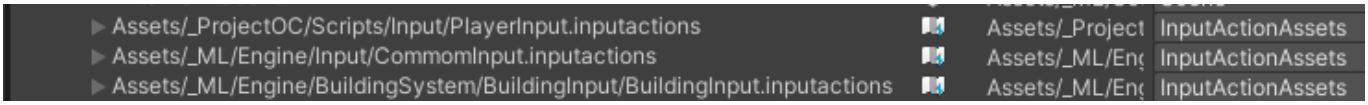
动态按键提示程序设计文档

背景描述

在文档Main Interact_主交互中，要求用户可以动态地更改按键绑定，所以UI中的按键提示文本也应动态显示。

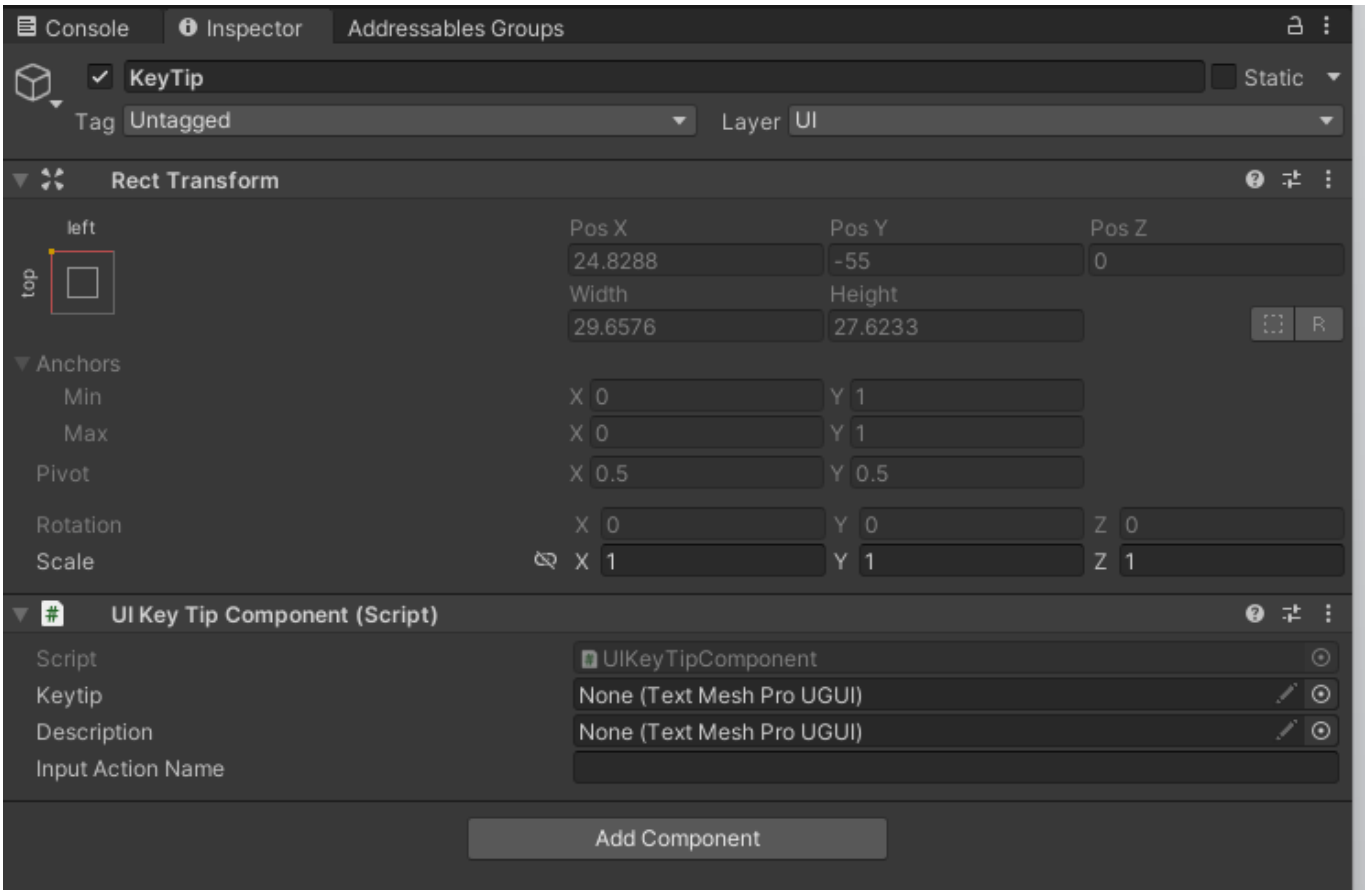
大致思路

1.首先在InputManager中读取当前项目中的所有按键设置并存入一个字典




```
private Dictionary<(string actionMapName, string actionName), InputAction>
actionDictionary = new Dictionary<(string, string), InputAction>();
```

