

Ayano\_Aishi 同人剧情组、亿绪联合协会 支持

# “Yayin” Story Player<sub>(Ver0.5.0)</sub> 雅音 剧情播放器<sub>(版本 0.6.0)</sub>

Story Player Official Language<sub>(Ver0.6.0)</sub>  
播放器官方语言<sub>(版本 0.6.0)</sub>

## 指导文档

2021 年 6 月 11 日

## 赠予参加内部测试的人员



“我们联合”

——您在亿绪联合协会的相关项目中做出了不可或缺的贡献。

## 前言：

《明日方舟》作为近年来较火热的一款二次元游戏，它动人心弦的故事一直为所有玩家所津津乐道，因此不少的明日方舟玩家尝试进行剧情的二次创作。这些玩家进行二次创作的平台通常是 PPT，或者是 Adobe PR 这类视频剪辑软件，但也有不少玩家会选择使用 Unity 或者 UE 引擎进行创作。

这些制作方式很显然有不少问题——麻烦、繁琐。例如，一页一页的写 PPT 或一个对象一个对象的对 PR 时间轴添加素材。这样的低效率制作不仅浪费时间，甚至还会因为手动添加会无法保证元件严格对齐，导致没办法百分百的使格式统一，从而导致剧情的质量起伏太大；又如使用大型游戏开发引擎，只为了展示自己的剧情就大动干戈的使用这些引擎实在是杀鸡用牛刀。并且由于引擎自身需要占用一定空间，就会导致为他人展示寥寥几个剧情就会占用很大空间，并且还对低配置设备不够友好。这些问题非常不利于剧情的传播。

所以面对这样的局面，我们很希望能开发一种程序——能够识别一定的定义语言，将其转换成剧情内容。玩家既可以向他人直接分享自己用这种定义语言所写的剧情文件，以供他人使用相同程序在本地播放，也可以快速录制程序页面为 MP4 视频，供广大同好欣赏。

于是，我们开发了 YSP，以及能让用户能够用最简单的符号来描述对白、场景、音频的变化的语言——SPOL。本文档即是 SPOL 语言的官方指导文档。本文档仅用来详细的描述 SPOL 语言的定义，不作为通行教程使用。

在编写本文档过程中，我假定本书面向的读者是没有任何程序编写基础的，因此本书的语言尽可能的简单，详实，拥有许多实际上并不必要的表述与过多的解释。

对于用 Python 编程，在一开始我是十分抵触的，因为我本人已经用了七八年的 C、C++ 语言。Python 的这一套语法结构实在不是 C 语言用户一时半会就能接受的。但是当我真正静下心来学 Python 之后，我发现 Python 相比 C 衍生的任何语言来说，的确是一个干净清爽，编写效率较高的语言。于是我就进一步学习了 Python，因此 YSP 是用 Python 开发的。我也尽力让 SPOL 成看起来像 Python。于是我选用了 >>> 作为一般情况下会最多出现的标识。

本项目受到了本人 Python 课程的徐新艳老师的支持与指导，我有许多 Python 的基本知识都是由她教授的。

本项目还受本人的兴趣爱好组织亿绪联合会内部分成员与部门的支持，没有他们参与测试，我想程序的 Bug 可能会比功能还多。

本人只对程序本体拥有所有权，程序内大部分素材都搜集自互联网，相关版权归鹰角网络所有。

青雅音

2021 年 4 月 26 日

## 目录

第零章：基本介绍.....	5
0.1 程序基本结构和使用介绍.....	5
0.2 SPOL 基本介绍.....	6
0.3 程序运行环境介绍.....	6
0.4 本文档着色情况基本介绍.....	6
第一章：SPOL 入门.....	7
1.1 编写环境.....	7
1.2 行首指示符与控制器.....	7
1.2 章节习题.....	9
第二章：文档标准、注释和标题.....	10
2.1 文档标准.....	10
2.2 文档单行注释.....	10
2.3 文档多行注释.....	11
2.4 标题控制器.....	11
2.5 章节习题.....	12
第三章：背景、音乐、音效控制器.....	13
3.1 背景控制器.....	13
3.2 音乐控制器与音效控制器.....	14
3.3 章节习题.....	14
第四章：讲述控制器.....	16
4.1 基本型单人讲述控制器.....	16
4.1.1 定义.....	16
4.1.2 用作旁白.....	16
4.1.3 用作独白.....	17
4.2 基本型多人讲述控制器.....	17
4.2.1 定义与基本对话.....	17
4.2.2 用作对齐手段.....	18
4.3 分离型讲述控制器.....	18
4.4 空场控制器、行进驻留控制器.....	19
4.4 章节习题.....	20
第五章：自由文本控制器.....	21
第六章：选择分支控制器.....	22
6.1 剧情分支控制器（大分支控制器）.....	22
6.2 对话分支控制器（小分支控制器）.....	23
第七章：剧情资源使用.....	25
其他致谢名单（排名不分先后）.....	26

