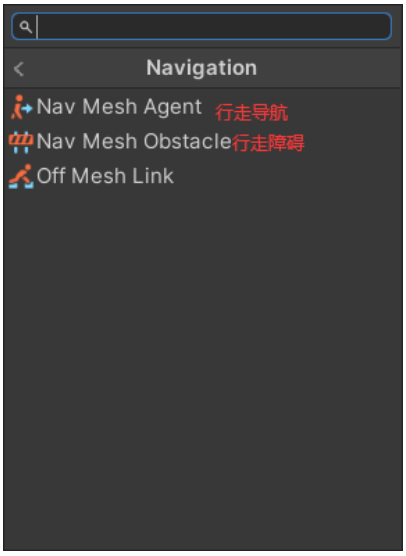


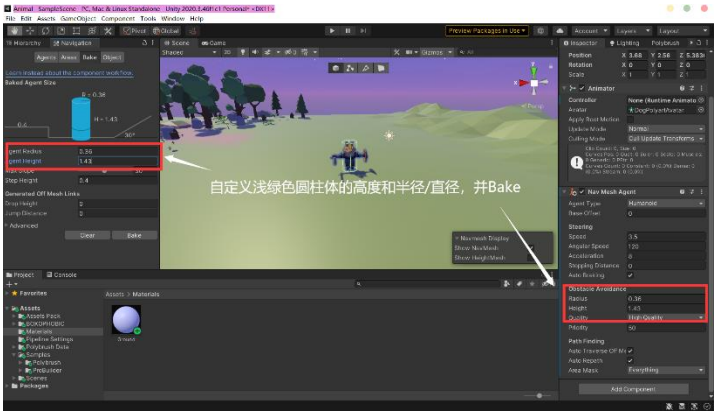
用类似的方法给树木、大石头添加 Not Walkable 并 Bake



选择项目角色，删除原有的 controller

Add Component-Navigation（如下图）





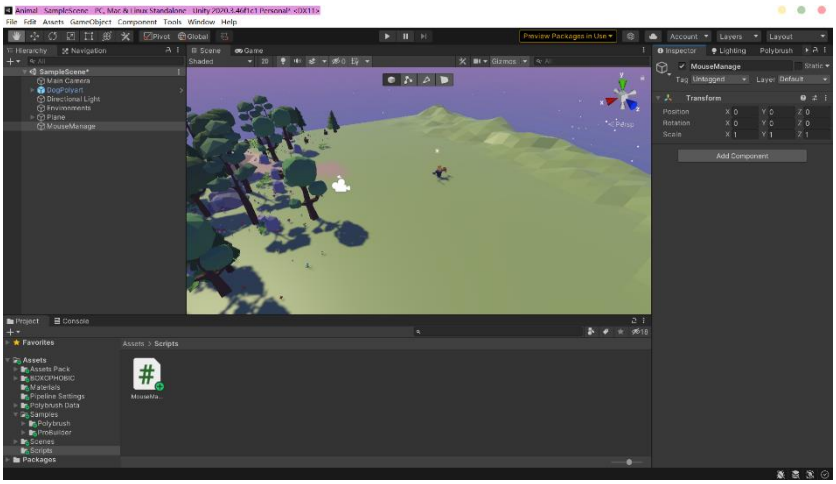
选用 Nav Mesh Agent

使用类似的方法将大石头等内容添加 Not Walkable 并 Bake

*5P. MouseManager（鼠标控制人物移动）

代码课程

在文件夹（新建）Scripts 中新建 C#脚本（命名为 MouseManager）



新建空项目同样命名 MouseManager，习惯养成：Reset

·MouseManager.cs 源码（此处请看视频 P5 的讲解，笔记说不清）：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;

[System.Serializable]
public class EventVector3 : UnityEvent<Vector3> { }
public class MouseManage : MonoBehaviour
{
    RaycastHit hitInfo;

    public EventVector3 OnMouseClicked;

    void Update()
    {
        SetCursorTexture();
        MouseControl();
    }

    void SetCursorTexture()
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
```

