

新年好


欢迎加入LightDir (光向) 研习社
欢迎大家一同探索开源共享的知识分享模式

新年好

欢迎加入LightDir (光向) 研习社
欢迎大家一同探索开源共享的知识分享模式

今日内容

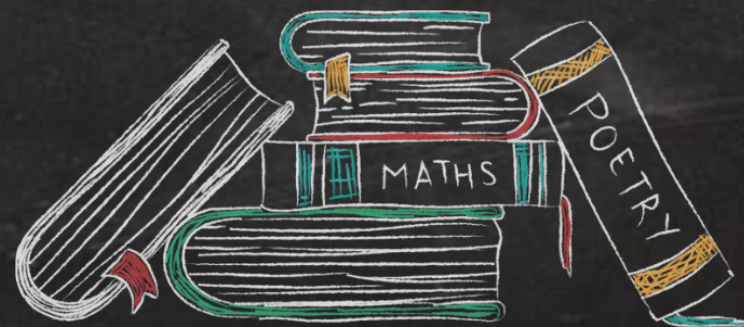


 化神·控制器面板

$$\begin{array}{r} 24 \times 5 \\ \hline 33 \end{array}$$

1

大乘·控制器面板



01 代码大概搂一眼

1. 引用命名空间;
2. 声明一个继承自Editor的类LightingControllerGUI;
3. 将此GUI类绑定到对应的Mono脚本;
4. 重载基类的OnInspectorGUI方法;
5. 绘制功能按钮区方法;
6. 绘制参数面板区方法和需要的开关变量;

```
using UnityEditor;
using UnityEngine;

[CustomEditor(typeof(LightingController))]
public class LightingControllerGUI : Editor
{
    // 重载GUI绘制方法
    // Frequently called
    public override void OnInspectorGUI() { ... }
    // 绘制功能按钮GUI
    // Frequently called 1 usage
    private void DrawFunctionButtons(LightingController controller) { ... }
    // 组开关变量
    private bool _groupAToggle;
    private bool _groupBToggle;
    private bool _groupCToggle;
    private bool _groupDToggle;
    private bool _groupEToggle;
    // 绘制Shader全局参数GUI
    // Frequently called 1 usage
    private void DrawGlobalProperties(LightingController controller) { ... }
}
```

02 面板结构与GUI绘制方法对应关系

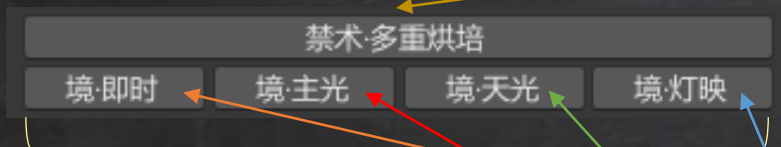


功能按钮区

参数面板区

```
// 重载GUI绘制方法
// Frequently called
public override void OnInspectorGUI()
{
    // 获取控制器
    var controller = target as LightingController;
    // 判空
    if(controller == null) return;
    // 绘制功能按钮区
    DrawFunctionButtons(controller);
    // 绘制参数面板区
    DrawGlobalProperties(controller);
}
```

03 功能按钮区绘制过程展开



```
// 绘制功能按钮GUI
Frequently called 1 usage
private void DrawFunctionButtons(LightingController controller)
{
    // 第一行 多重烘培大按钮
    if (GUILayout.Button(text: "禁术·多重烘培"))
        controller.MultiBake();

    // 第二行 搞个水平布局
    EditorGUILayout.BeginHorizontal();
    {
        // 实时光照启动按钮
        if (GUILayout.Button(text: "境·即时"))
        {
            Lightmapping.Clear();
            controller.ArrangeBakeScene(LightingController.BakeMode.Default);
        }

        // 烘培并预览主光成分按钮
        if (GUILayout.Button(text: "境·主光"))
            controller.Bake(LightingController.BakeMode.BakeMainLight);

        // 烘培并预览天光成分按钮
        if (GUILayout.Button(text: "境·天光"))
            controller.Bake(LightingController.BakeMode.BakeSkyLight);

        // 烘培并预览自发光GI成分按钮
        if (GUILayout.Button(text: "境·灯映"))
            controller.Bake(LightingController.BakeMode.BakeEmissionGI);
    }
    EditorGUILayout.EndHorizontal();
}
```


04 参数面板区绘制过程展开

记录各组开关状态

绘制各参数组

- ▶ 材质属性
- ▶ 主光配置
- ▶ 天光配置
- ▶ 反射配置
- ▶ 自发光GI配置

```
// 组开关变量
private bool _groupAToggle;
private bool _groupBToggle;
private bool _groupCToggle;
private bool _groupDToggle;
private bool _groupEToggle;
```

检测参数区是否有修改

如有修改执行的操作

```
// 绘制Shader全局参数GUI
// Frequently called 1 usage
private void DrawGlobalProperties(LightingController controller)
{
    // 开始参数修改检测
    EditorGUI.BeginChangeCheck();
    {
        // 参数组A: 材质属性
        _groupAToggle = EditorGUILayout.BeginFoldoutHeaderGroup(_groupAToggle, content: "材质属性");
        if (_groupAToggle){...}
        EditorGUILayout.EndFoldoutHeaderGroup();

        // 参数组B: 主光配置
        _groupBToggle = EditorGUILayout.BeginFoldoutHeaderGroup(_groupBToggle, content: "主光配置");
        if (_groupBToggle){...}
        EditorGUILayout.EndFoldoutHeaderGroup();

        // 参数组C: 天光配置
        _groupCToggle = EditorGUILayout.BeginFoldoutHeaderGroup(_groupCToggle, content: "天光配置");
        if (_groupCToggle){...}
        EditorGUILayout.EndFoldoutHeaderGroup();

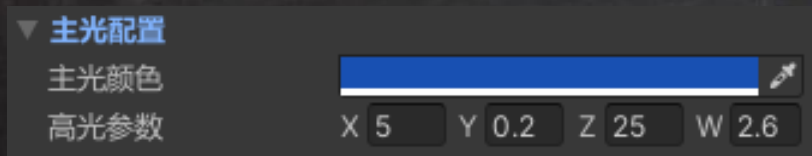
        // 参数组D: 反射配置
        _groupDToggle = EditorGUILayout.BeginFoldoutHeaderGroup(_groupDToggle, content: "反射配置");
        if (_groupDToggle){...}
        EditorGUILayout.EndFoldoutHeaderGroup();

        // 参数组E: 自发光GI配置
        _groupEToggle = EditorGUILayout.BeginFoldoutHeaderGroup(_groupEToggle, content: "自发光GI配置");
        if (_groupEToggle){...}
        EditorGUILayout.EndFoldoutHeaderGroup();
    }

    // 结束参数修改检测 变则设置Shader全局参数 参数非常多时逐个或按组做此操作
    if (EditorGUI.EndChangeCheck())
    {
        controller.UpdateGlobalProperties();
        EditorUtility.SetDirty(controller);
    }
}
```

05 其中一组参数的绘制

获取组开关状态选择是否渲染该参数GUI



```
// 参数组B: 主光配置
_groupBToggle = EditorGUILayout.BeginFoldoutHeaderGroup(_groupBToggle, content: "主光配置");
if (_groupBToggle)
{
    controller.mainLightCol = EditorGUILayout.ColorField(
        label: "主光颜色",
        controller.mainLightCol);
    controller.specParams = EditorGUILayout.Vector4Field(
        label: "高光参数",
        controller.specParams);
}
EditorGUILayout.EndFoldoutHeaderGroup();
```




Thanks