# 第零章：基本介绍

## 0.1 程序基本结构和使用介绍

文件夹下的 YSPMain.exe 为 YSP 播放器的主程序。文件夹下的 click\_run.exe 为接受双击打开剧情的链接程序。这就是说，如果您想先打开播放器，在播放器里选择剧情播放，那么请您使用 YSPMain.exe<sup>1</sup>；如果您想通过双击剧情文件来播放剧情，那么请您在设置默认应用的时候设置启动程序为 click\_run.exe。click\_run.exe 仅用于识别 Windows 的相关启动程序参数并传递给 YSPMain.exe，click\_run.exe 本身无法解释剧情。

在打开 YSPMain.exe（下称 YSPM 平台）后，您首先看到的一定是黑白色的控制台窗口，这就是 YSPM 平台的第一级交互页面。YSPM 平台采用的是一种近似于命令提示符的操作模式。在这个页面中，您可以通过输入各种各样的指令来使用程序的不同功能。

如果您拿到的不是 Pre 版本，那么您甚至可以体验到完整的多语种翻译（目前支持简体中文与英语）。不过需要首先声明，一切文本的正确意思以简体中文为准，不保证英文翻译的准确性，更不保证英文就是标称的“英式英语”。

如若日后更新繁体中文版，那么繁体中文版将只是简体中文版的繁体化，不会按照繁体中文使用地区的语言习惯修改文字内容。

YSPM 平台是最基本的顶层交互，在这一层您可以使用 help 命令来获取帮助内容。帮助内容有对各个支持命令的最基本解释。或者您也可以看下面列出来的解释：

YSPM 平台交互命令一览		
命令文本	基本解释	适用层
about	获取程序相关信息	user input
clear	清理损坏的缓存文件	user input
exception	使程序崩溃（供测试用）	user input
exit	退出当前层（如果在 user input 层则退出程序）	任何层
help	获取关于命令的帮助	user input
lang	调整程序的语言	user input
line	进入测试单行剧情的解释模式	user input
spawn	进入整个剧情文件的解释模式	user input
ui	进入程序 UI 页面	user input

spawn 命令所指向的“整个剧情文件的解释模式”和 line 所指向的“测试单行剧情的解释模式”仅供测试解释器核心代码使用，这两个解释模式背后的解释器可能不是最新版，也可能是 Bug 满天飞的超前版——最严重的情况甚至可能是，两个标号相同的解释器版本，有着完全不同的处理逻辑与显示效果。

**总之，如果您只是播放剧情或者测试自己写的剧情，请您直接输入 ui，进入 UI 模式即可。**

还有一点需要指出的是，播放器的“多语种翻译”是限于控制台窗口下的，UI 页面内程序自带的含有文字的内容绝大部分都是贴图，无法做到这样的翻译。

<sup>1</sup> 下文所有程序说的格式均为 exe，即可执行文件。如果您拿到的包是来自 Github 的源码包，您只需要把文章中所有 exe 后缀自行脑补为 py 后缀即可，

如果您在使用 YSP 程序的时候发生了崩溃, 请留意 CrashReport 文件夹下有没有以您遇到崩溃时的时间命名的文件。该文件内有一些能够帮助我们定位崩溃的信息。如果可以的话, 欢迎把它发送给开发组。

如果您在退出 UI 页面时发现控制台窗口出现里含有 Qmutex 等类似的在文本中含有大写字母 Q 的错误提示, 则请不要发送相关的错误报告。那可能是程序正常行为, 也可能一直没有解决但不影响使用的问题。

## 0.2 SPOL 基本介绍

SPOL, 全称 Story Player Official Language (剧情播放器官方语言)。这款语言是仅用于在 YSP 程序内播放剧情所用的语言。它的最小识别范围是文件中的每一行——这就是说, 播放器中所能展示出来的内容, 都需要至少一行的内容去定义, 并且在绝大多数的情况下, 行与行之间没有任何关联。

