Aegisub 自动化简明教程

本文的知识背景要求

本文或许会需要用到的知识有：

字幕方面

- **Ass 的 Tag**
  请阅读以下网页
  ASS标签
- **自动化的基本操作方法和简单的例子** 请阅读以下几个网页)
  卡拉 OK 模版执行器
  一个简单的例子
  使用数学表达式
- **了解模板修饰语、内联变量、内置函数、字幕行数据结构等等**
  请阅读以下网页
  卡拉 OK 模版执行器
  Template 修饰语
  内联变量（$变量）
  Code 区和 Code 行的规则
  Code 区/行的执行环境的内容
  karaskel.lua
- **使用自动化的一定经验**

编程方面

- **知道「变量」的概念**
- **了解「变量」的类型**
  如「全局变量」与「局部变量」
- **懂得「数据类型」**
  如「整形量」与「字符串量」
- **一定的编程经验**

自动化的基本逻辑

一个典型的字幕行，如下所示：

```
Format: Layer, Start, End, Style, Name, MarginL, MarginR, MarginV, Effect, Text
Dialogue: 0,0:00:22.01,0:01:30.00,Default,,0,0,0,karaoke,今天是个好日子
```

我们可以看到，一行字幕不仅仅包括其显示的内容，即最后的 Text，还包括前面的 Layer, Start, End, Style, Name, MarginL, MarginR, MarginV, Effect, Text 等属性信息。对字幕行的修改，一是修改其内容，二是修改其属性。

自动化的基本逻辑就是用 Template 行替换掉 Karaoke 行中的 K 标签，通过 Code 行修改 Karaoke 行的各种属性，然后将其输出为新的字幕行，插入字幕文件。

在自动化中，有三种行，Template 行，Code 行，Karaoke 行。每次「应用卡拉 OK 模板」的时候，模板执行器会依据这三种行，生成新的 Fx 行。Template 行和 Code 行本来就是被注释的，模板执行器在执行完毕之后，也会把 Karaoke 行注释掉，所以最后起作用的，只有新生成的 Fx 行。

下面我们来详细地讨论这三种行，顺便说明模板执行器的大概工作流程。

Code 行

Code 行用于声明自定义变量，声明函数，修改已经存在的变量的值等等。它一般看起来像这样：

```
Format: Layer, Start, End, Style, Name, MarginL, MarginR, MarginV, Effect, Text
Comment: 0,0:00:00.00,0:00:05.00,Default,,0,0,0,code once,X=42  //声明变量
Comment: 0,0:00:00.00,0:00:05.00,Default,,0,0,0,code once,function shuzi1() Y=23 r
eturn 32 end //声明一个函数，在其中声明一个变量，并返回整形数字32
Comment: 0,0:00:00.00,0:00:05.00,Default,,0,0,0,code once,function shuzi2() return
 "32" end //声明一个函数，返回字符串"32"
Comment: 0,0:00:00.00,0:00:05.00,Default,,0000,0000,0000,code line,fxgroup.funky =
 line.actor == "funky" //改变行的 fxgroup.funky 布尔变量的值
Comment: 0,0:00:00.00,0:00:05.00,Default,,0,0,0,code syl,line.start_time = 22222
//改变行的开始时间
```

我们可以看到，在 Code 行的 Text 中，写的都是 Lua 代码，用于声明变量与函数，或者改变已有的变量的值。由于必须要写在一行之内，所以本来应该换行的地方，都被空格代替了，不过仍然能正常运行。上面三个例子中，修饰语有所不同，这会使得它们有不同的执行顺序和次数，这里不打算详细讨论，请阅读卡拉OK模板执行环境和顺序，并实际动手实验。Code 行里声明的变量和函数，都不会直接起作用，而是要在 Template 行中显式地使用它们。例外是最后两个 Code 行，他们直接改变 Aegisub 自动化环境提供的变量，从而影响新产生的字幕行的属性。这两种例子我们下面都会看到。

Template 行

Template 行一般被称为模板，就像我们常用的 PowerPoint 模板一样，它具有一个骨架，以及可以灵活改变的细枝末节。它看起来通常是这样的：

```
Format: Layer, Start, End, Style, Name, MarginL, MarginR, MarginV, Effect, Text
Comment: 0,0:00:00.00,0:00:05.00,Default,,0,0,0,template syl,{\pos($sx,$sy)}  //拆字成行
Comment: 0,0:00:00.00,0:00:05.00,Default,,0,0,0,template line,{\r\k$kdur\t($start,$end,\1c&H00FF00&)\t($start,$mid,\fscy120)\t($mid,$end,\fscy100)} //添加一个简单的变色和缩放效果
Comment: 0,0:00:00.00,0:00:05.00,Default,,0,0,0,template line,!Y!!shuzi1()!  //给每个音节前面加上2332
Comment: 0,0:00:00.00,0:00:05.00,Default,,0,0,0,template line,!X!!shuzi2()!  //给每个音节前面加上4232
```