```
        if (Physics.Raycast(ray, out hitInfo))
        {
            //切换鼠标贴图
        }
    }

    void MouseControl()
    {
        if(Input.GetMouseButtonDown(0) && hitInfo.collider!=null)
        {
            if (hitInfo.collider.gameObject.CompareTag("Ground"))
                OnMouseClicked?.Invoke(hitInfo.point);
        }
    }
        }
```

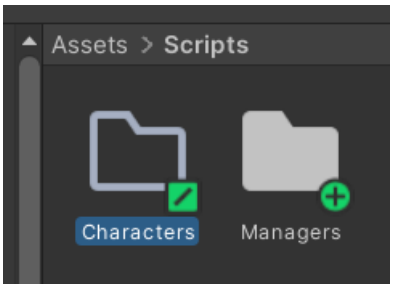
·Stop Distanse 指的是在你所点击的点的 x（设置的数值）格的位置时就自动停下（适用于制作打怪时的停止位置）

*6P. SetCursor（设置鼠标指针）

在 6P 中重新修改完成的 mousemanage.cs 源码：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;
using System;


public class MouseManage : MonoBehaviour
{
    public static MouseManage Instance;

    RaycastHit hitInfo;

    public event Action<Vector3> OnMouseClicked;

    void Awake()
    {
        if (Instance != null)
            Destroy(gameObject);
        Instance = this;
    }
    void Update()
    {
        SetCursorTexture();
        MouseControl();
    }

    void SetCursorTexture()
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);

        if (Physics.Raycast(ray, out hitInfo))
        {
            //切换鼠标贴图
        }
    }

    void MouseControl()
    {
```

```
            if(Input.GetMouseButtonDown(0) && hitInfo.collider!=null)
            {
                if (hitInfo.collider.gameObject.CompareTag("Ground"))
                    OnMouseClicked?.Invoke(hitInfo.point);
            }
        }
}
```

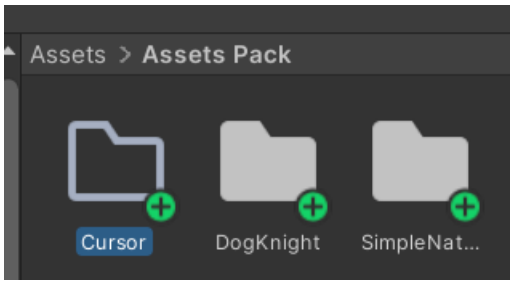


新建 Folder 对脚本进行分类

并新建 Characters 文件夹放置控制器，在内新建 cs 脚本

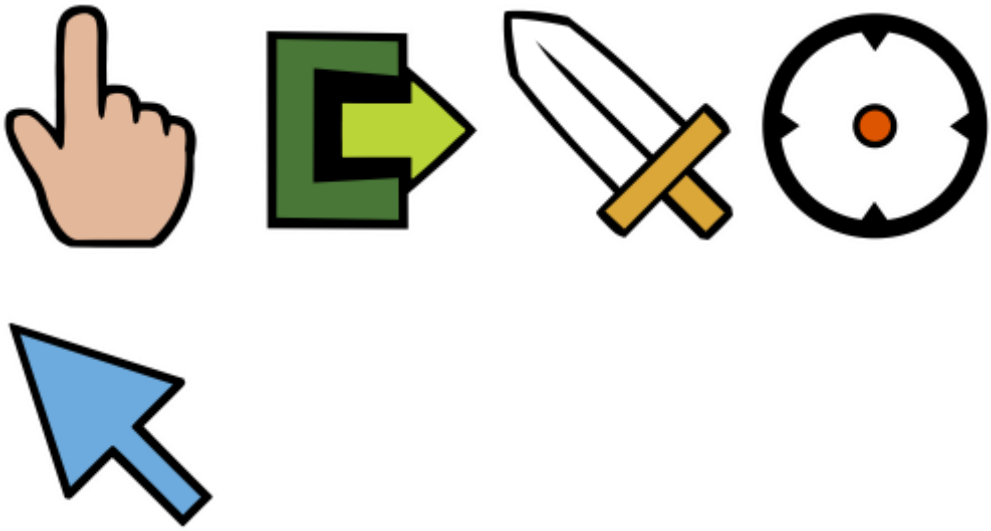
PlayerController.cs 源码：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

public class PlayerController : MonoBehaviour
{
    private NavMeshAgent agent;

    void Awake()
    {
        agent = GetComponent<NavMeshAgent>();
    }

    void Start()
    {
        MouseManage.Instance.OnMouseClicked += MoveToTarget;
    }