SPOL 看起来有些像 Python, 比如大量出现的**讲述控制器**的开头是在 Python 提示符模式也同时大量出现的字符“>>>”; 还有和 Python 列表看上去差不多的**背景控制器**以及和 Python 如出一辙的用“#”字符做为注释的标识符等等。

这是有意为之的, 不仅是因为程序本身就是用 Python 开发, 因此需要视觉协调感, 更是因为 Python 是现在所有主要的、流行的编程语言里, 版面看上去最简洁明了, 学习时的接受度也是最高的语言。那么作为一个需要面向一般玩家的“仿编程语言”, 我们就必须学习 Python 的这些基本的视觉特点, 使有编程基础的、没有编程基础的各种玩家群体都能快速的接受与上手。

## 0.3 程序运行环境介绍

YSP 程序在开发的时候运行在 Windows10 的 2017 年秋季更新 (1709) 之后的版本。我们现在并没有设置系统限制, 不过开发者无法担保在非开发环境所用的系统下程序运行的可靠性。

如若您有多个显示器, 那么程序在确认显示器分辨率时, 只识别在系统中标识为 1 的那个显示器 (主显示器), ui 启动在哪个显示器上取决于在输入完“ui”之后按下回车的一瞬间光标所在的显示器。

您必须确保您标识为 1 的主显示器的分辨率不低于 1366x768, 一旦低于这个分辨率, ui 将不予启动。与此同时, 我们推荐您的**屏幕长宽比最好为 16:9 (例如 1920x1080)**, 在这种情况下, ui 显示的控件相对位置才是正常的。

## 0.4 本文档着色情况基本介绍

在本文档中, 和 SPOL 语言有关的专有名词都会用**蓝色加粗字体**标记出来, 重点内容会用**双下划线以及加黑字体**标记出来, 第一次出现的标准定义会放在**浅蓝色底纹的长方形框内, 里面的文字是蓝色**的。对于一些错误的想法, 我们会用**红色加粗字体, 并且画上划去线**。

在上文中, 我们已经用这种方式做了一些标记。

# 第一章：SPOL 入门

## 1.1 编写环境

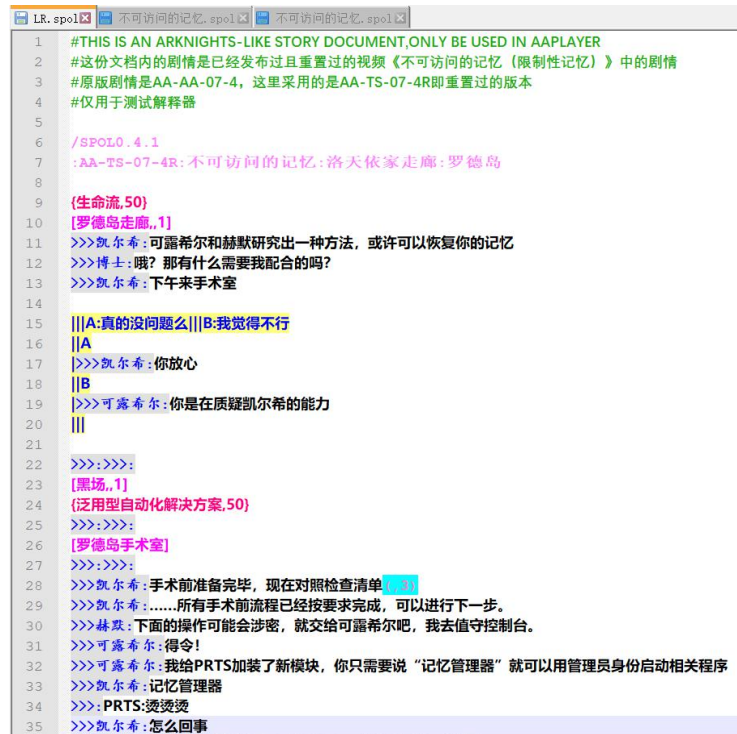
编写 SPOL 文档最简单的工具是记事本,甚至可以是手机上任何能够创建 txt 文档的文字编写程序。您只需要确保写完之后的文本文档保存格式是 spol 即可。如若您需要自己改解释器源代码,那么在测试时直接使用 Visual Studio 之类的大型集成开发环境来写 SPOL 文件也未尝不可。(作者本人就是这么干的)

