Visual Editor Tools(VET)

目标

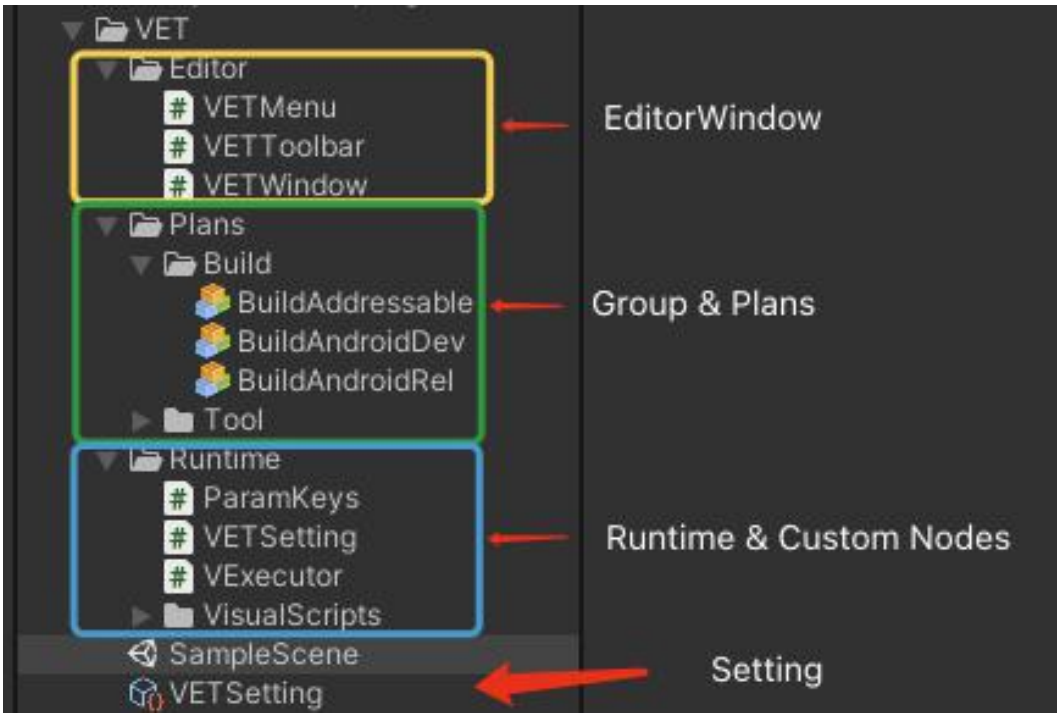
VET 是一个极简的工具框架，目的在于提出一个统一思路，来解决项目中 Editor 工具链维护困难、不够直观的问题,VisualScripting（VS）是一个好的机制，但是它当前版本不支持 Editor 下运行，稍作处理即可