在以上例子中，修饰语 syl、line 的不同效果，读者一试便知，在此不做讨论。模板中以 `$` 开头的是内联变量，即 Aegisub 的自动化环境提供的变量，索引在内联变量；以 `!!` 夹起来的即是引用变量，或者调用函数，它们有些是 Aegisub 的自动提供的，有些是使用者在 Code 行中自定义的；剩下的则是普通的字符。模板如果直接用于替换 Karaoke 行的 K 标签，是无法达到预期效果的，因为本身不是合法的 Ass Tag，在渲染的时候会报错或者被忽略。模板执行器会首先计算一行模板中的变量与函数的具体值，然后就会得到合法的 Ass Tag，就像下面这样。

```
Text
{\pos($sx,$sy)} >>>>> {\pos(233,333)}  //此处值为假设值
{\r\k$kdur\t($start,$end,\1c&H00FF00&)\t($start,$mid,\fscy120)\t($mid,$end,\fscy100)} >>>>> {\r\k23\t(23,46,\1c&H00FF00&)\t(23,35,\fscy120)\t(23,35,\fscy100)}  //此处值为假设值
!Y!!shuzi1()! >>>>> 2332
!X!!shuzi2()! >>>>> 4232
```

可以看到，对于内联变量和引用变量，直接获得变量的值；当调用函数时，则获得返回值。对于不同的行和音节，变量的值和函数的返回结果可以不同，于是就对其 K 标签的替换内容也就不同。这样通过合适的计算之后，就可以给每个音节和行都添加适宜的效果。

在这个例子中我们可以看到，单独用一行 Code 行声明的变量固然可以引用，在函数中声明的变量也是可以引用的，如上面的函数 `shuzi1()` 中的 `Y`。这是因为 Lua 中所有的变量默认都是全局变量，因此，无论在哪里声明变量，都可以在模板行中引用。相对的，为了避免这种特性引起的重名问题，使用者可以定义局部变量，方法自寻。同时，由最后两个模板我们可以知道，返回整型数和字符串一般具有相同的效果，读者可以根据需要灵活运用。

Karaoke 行

Karaoke 行即是带有 K 标签的字幕行，一般像下面这样。

```
Format: Layer, Start, End, Style, Name, MarginL, MarginR, MarginV, Effect, Text
Dialogue: 0,0:00:22.01,0:01:30.00,Default,,0,0,0,karaoke,{\k71}今{\k71}天{\k72}是{\k71}个{\k71}好{\k72}日{\k72}子
```

当应用模板时，其内容被 Template 行按规则替换，属性被 Code 行修改，然后被注释掉。

Aegisub 自动化环境提供的变量

刚才我们两次提到了「Aegisub 的自动化环境提供的变量」，现在来详细讨论一下。

我们刚才见到了 `fxgroup.funky` 、 `line.start_time` 、 `$sx` 、 `$sy` 等变量，它们都是在模板执行器开始运行时，根据 Karaoke 行信息计算好的变量。其中后两者是内联变量，它们的值是当前行或者当前音节的相关信息，是只读的，仅仅是为了让使用者方便的引用其值，索引在上文已经给出。前两者则是 Aegisub 自动化环境创建的一系列变量的一部分，这套变量在karaskel.lua、Code 区和 Code 行的规则和Code 区/行的执行环境的内容中各有提及，也有不同的特性和使用限制。一方面来说，这套变量相比内联变量更为丰富和全面，方便使用者在自己的 Lua 程序中进行相关计算；另一方面来说，这套变量的值可以被改变，同时会影响新创建的行的效果，给了使用者方便的工具来编辑新字幕行。如刚才提到的 Code 行:

```
Comment: 0,0:00:00.00,0:00:05.00,Default,,0,0,0,code syl,line.start_time = 22222
//改变行的开始时间
```

这个 Code 行具有修饰语 syl，所以对每个音节会执行一次，作用是修改了变量 `line.start_time` 的值，当这个值被修改的时候，新字幕行的开始时间也会被修改为同样的值。这是 Aegisub 自动化环境提供的便利。上面这个 Code 行和以下写法是等价的:

```
Comment: 0,0:00:00.00,0:00:05.00,Default,,0,0,0,code once,function changestart() l
ine.start_time = 22222 return "" end  //定义一个函数
Comment: 0,0:00:00.00,0:00:05.00,Default,,0,0,0,template syl,!changestart()!  //调
用函数
```

可以看到，给 Code 行安排合适的修饰语，有时能达到在模板行里手动调用同样的效果。具体的模板执行过程，请参阅卡拉OK模板执行环境和顺序。

总结

总的来说，自动化的工作，就是使 Code 行和 Template 行协同工作。使用者只要明白了其工作逻辑，就可以利用自动化环境提供的种种变量，方便的进行计算和修改，做出漂亮的效果。如 `AutoTags` ， `Yutils` 等。