不过,我们推荐使用的软件是 Notepad++, 因为这款软件支持自定义语法着色,我们已经根据 SPOL 现有的语言特性制作了用于 Notepad++ 程序的着色定义文档。

**不过请注意,无论用什么程序编写,您必须保证您的文本文件编码格式为“UTF-8”格式。**

Notepad++ 程序是免费的,您可以在网上下载之后,在菜单里面的“语言”→“自定义语言”→“自定义语言格式”的对话框里面,导入剧情源文件夹下一个名为“SPOLcolor.xml”(暂未包含在 Pre 版)的文件。这样一来,您便能获得较好的 SPOL 语言编写视觉效果。

如图所示, N2. spol 文件在 Notepad++ 里面看上去颜色分明,重点突出,总比面对一团黑色文字要好看许多。(图中的例子是 SPOL0.4.1 相关版本开发过程中的截图。文本的相关语法不作为目前最新语法的范例。)



```

1  #THIS IS AN ARKNIGHTS-LIKE STORY DOCUMENT, ONLY BE USED IN AAPLAYER
2  #这份文档内的剧情是已经发布过且重置过的视频《不可访问的记忆（限制性记忆）》中的剧情
3  #原版剧情是AA-AA-07-4, 这里采用的是AA-TS-07-4R即重置过的版本
4  #仅用于测试解释器
5
6  /SPOL0.4.1
7  :AA-TS-07-4R:不可访问的记忆:洛天依家走廊:罗德岛
8
9  {生命流,50}
10 [罗德岛走廊,1]
11 >>>凯尔希:可露希尔和赫默研究出一种方法,或许可以恢复你的记忆
12 >>>博士:哦?那有什么需要我配合的吗?
13 >>>凯尔希:下午来手术室
14
15 |||A:真的没问题么|||B:我觉得不行
16 ||A
17 >>>凯尔希:你放心
18 ||B
19 >>>可露希尔:你是在质疑凯尔希的能力
20 |||
21
22 >>>:
23 [黑场,1]
24 {泛用型自动化解决方案,50}
25 >>>:
26 [罗德岛手术室]
27 >>>:
28 >>>凯尔希:手术前准备完毕,现在对照检查清单(0.7)
29 >>>凯尔希:.....所有手术前流程已经按要求完成,可以进行下一步。
30 >>>赫默:下面的操作可能会涉密,就交给可露希尔吧,我去值守控制台。
31 >>>可露希尔:得令!
32 >>>可露希尔:我给PRTS加装了新模块,你只需要说“记忆管理器”就可以用管理员身份启动相关程序
33 >>>凯尔希:记忆管理器
34 >>>:PRTS:烫烫烫
35 >>>凯尔希:怎么回事
  
```

## 1.2 行首指示符与控制器

在 SPOL 中最重要的是每一行开始的若干字符。由于 YSP 的解释器是通过每一行开始的若干字符来确定这一行是作何用处,所以每一行开始的若干字符的区分与使用就尤为重要。

我们先来了解一下它的相关定义和基本类型。

每一行开始的若干用作告诉解释器这行是作何用处的字符,

称之为:行首指示符

SPOL0. 6+ 行首指示符	
字符（英文半角）	指示内容
/	文档标准声明行
:	标题内容定义行
#	单行文本注释行
###	多行注释起止行
[	背景内容定义行
>>>	剧情文本定义行
{	背景音乐定义行
>^>	自由文本定义行
-->	剧情分支定义行
	对话分支选项起止行
	对话分支选项标签行
	对话分支内容定义行
<	背景音效定义行

上面的几种标识符中，有些标识符是所谓“**控制器**”的一部分。

如若一系列指示符以及指示符临近的一些文本定义了某个用于剧情播放（即背景、讲述人和台词、音乐等）的内容，则把指示符和相关的內容统称为：**控制器**

SPOL0. 6+ 控制器	
字符（省略号代表中间若干内容）	控制内容
:.....:.....:.....:.....	标题控制器
[.....]	背景控制器
{.....}	音乐控制器
>>>.....:.....	基本型讲述控制器
>>>.....:.....:.....	分离型讲述控制器
(.....)	行进驻留控制器
>^>.....:.....	自由文本控制器
-->.....:.....:.....	剧情分支控制器
.....	对话分支选项控制器
.....	对话分支标签控制器
.....	对话分支内容控制器
<.....>	音效控制器

