# Leibt说明文档

## 一、窗口概览

如下图1-1所示。

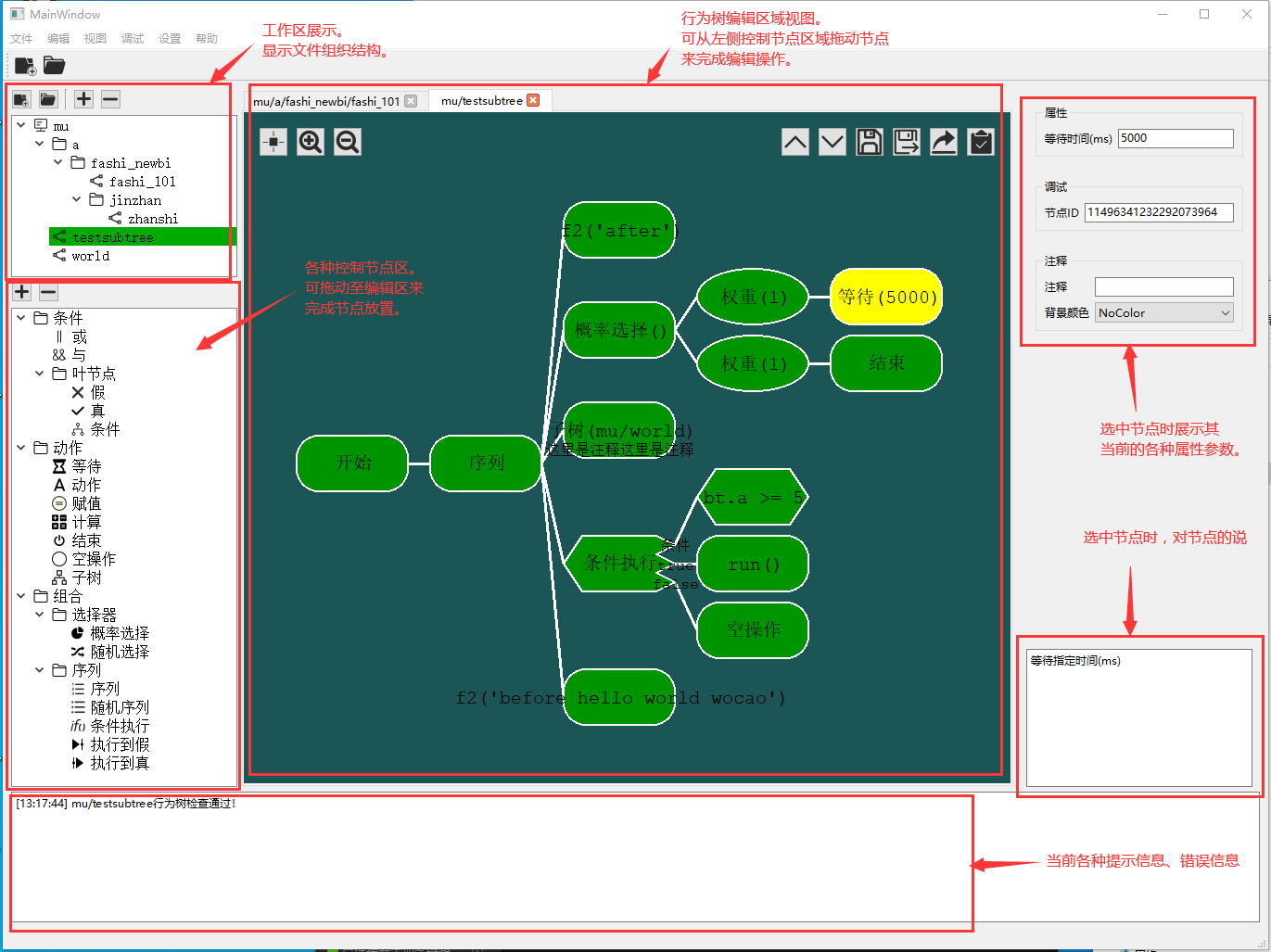


图1-1 窗口概览

## 二、各子窗口说明

### 1 工作区

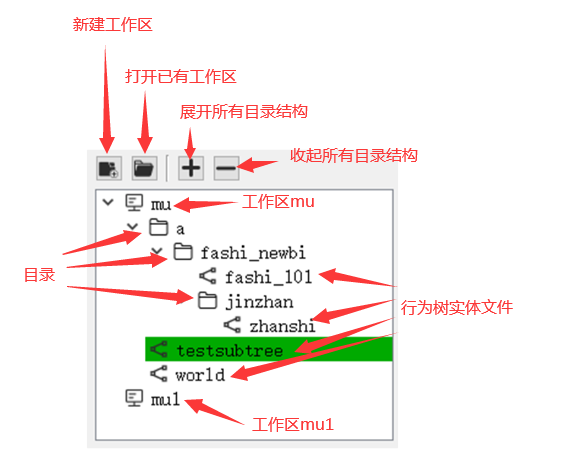


图2-1 工作区子窗口

### 2 视图区按钮说明

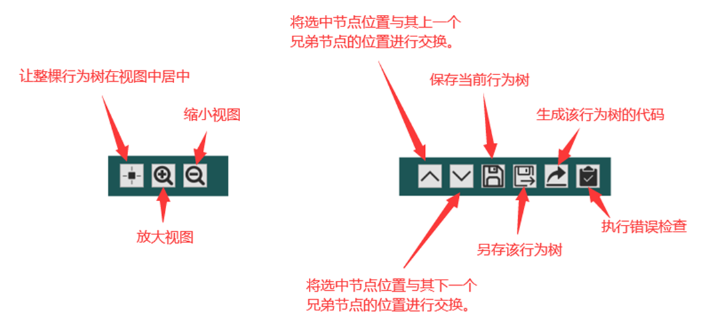


图2-2 视图区按钮

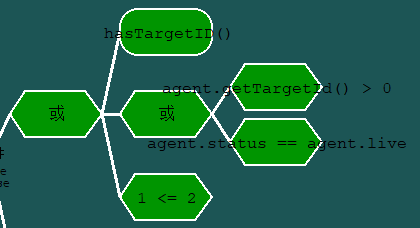
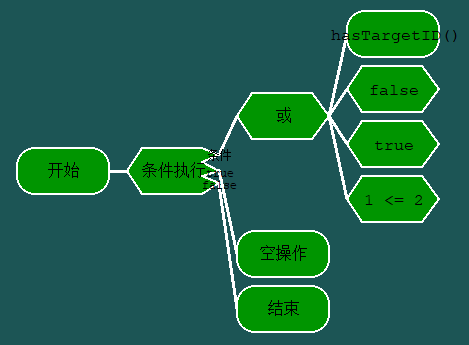
。。。。。。