    public void MoveToTarget(Vector3 target)
    {
        agent.destination = target;
    }
}
```

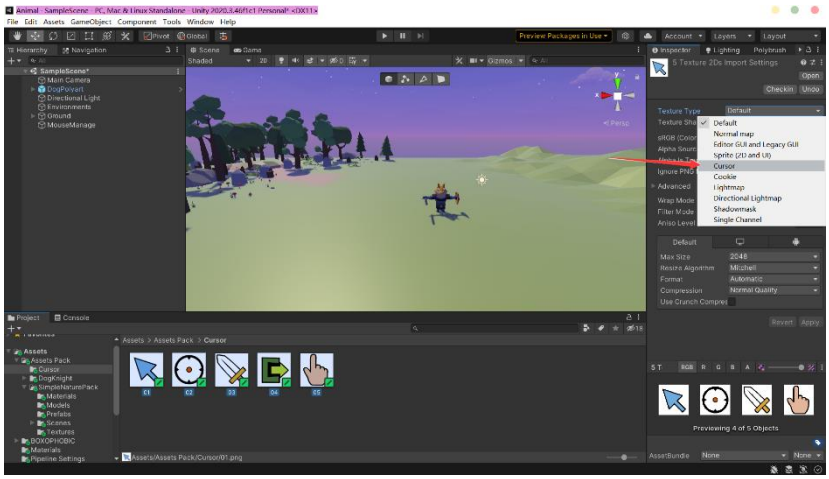


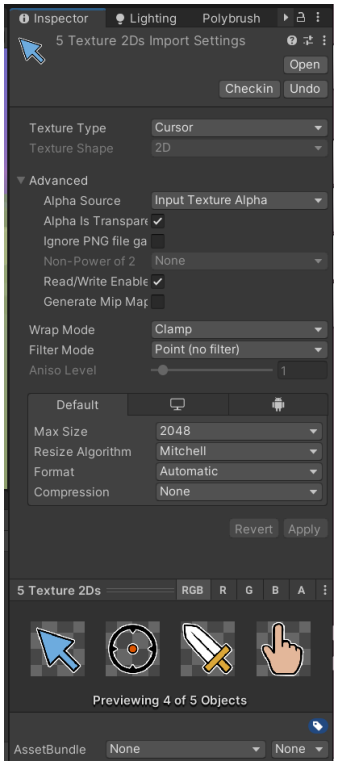
新建 Cursor（鼠标）文件夹

·Cursor 下载（png，与教程配套）

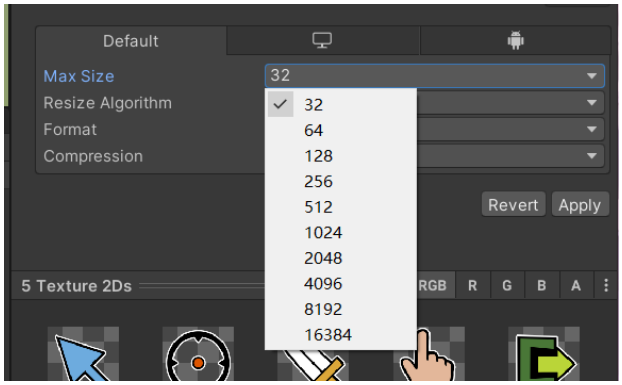


导入并全选，Texture Type 修改为 Cursor





设置为如图



在此处修改鼠标的大小

【附录】

MouseManager.cs 源码：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;
using System;


public class MouseManage : MonoBehaviour
{
    public static MouseManage Instance;

    public Texture2D point, doorway, attack, target, arrow;

    RaycastHit hitInfo;

    public event Action<Vector3> OnMouseClicked;

    void Awake()
    {
        if (Instance != null)
            Destroy(gameObject);
```

```
        Instance = this;
    }
    void Update()
    {
        SetCursorTexture();
        MouseControl();
    }

    void SetCursorTexture()
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);

        if (Physics.Raycast(ray, out hitInfo))
        {
            //切换鼠标贴图
            switch(hitInfo.collider.gameObject.tag)
            {
                case "Ground":
                    Cursor.SetCursor(target, new Vector2(16, 16), CursorMode.Auto);
                    break;
            }
        }
    }

    void MouseControl()
    {
        if(Input.GetMouseButtonDown(0) && hitInfo.collider!=null)
        {
            if (hitInfo.collider.gameObject.CompareTag("Ground"))
                OnMouseClicked?.Invoke(hitInfo.point);
        }
    }
}
```

PlayerController.cs 源码：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

public class PlayerController : MonoBehaviour
{
    private NavMeshAgent agent;
```

```
    void Awake()
    {
        agent = GetComponent<NavMeshAgent>();
    }

    void Start()
    {
        MouseManage.Instance.OnMouseClicked += MoveToTarget;
    }

    public void MoveToTarget(Vector3 target)
    {
        agent.destination = target;
    }
}
```