需要指出的是，**行首指示符**和**控制器**这两个概念完全不一样。**行首指示符**只是解释器对于 SPOL 中某一行的功能的基本判断，而真正定义功能的是**控制器**。有些控制器不一定位于一行的开头，比如行进驻留控制器。有些时候含有行首指示符的控制器也不一定从一行的第一个字符开始，比如在前面存在对话分支的情况下。



这些特例我们后面都会讲到，总之，现在请一定不要在脑海中形成这样的成见：~~既然行首指示符是控制器的一部分，那么控制器一定位于一行的开头。~~

## 1.2 章节习题

1. 记忆现有的行首指示符和控制器。
2. 浏览 YSP 文件夹中的剧情文件，尝试在没有进行系统的学习之前先自行理解它的意思。
3. 尝试在 YSPM 平台上用命令切换语言
4. 尝试在 YSPM 平台上用命令播放一个剧情文件
5. 尝试使用 YSPM 平台启动 UI 并播放一个剧情文件

## 第二章：文档标准、注释和标题

### 2.1 文档标准

众所周知，每一种语言都有自己的标准，SPOL 也不例外。

由于 SPOL 迭代速度时快时慢，增删与修改时有发生，因此必须首先自报家门告诉程序，自己使用的是哪一个版本的 SPOL，以便程序能够按相同标准正确的理解你所编写的剧情文档。

例 2.1

```
/SPOL0.6.0
/SPOL0.2
/AASP0.1
```

上面的三行分别意味着，使用 SPOL0.6.0 标准，SPOL0.2 标准，使用 AASP0.1<sup>2</sup> 标准。

当然，请注意这只是个使用的例子，一个剧情文件只能有效的声明一个标准。您可以把声明随便写在文档中的任何地方。第一行，第十行，最后一行，怎么都行，不过为了方便阅读、节省时间，我们推荐把他写在第一行。

现在就需要指出的一点是，到后面我们会学习剧情的大分支写法（即把不同的剧情走向写在不同的剧情文件里面），在含有大分支的一组剧情里面，只有第一个剧情文件所声明的文档标准会被识别，后面的分支剧情习惯上都不声明文档标准，这样也可以避免直接打开分支剧情文件。因为没有声明文档标准的文档无法单独打开。

### 2.2 文档单行注释

许多的程序也都有自己的注释系统，比如 C 语言是//注释，Python 是#注释。注释是用来给用户看的，方便用户理解程序的内容，解释器并不会理会注释中的东西。在 SPOL 中也是这样，如果一行以“#”开头，那么这一行的内容就作为文档的注解，不会被解释器进一步识别，在“#”的后面可以写上任何东西而不干扰解释器正常运行。

例 2.2

```
#/SPOL0.2
/SPOL0.6.0
#这是一行注释
这是一行本想用作注释的文本
```

上面的例子中，第一行#/SPOL0.2 就是一行注释，即使这一行出现了形如“/SPOL0.2”的内容，这也不意味着把 SPOL0.2 作为文档的标准。

第二行就是正式的指示文档标准的标准声明行。

而第三行是中规中矩的注释。

<sup>2</sup> AASP0.1 标准：在 SPOL 被称之为 SPOL 之前，用的是播放器的原名称 AASP 作为标准的名称

至于第四行，可能作者原本想要写一些注释，但是忘记加“#”号，这就导致这一行没法被正确识别，它会被解释器跳过，如若在 spawn 即解释整个文件的模式或在 UI 模式下，则会于剧情播放完毕之后在控制台页面上被列入警告。

如果您不想键入字符#，单行注释实际上可以用空格起头，但是我们不推荐这么做。因为这样会让格式看起来非常混乱，尤其是在支持语法着色的程序当中。

这里要顺带提一句，程序的 line 命令仅供测试讲述控制器、自由文本控制器和行进驻留控制器。不能测试其他内容。

## 2.3 文档多行注释

在某些特殊的情况下，我们需要进行长篇大段的注释。在这种情况下，为每一行文本的开头都添加字符“#”非常繁琐，且不说影响美观，如若用户输入自己的注释过于专注，则非常有可能漏掉字符“#”。