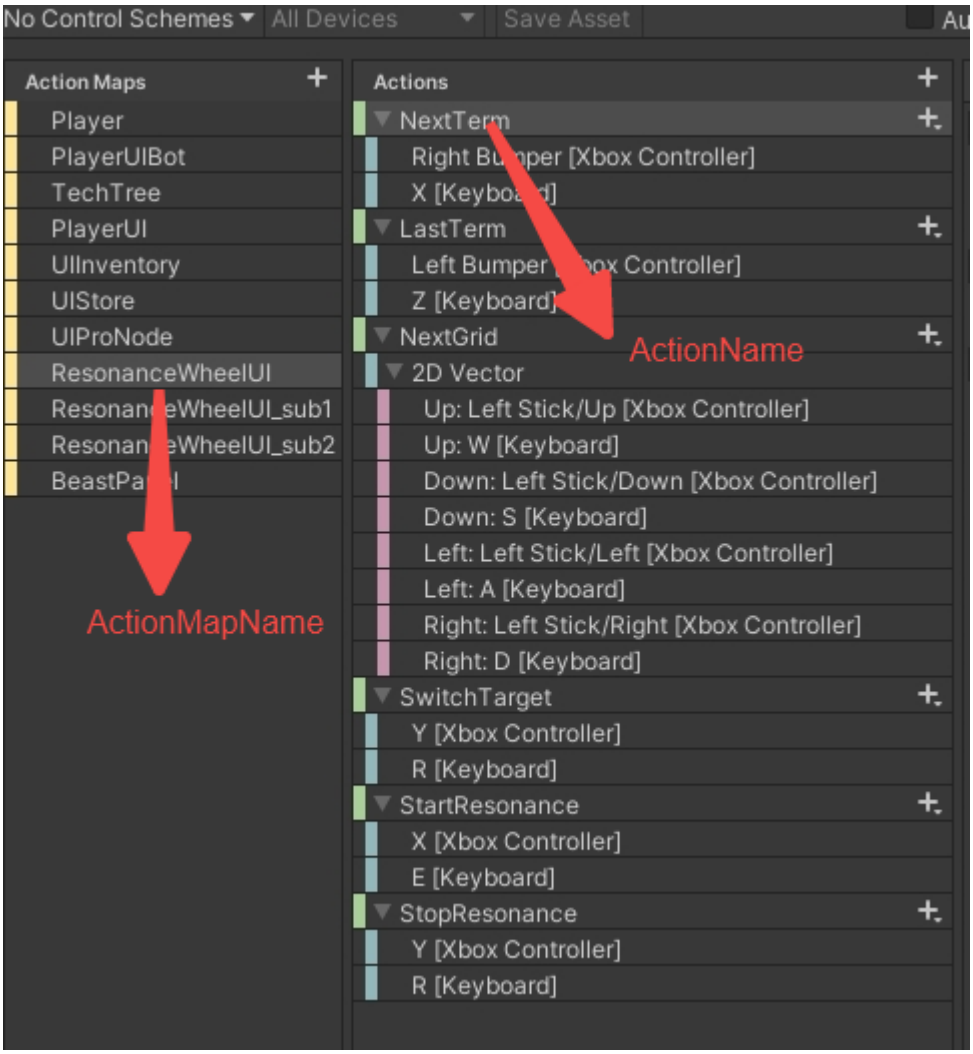
2.UIPanel中的配置



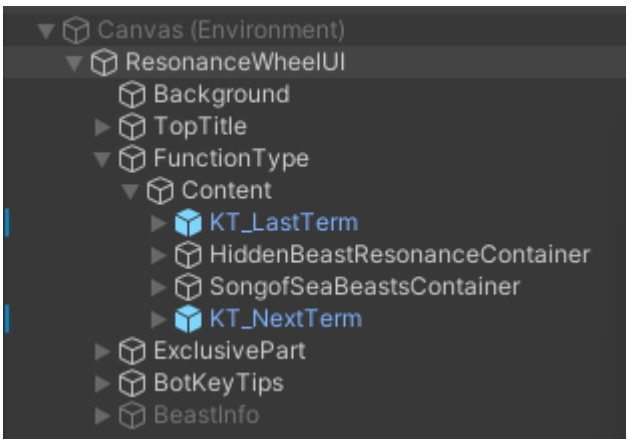
首先在KeyTip的预制体中加入UIKeyTipComponent组件，该组件为MonoBehaviour，原因是方便之后在PanelUI的脚本中awake初始化时找出当前ui中所有的KeyTip预制体，方法用

```
transform.GetComponentsInChildren<UIKeyTipComponent>(true)
```