特性

1，基于节点，逻辑清晰
2，复用性强，避免重复造轮子
3，扩展性强，扩展规则大体同 VS
4，易于维护，集中式管理
5，支持命令行及参数

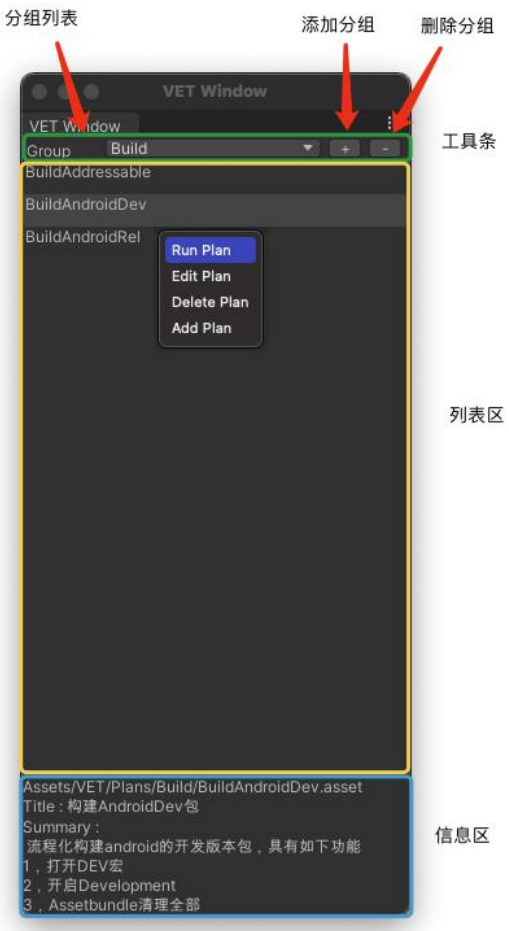
开始

1，目录结构

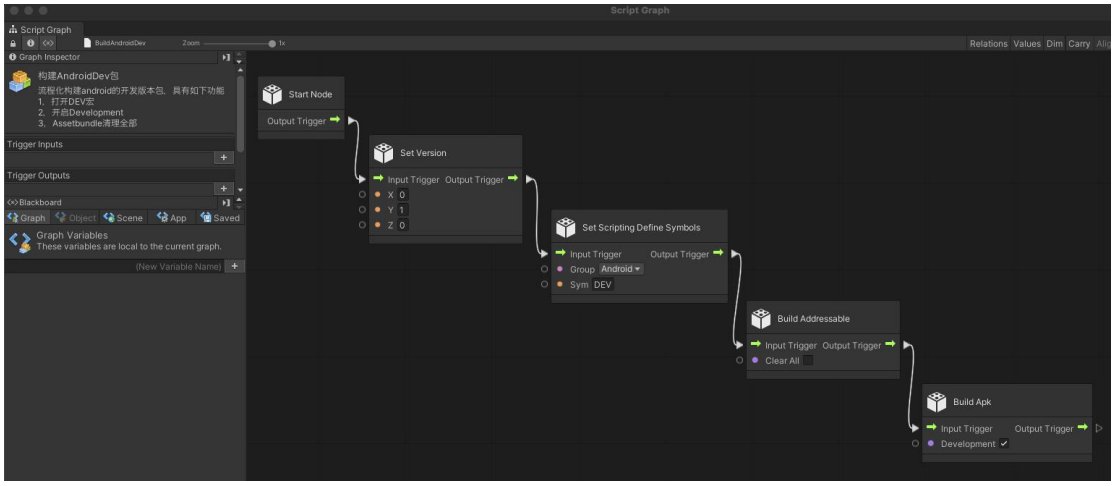


2 ,VETSetting(右键->Create->VET->VETSetting)，这里只需要设置 Plan 路径，以便支持玩家自定义 VET 位置

3 ， VETWindow(Window->VET->VETWindow



4, Edit Plan (使用 unity 内置的 Visual Script 机制), 特别的, 第一个节点必须是 Vet/Start



命令行调用

[UnityDir]/Unity -quit -batchmode -projectPath [ProjDir] -executeMethod VET.VExecutor.BatchRun -logFile [xxx.log] group=[] plan=[] XXX1=VVV1 XXX2=VVV2

自定义 Node

可以按照 VisualScript 规则随意定义，但是 EditorTool 建议如下定义，便于统一管理

```
[UnitCategory(fullName:"VET/Build/SetVersion")]
Kyo Chow
public class SetVersion : BaseBuildNode        Base class
{
    public ValueInput X;
    public ValueInput Y;
    public ValueInput Z;

    0+4 usages  Kyo Chow
    protected override void Definition()
    {
        base.Definition();
        X = ValueInput(key:"X", default:"0");
        Y = ValueInput(key:"Y", default:"0");
        Z = ValueInput(key:"Z", default:"0");
    }

    0+1 usages  Kyo Chow
    public override void Process(Flow flow)
    {
        if (Variables.IsDefined(variable:ParamKeys.KEY_VERSION))
        {
            Debug.Log(message:$"process setversion {Variables.Get(variable:Pa
        }
        else
        {
            Debug.Log(message:$"process setversion self {flow.GetValue(X)},
        }
    }
```