因此，这个时候就有必要使用多行注释。多行注释以字符###作为开头和结尾。夹在一对###之间的内容即为多行注释。

例 2.3

```
#这是单行注释
###这一行也可以写注释
这是多行注释的第一行
这是多行注释的第二行
/SPOL0.2
###
/SPOL0.6.0
```

上面的例子中，我们需要注意到，字符###作为多行注释的起始时可以在后面跟着写文本，但在作为终止时需要独占一行。

写在两个###之间的多行注释内容会被解释器跳过，因此多行注释内的“/SPOL0.2”虽然看起来的确是一个标准声明行，但是由于其被写在了多行注释的区域之内，因此并不会起作用。这个片段的标准声明即为最后一行的“/SPOL0.6.0”

## 2.4 标题控制器

剧情是要拥有自己的标题与副标题的。在《明日方舟》中，写在上侧位置的标题是编号标题，比如“1-7”“END8-1”等等，而下面较小的文字则是副标题。

标题控制器即是负责定义编号标题和背景图片、派别 logo 的控制器，它的标准定义如下：

```
:Title:Sub_Title:Background_Name:Logo
```

标题控制器一共有三个字段，每个字段用英文（半角）逗号隔开。

第一个字段代表的是剧情的编号标题（大标题）

第二个字段代表的是剧情的文字标题（副标题）

第二个字段代表的是展示标题的时候展示的背景，

第三个字段代表的是展示标题的时候展示的派别的 logo。

标题和副标题的声明也没有强制性的位置限制，不过推荐把它们放在文件的

开头位置。

例 2.4

```
/SPOL0.6.0
:TE-EX-1:教学范例 1:罗德岛走廊:罗德岛
```

上面的例子中，编号标题即为“TE-EX-1”，文字标题即为“教学范例 1”，文档采用的标准是“SPOL0.6.0”。展示标题的时候展示的背景是罗德岛走廊，派别 logo 是罗德岛。

SPOL 语言的一个特点就是“所见即所得”，正如您在上面看见的背景“罗德岛走廊”和派别 logo“罗德岛”一样，您可以分别在源文件目录下的 Visual\source\BGP 和 Visual\source\Logo 下面找到同名文件“罗德岛走廊.png”和“罗德岛.png”。

这就是说，在 SPOL 中所写下的资源名称，必须在对应位置拥有同名的、指定格式的文件才能够使用。如果资源引用失败，则会以默认值填充。关于资源的具体添加要求会在后面的章节会继续详细讲解。

一旦解释器识别到了标题控制器，便会开始计时。这个特性需要记住，后期如若推出直接输出到 Adobe PR 等软件的功能的话，会以这个为参考时轴。（笔者亦即开发者本人实际上对实现这个功能不抱任何期望。。。)

我们在下文介绍背景控制器的时候会遇到被称作“[按位略去](#)”和“[后方略去](#)”的两种控制器简写方法，需要指出的是，标题控制器并不支持这两种方法。

YSP 支持对有错误的行采取暂时忽略、解释完毕后再报错的特性，但是如果标题控制器出现了定义错误的情况，解释器不会进一步读取文件，而是直接退出解释。

## 2.5 章节习题

1. 判断下列说法的正确与否。

- (1) 文档标准声明可以写在文件的最后一行
- (2) 多行注释起始行的字符###后面可以有内容
- (3) 字符###作为多行注释的终止时可以写在注释内容的末尾

2. (供有编程基础的用户思考) 观察源代码，判断以下说法正误

- (1) 在声明完文档标准之后，字符/也可以当做注释行首指示符使用
- (2) 在声明完两个标题之后，字符:也可以当做注释行首指示符使用

## 第三章：背景、音乐、音效控制器

### 3.1 背景控制器

在剧情当中，背景是必不可少的重要组成元件。背景控制器即是负责控制定义背景相关属性的内容。

背景控制器的标准定义如下：

**[Background\_name, filter, effect, fade\_in\_time]**

背景控制器一共有四个字段，每个字段用英文（半角）逗号隔开。

第一个字段代表的是背景的名称，解释器会根据键入的名称显示源文件下为这一名称的背景。这一项的默认值是“黑场”，也就是黑屏。

第二个字段是滤镜，用数字 0、~5 分别表示正常显示、变暗滤镜、褪色滤镜、褪色变暗滤镜、黑白滤镜、黑白变暗滤镜——这项功能是为了方便用户能够使用效果统一的背景，您只需要一张正常的背景图片，然后在必要时加上相关滤镜即可完成褪色、黑白等操作。这一项的默认值是 0。