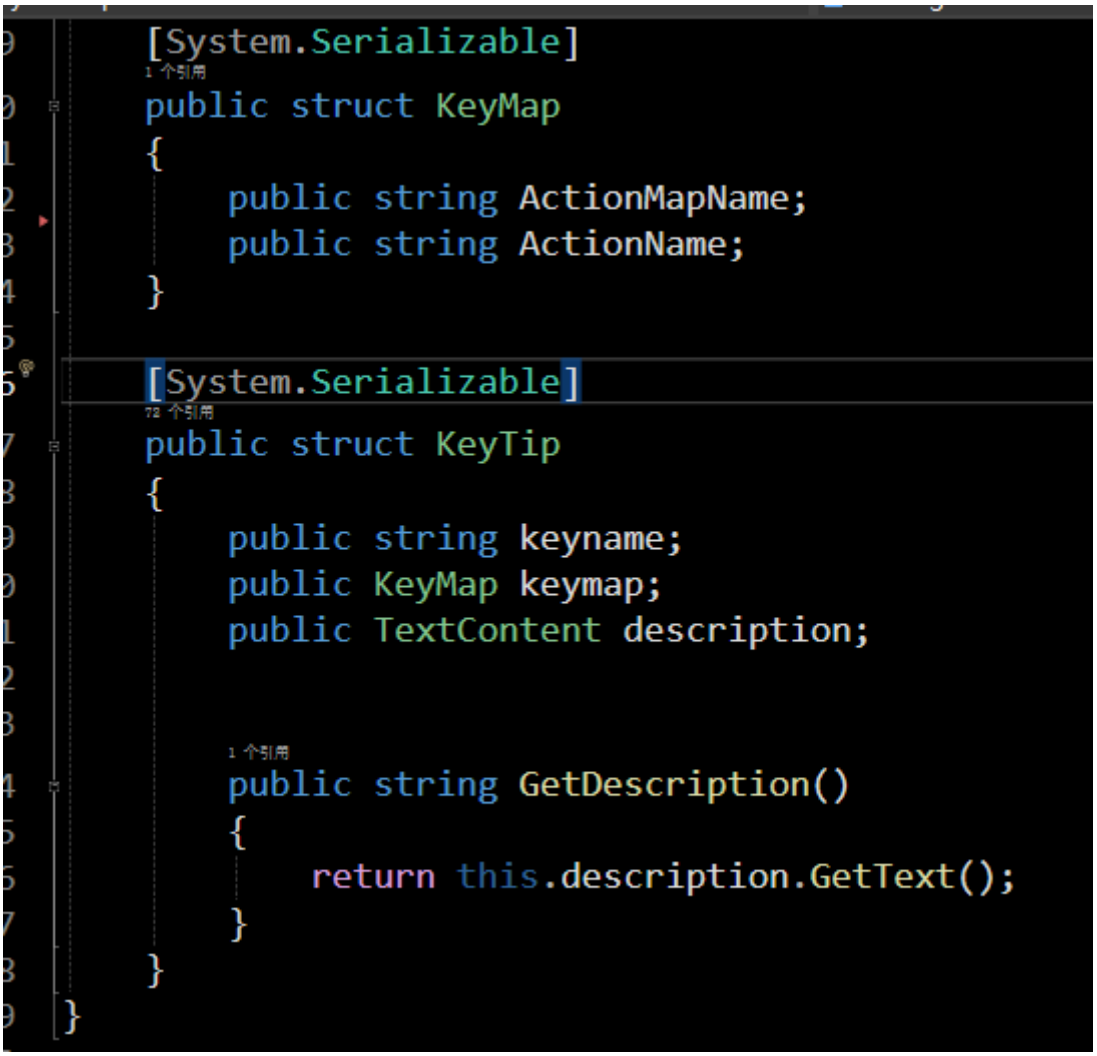
该组件替代原来的UIKeyTip功能



配置KeyTip的命名规范： KeyTip的名字暂定为KT_ActionName



配置json的命名规范：



```
    [System.Serializable]
    1 个引用
    public struct KeyMap
    {
        public string ActionMapName;
        public string ActionName;
    }

    [System.Serializable]
    72 个引用
    public struct KeyTip
    {
        public string keyname;
        public KeyMap keymap;
        public TextContent description;


        1 个引用
        public string GetDescription()
        {
            return this.description.GetText();
        }
    }
}
```