## 三、各种控制节点功能

### 1 条件类

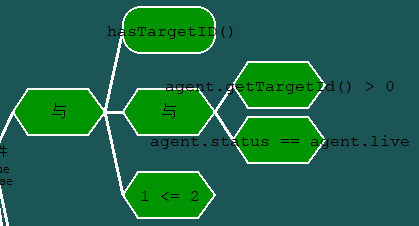
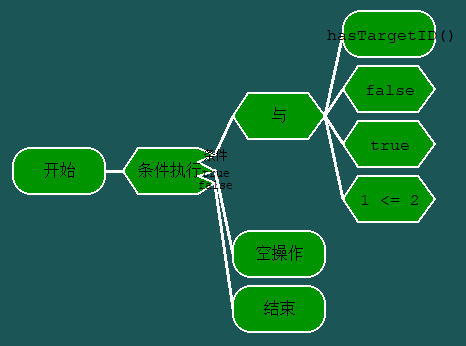
#### 《或》：

或节点是作为《条件》节点的条件使用。它的所有子节点中，只要有一个节点为真，则或节点返回为真。判断从上到下按序进行，如果期间碰到结果为真的子节点，则直接返回，不再进行后续判断。或节点的子节点可以是其他条件类节点。



#### 《与》：

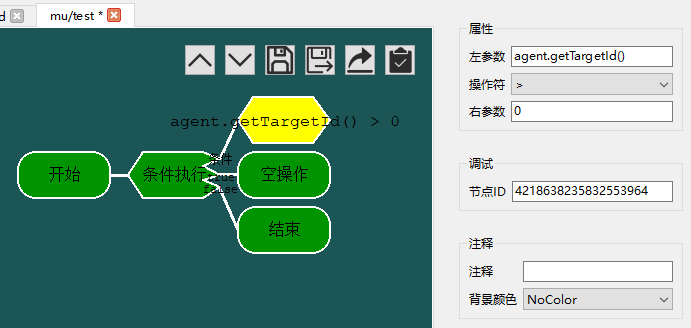
与节点是作为《条件执行》节点的条件使用。它的所有子节点中，只要有一个节点为假，则与节点返回为真。判断从上到下按序进行，如果期间碰到结果为假的子节点，则直接返回，不再进行后续判断。与节点的子节点可以是其他条件类节点。