第三个字段是特效，用数字 0、1、2、3 分别表示正常显示、画面抖动（一次）、画面白闪（一次，快）、画面白闪（一次，慢）。这一项的默认值是 0。

第四个字段是淡入时间。这一项的默认值是 0.5。

背景控制器支持一种叫做“**按位略去**”的操作，即，可以不填充不想更改默认值的字段，但要必须表示出这个字段。也支持一种叫做“**后方略去**”的操作，即，在修改了欲修改的一个字段之后，如若其后字段不需要更改，就可以不表示出后方的任何字段。

例 3.1

```
[罗德岛走廊, 2, 2, 0.3]
[罗德岛走廊]
[罗德岛走廊, 2, , 1]
[汐斯塔沙滩, , 1]
[, , 2]
[, , , 0.2]
```

上面的每一行都是能被解释器正确识别的背景控制器，现在我们来逐个分析。

第一行“[罗德岛走廊, 2, 2, 0.3]”代表的是，设置背景为“罗德岛走廊”，图片为褪色，并且需要在 0.3 秒的淡入之后有一次快速白闪。很容易想象出，这个背景设置可以用于“回忆很久以前发生在罗德岛内部的械斗事件”，想必用来描写特蕾西娅的相关事件是再合适不过了。

而第二行“[罗德岛走廊]”则使用了“**后方略去**”这一特性。即，用户只想直接设置背景为“罗德岛走廊”，不需要任何滤镜或特效，淡入时间也不需要特殊设置，那么后面的既然都不改，就可以直接全部略去。

第三行“[罗德岛走廊, 2, , 1]”则使用了“**按位略去**”这一特性。即，用户不需要制定特效，只需指定滤镜与淡入时间，那么与其在第三个字段上多耗时间敲一个数字 0，不如直接把这一位空出来不去填写，系统会自动按照默认值 0 进行填充。

第四行“[汐斯塔沙滩, 1]”则混合使用了上述两个特性。即，用户指定了背景为“汐斯塔沙滩”，并且需要一次画面抖动，并且用户不需要修改滤镜和淡入时间，于是前面的滤镜位置就需要“**按位略去**”，即不填写这一位具体内容，但要表示出这个空位，而后面的淡入位置则进行“**后方略去**”，即直接省略不写。

讲完了上述四行，我们应该很容易理解，第五行的意思是“背景为黑场，正常显示，画面慢速白闪一次，淡入时间为默认值 0.5”

第六行则是“背景为黑场，正常显示，没有特效，淡入时间为 0.2”

一开始背景控制器是拥有淡出参数的，不过后来被取消了，现在，一个场景的淡出由下一个场景的淡入决定，这样也更符合逻辑。

如若需要使用多次白闪，可以将控制器复制多次，并把后续的淡入时间改成 0 即可。

### 3.2 音乐控制器与音效控制器

作者一直认为，背景控制器与音乐控制器对于没有编程基础的用户来说，最难的知识点就是两种略去的使用，但作者相信通过上面的简单小例子，所有读者应该都能理解这两种略去的使用方法，那么，由此一来，音乐控制器教程的长度就可以大幅缩短了，因为音乐控制器也支持上面的两种略去方式，所以只需给出标准定义和基本解释即可。

音乐控制器的标准定义如下：

**[Music\_name, volume]**

音乐控制器一共只有两个字段，作者还在慎重考虑是否需要其他字段来增加控制对象。就目前来说，第一个字段代表音频名称，第二个字段代表音量，默认值为 50，范围是 1 到 100。后期可能会考虑增加两个字段，即音频的起止时间，以方便用户快速使用音频的某个片段。

在剧情播放时，如果正在播放某个音频，并且遇到了下一个音乐控制器，那么当前播放的音频会停止，程序会立即播放下一个音乐控制器指定的音频。

**请务必注意，为了用户更灵活更自主的调度音频以及管理剧情运行时长，背景音乐是不支持设置循环播放的。**

音乐控制器是专门用来播放背景音乐的，如若用户需要播放其他音效，比如枪声等等，那么就需要使用音效控制器。音效控制器与音乐控制器的最大差别在于，**音效控制器播放的音频不会被后一个相同的控制器替换掉，音效控制器强制播放整段音频。**