配置KeyTip结构体的json时，

```
  "LastTerm": {
    "keyname": "KT_LastTerm",
    "keymap": {
      "ActionMapName": "ResonanceWheelUI",
      "ActionName": "LastTerm"
    },
```

KeyTip = ActionName KeyTip.keyname = KT_ActionName KeyTip.keymap.ActionMapName = ActionMapName KeyTip.keymap.ActionName = ActionName

3.具体实现逻辑

在UIPanel中 在Awake方法中找到所有UIKeyTipComponent

```
//KeyTips
UIKeyTipComponents = this.transform.GetComponentsInChildren<UIKeyTipComponent>
(true);
foreach (var item in UIKeyTipComponents)
{
```

```
    uiKeyTipDic.Add(item.InputActionName, item);
}
```

在Refresh方法中通过InputManager的字典找到当前ui中每个UIKeyTipComponents所对应的InputAction。找到InputAction后，再获取其bingding的按键字符串，最终赋值到KeyTip预制体的TMPro.TextMeshProUGUI的组件中

```
if (UikeyTipIsInit == false)
{
    KeyTip[] keyTips = inputManager.ExportKeyTipValues(PanelTextContent_Main);
    foreach (var keyTip in keyTips)
    {
        InputAction inputAction =
inputManager.GetInputAction((keyTip.keymap.ActionMapName,
keyTip.keymap.ActionName));
        inputManager.GetInputActionBindText(inputAction);

        UIKeyTipComponent uIKeyTipComponent = uiKeyTipDic[keyTip.keyname];
        if (uIKeyTipComponent.keytip != null)
        {
            uIKeyTipComponent.keytip.text =
inputManager.GetInputActionBindText(inputAction);
        }
        if (uIKeyTipComponent.description != null)
        {
            uIKeyTipComponent.description.text = keyTip.description.GetText();
        }

    }
    UikeyTipIsInit = true;
}
```

InputManager中的一些方法：

```
//该函数读取项目中的InputActionAsset，并写入字典
        public void InitUIInputActionAssets()
        {

GameManager.Instance.ABResourceManager.LoadAssetsAsync<InputActionAsset>
("InputActionAssets", (ia) =>
            {
                lock (actionDictionary)
                {
                    // 遍历所有的 Input Actions
                    foreach (InputActionMap actionMap in ia.actionMaps)
                    {
                        // 遍历当前 actionMap 中的所有 Input Actions
                        foreach (InputAction action in actionMap.actions)
                        {
```

```
                            (string, string) key = (actionMap.name, action.name);
                            actionDictionary[key] = action;
                        }
                    }
                }
            });
        }
```

```
//该函数通过键(string actionMapName, string actionName)找值InputAction
        public InputAction GetInputAction((string, string) key)
        {
            if(this.actionDictionary.ContainsKey(key))
            {
                return this.actionDictionary[key];
            }
            return null;
        }
```

```
//该函数查找InputAction中的所有binding并返回用户能读的按键提示字符串
{
    HashSet<string> keys = new HashSet<string>();
    string t = "";
    // 遍历当前 InputAction 的所有绑定
    var options = InputControlPath.HumanReadableStringOptions.OmitDevice |
InputControlPath.HumanReadableStringOptions.UseShortNames;
    string humanReadableString;
    foreach (InputBinding binding in inputAction.bindings)
    {

        humanReadableString = InputControlPath.ToHumanReadableString(binding.path,
options);
        if (Config.inputDevice == Config.InputDevice.Keyboard &&
binding.path.StartsWith("<Keyboard>"))
        {
            keys.Add(ExtractString(humanReadableString));
        }
        else if (Config.inputDevice == Config.InputDevice.XBOX &&
binding.path.StartsWith("<XInputController>"))
        {
            keys.Add(ExtractString(humanReadableString));
        }
    }

    foreach (var item in keys)
    {
        t += item;
    }
```

```
    return t;
}
```