## 概述

### 工具介绍

**1）灵活分类树（常规）**

代码：

|  |  |
| --- | --- |
| 灵活分类树 | p\_FlexiblePageTree.h  p\_FlexiblePageTree.cpp |
| 树控件 | i\_FPT\_Branch.cpp  i\_FPT\_Branch.h  i\_FPT\_Leaf.cpp  i\_FPT\_Leaf.h |
| 树设置 | c\_FPT\_Config.cpp  c\_FPT\_Config.h  w\_FPT\_Config.cpp  w\_FPT\_Config.h  w\_FPT\_Config.ui |

所需环境：

|  |  |
| --- | --- |
| 排序控制器 | c\_ObjectSortData. cpp  c\_ObjectSortData.h  p\_ObjectSortController.cpp  p\_ObjectSortController.h |

**2）灵活分类树（常规+种类分支）**

代码：

|  |  |
| --- | --- |
| 灵活分类树 | p\_FlexibleClassificationTree.h  p\_FlexibleClassificationTree.cpp |
| 树设置 | c\_FCT\_Config.cpp  c\_FCT\_Config.h |
| 种类分支 | c\_FCT\_Classify.cpp  c\_FCT\_Classify.h  w\_FCT\_Classify.cpp  w\_FCT\_Classify.h  w\_FCT\_Classify.ui  w\_FCT\_ClassifySelector.cpp  w\_FCT\_ClassifySelector.h  w\_FCT\_ClassifySelector.ui |

所需环境：

|  |
| --- |
| 灵活分类树（常规）的全部代码 |

### 使用注意

|  |
| --- |
| 由于 树配置 和 资源数据 传的都是指针，控制修改后，是**直接生效**的。  但你仍然需要调用 set方法，确保数据在树中及时刷新。  （传指针有一个优势，就是能无限继承，但使用时注意用dynamic\_cast强转。） |

## 结构

### 快速理解

1. 首先把这棵树看成一个封装的黑盒结构：输入数据、输入配置， 输出数据、配置。



2. 黑盒内部结构，对数据操作分为： 增删改查。

增加 x

删除 x

修改 √ -> 只修改 type 数据

查询 √ -> 四种方式查询排序



3.如果外部数据被修改了，那么树也要跟着刷新一下，但是树如果直接重建，性能代价巨大。所以出现了函数： outerModify。

某个具体一项修改了名称或类型，调用一下即可。（id不可修改）



4.除此之外，右键菜单包含了复制粘贴等按钮选项，由于该树不提供额外操作功能，所以只发信号。这部分名称为“叶子接口”

### 概念

**分支：**分支即 树枝组 的分配方式。在rebuildTreeData重建树时，会根据分支的规则，建立相应的树枝。

**分支模式：**分支模式即可以随时调整切换的分支设置。

灵活分类树（常规）包含： Id分支、名称分支。

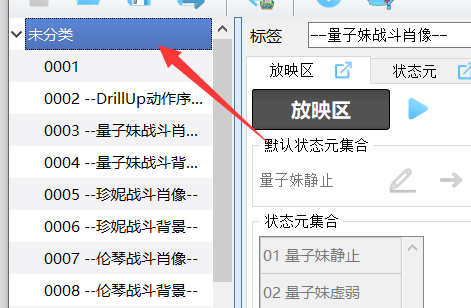
灵活分类树（常规+种类分支）包含： Id分支、名称分支、两个种类分支。

**种类（classify）**：专有名词，表示自定义的种类划分。

注意，种类（classify）和 类型（type）不是同一类数据。

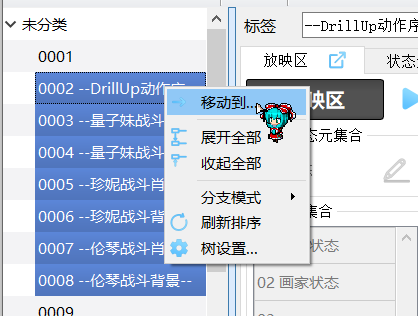
### 应用功能

GIF动画序列编辑器中，默认都属于未分类状态。

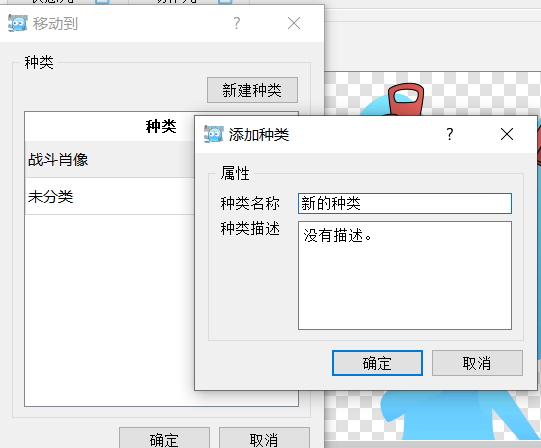


你可以按住shift或者ctrl多选 动画序列的项，

然后右键菜单，点击“移动到..”。



在种类选择器中，添加一个 种类。



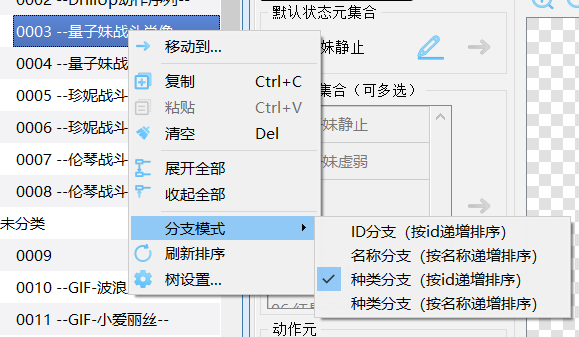
添加种类后，移动到指定种类，则全部移动过去了。



需要说明的是，这些分类可以保存的工程文件中，并且后期打开能够识别。

但是rmmv编辑器中，并不能识别这些分类数据。

这些操作都必须通过 右键点击 树中的项才能显示的功能。



## 脚本细节

### 刷新机制

在使用树时，你不需要手动去控制调用这些刷新函数。

刷新分为三个阶段：

1）loadSource

该阶段会重读所有数据，并且创建叶子，然后执行2), 3)。

2）rebuildTreeData

该阶段会对叶子进行排序，重建树枝，然后执行3)。

3）refreshTreeData

该阶段只刷新树枝和叶子的文本显示。