#### 《条件》：

条件节点是作为《条件执行》节点的条件使用。它没有子节点。导出的代码中，目前就是暴力地直接判断“左参数 操作符 右参数”，操作符支持“>=、==、!=、<、<=、>、>=”。

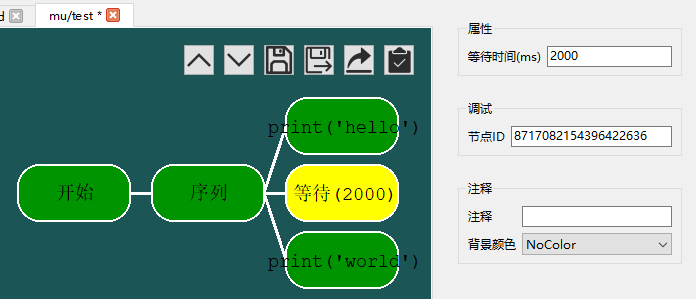
所以参数中可以调用代理的接口；比如左参数为“agent.getTargetId()”，操作符为“>”，右参数为“0”，则翻译为“agent.getTargetId() > 0”，表示调用代理的getTargetId()接口并判断它的返回是否大于0。其中前缀“agent.”不能少（目前还不够智能）。参数中也可以使用行为树中自定义的临时变量，比如“bt.currentTargetId”，前缀“bt.”表示使用临时参数（不能少）。因为是直接翻译为python代码，所以也可以随意在接口中使用各种参数，比如“agent.getIDist(agent.getTargetId(), bt.currentId())”。



### 2 动作类

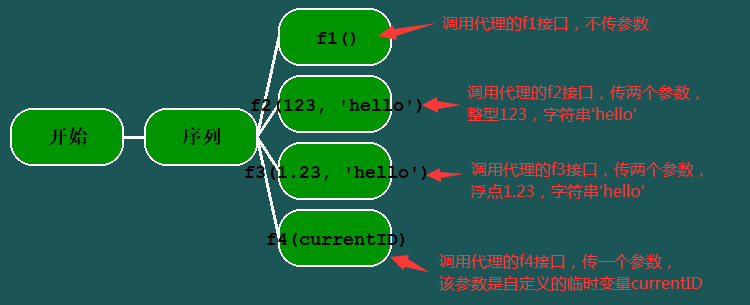
#### 《等待》：

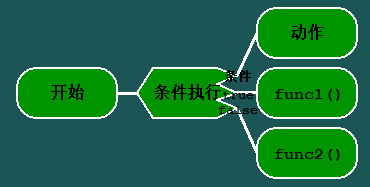
等待节点，将使行为停留在该节点指定时间。指定时间后，继续从该节点继续执行。比如下图中，先执行“print(‘hello’)”，然后等待2000毫秒，再执行“print(‘world’)”。



#### 《动作》：

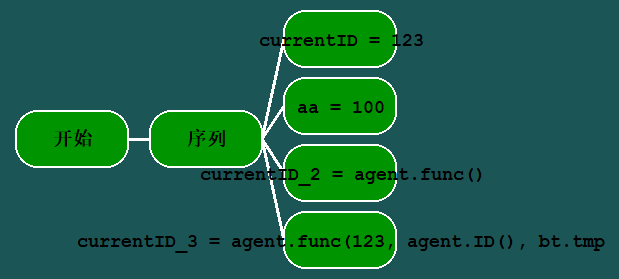
只能执行指定的代理的接口。接口的参数只能是“临时参数、字符串、整型、浮点数”四种。动作节点也可以作为条件执行节点的条件，它的返回值作为真假值的参考。





#### 《赋值》：

执行“左参数 = 右参数”，同时将左参数设置为行为树的临时变量。左参数填写要满足变量名的规范。因为是直接翻译为python代码，所以右参数可以调用各种接口等，如下图所示。

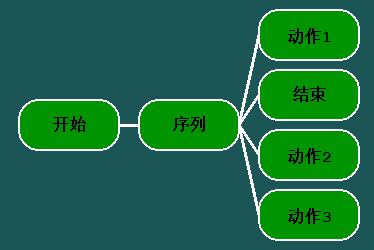


#### 《计算》：

目前计算节点的功能和《赋值》节点的功能一毛一样。不排除未来会进行差异化扩展。

#### 《结束》：

遇到结束节点时，行为树将直接退出，不再执行后续的逻辑。比如下图，执行“动作1”之后，遇到结束，“动作2”、“动作3”将不会执行。

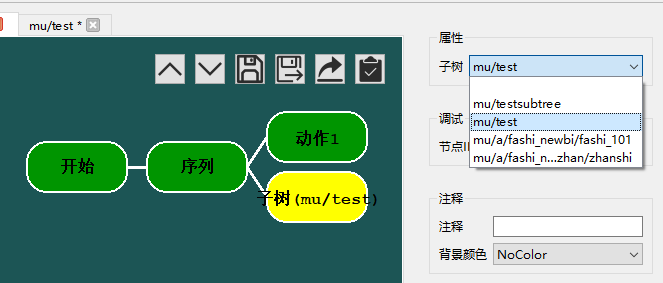


#### 《空操作》：

什么都不干，一般作为占位使用。

#### 《子树》：

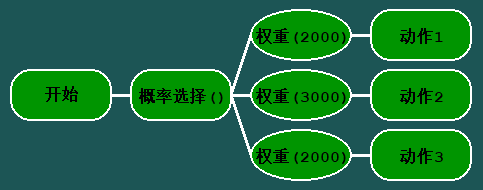
引用其他已存在的子树，可以在属性面板中选择可引用的其他行为树。不能引用自己。逻辑执行直接进入该子树的逻辑。注意，如果子树中有遇到等待节点，父树不会等待，而是继续执行；但是下一帧执行到子树时，会判断子树的等待时间，如果还处于等待状态，则子树会直接返回WAIT状态，而父树会继续执行。子树的OVER状态也不会影响父树。



### 3 组合类

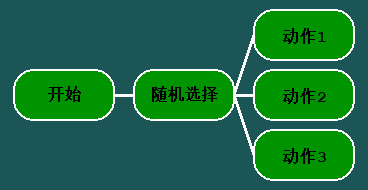
#### 《概率选择》：

根据权重随机选择一个分支进行执行，可以点击权重节点来编辑各个分支的权重。



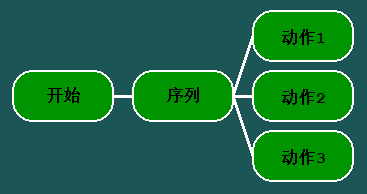
#### 《随机选择》：

从所有子节点中，随机选择一个分支进行执行。相当于，把概率选择节点的所有子节点的权重设置为一样。



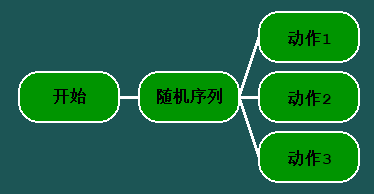
#### 《序列》：

从上到下按顺序执行全部子节点。



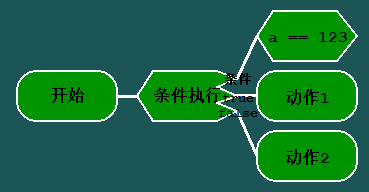
#### 《随机序列》：

按照随机顺序执行所有子节点。



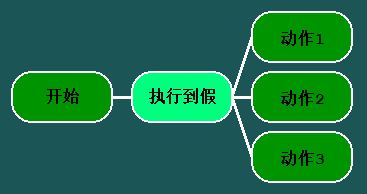
#### 《条件执行》：

根据条件（第一个）子节点的返回值，如果为“真”则执行第二个子节点，否则执行第三个子节点。



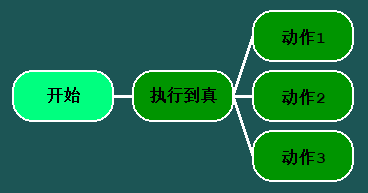
#### 《执行到假》：

从上到下按顺序执行全部子节点，直到遇到某个节点返回为“假”。



#### 《执行到真》：

从上到下按顺序执行全部子节点，直到遇到某个节点返回为“真”。



## 四、程序开发篇

### 1 目录说明

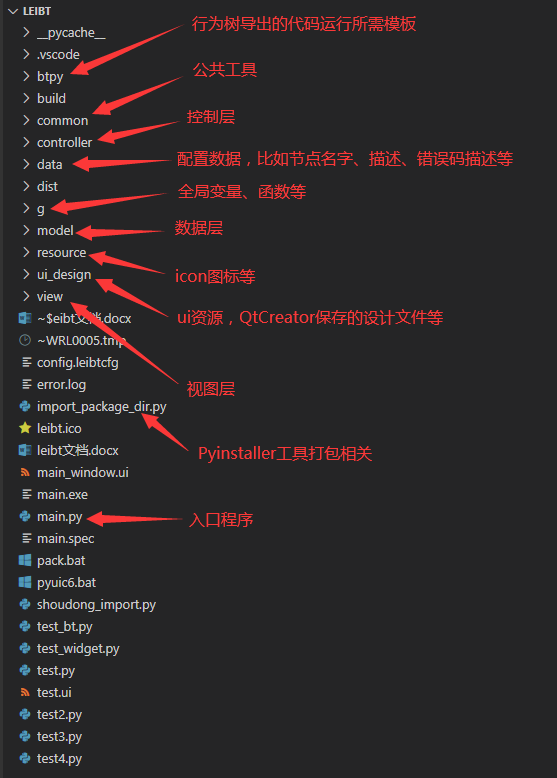


图4-1 目录说明

软件使用mvc模式开发，数据层（model）与视图层（view）解耦，二者通信需要通过控制层（controller）。

视图层目录说明：



图4-2 视图层目录说明

数据层目录：

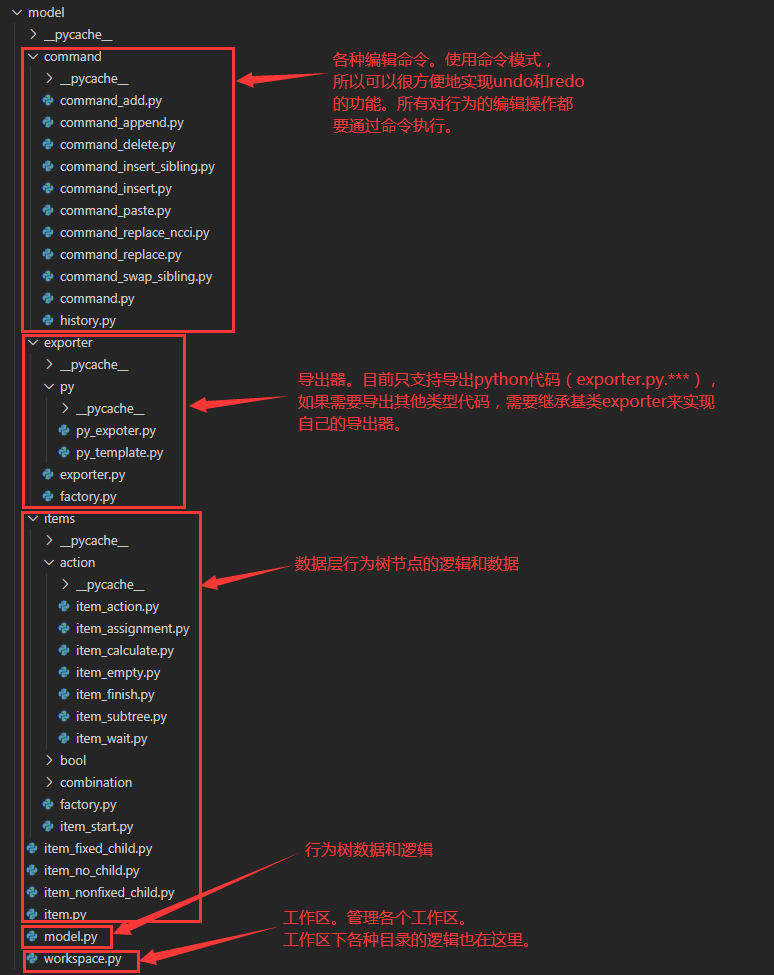


图4-3 数据层目录说明

View层中，view/widget\_scene.py:WidgetScene是行为树编辑场景，行为树的组织结构必然也是树状（view/widget\_scene.py:SceneTree），树中的每一个节点都是view/scene\_tree\_nodes/scene\_node:SceneNode的子类。每种类型的控制节点对应一种子类。可自定义各种节点绘制。

Model层中，一棵行为树的数据结构显然也是树状结构（model/model.py:Model），树中的每个节点都是model/item.py:Item的子类。每种类型的控制节点对应一种子类。子类节点可自定义各种节点的逻辑，比如放置子类、替换等。

Model层中，行为树文件的组织方式与视图层中工作区子窗口的结构一样，是一个目录状的组织结构。详情可见model/workspace.py。如下图所示：

WorkspaceName

工作区

ItemDirectory

目录

ItemDirectory

目录

ItemDirectory

目录