音效控制器的标准定义如下：

**<Wave\_name, volume>**

音效控制器的字段定义和音乐控制器是一样的。

### 3.3 章节习题

1. 完整表示背景设定要求“背景为罗德岛走廊，正常显示，没有特效，正常淡

- 入”。
2. 缩写表示背景设定要求“背景为罗德岛走廊，正常显示，画面抖动，正常淡入”。
  3. 缩写表示音乐设定要求“音乐为生命流”
  4. 完整表示音乐设定要求“音乐为自动化解决方案，音量为 35”
  5. 现在需要体现陈和塔露拉斗争的场面，试给出一个贴切的控制器方案（不一定只有一行），假设背景名称为“控制塔”。



## 第四章：讲述控制器

### 4.1 基本型单人讲述控制器

#### 4.1.1 定义

在学习完前三章的内容之后，我们进入了重点，即承载剧情的主体：人物对话和旁白。动人心弦的文字才是打动读者的第一要素。

我们先来考虑最简单的情形：旁白，或者场上只一个人在独白。

旁白和独白统称为单人讲述，基本型单人讲述控制器标准定义如下：

**>>>Name/countenance/turn/filter/fade\_in\_time/fade\_out\_time:Words**

单人讲述控制器一共有七个字段，隔开它们的字符均为英文（半角）字符。

第一个字段决定显示名称和立绘名称。即这个字段既作为讲述人姓名直接显示在屏幕上，也作为在源文件中查找对应立绘并显示的依据。默认值为空

第二个字段决定表情差分，用源文件中对应的数字来调用对应立绘，比如名为“凯尔希\_1”的立绘就需要前两个字段为“凯尔希/1”。这就意味着，文件名时必须采取名称加下划线加数字的形式，且在没有数字的时候下划线也不能省略。第二字段默认值是空

第三个字段决定是否镜像，即决定这个立绘是否需要水平镜像处理，默认值为 0，即不需要镜像。这一字段的默认值是 0

第四个字段是立绘的特效，用数字 0~5 表示无特效、立绘上隐、立绘褪色、立绘褪色上隐、立绘黑白、立绘黑白上隐。这一字段的默认值是 0

第五个字段和第六个字段是淡入淡出时间，默认值均为 0

第七个字段即人物独白或旁白的内容。

**\*请注意，立绘的淡入淡出时间目前没有强烈需求，所以暂未实装，不能在 UI 里体现出来。**

基本型单人讲述控制器也同样支持**按位略去**和**后方略去**的特性。不过它的最简形式必须至少为“>>>:”，即符号“>>>”和符号“:”必不可少。实际上，“>>>:”这个形式本身也是一种固定用法，下面会一一介绍。

#### 4.1.2 用作旁白

例 4.1.2-1

```
/SPOL0.6.0
:TE-EX-2
:教学示范 2
[黑场]
>>>:西部的苍凉之地
>>>:罗德岛某舱室，时间未知
```

在上面的例子中，第五行和第六行即为旁白的内容。实际上旁白的原理就是一个没有姓名的讲述人而没有姓名键入也正好导致了不调用立绘。



### 4.1.3 用作独白

例 4.1.3-1

```
/SPOL0.6.0
:TE-EX-2
:教学示范 2
[黑场]
>>>凯尔希:而我也曾行于那茫茫的大地
>>>凯尔希:在漫天黄沙之下
```

在上面的例子中，第五行和第六行即为独白的内容。这是基本型单人讲述控制器最基本的用法之一。它只包含有姓名和所说的话，其他字段的设定全部略去。

如果需要调整立绘的显示情况，比如说调用各种表情差分，镜像翻转等等，只需加上对应的参数，下面的例子就是调用一号表情，立绘水平翻转的例子：

例 3.1.3-2

```
/SPOL0.6.0
:TE-EX-2
:教学示范 2
[黑场]
>>>凯尔希/1/1:从来没有一束令人不愿直视的阳光能畅通无阻的照向大地
>>>凯尔希/1/1:那里永远是黑暗最爱的栖身之所
```

## 4.2 基本型多人讲述控制器

### 4.2.1 定义与基本对话

上一小节我们学习了基本型单人讲述控制器，他的标准定义如下：

**>>>Name/countenance/turn/filter/fade\_in\_time/fade\_out\_time:Words**

基本型多人讲述控制器就是在基本型单人讲述控制器的基础上发展而来。基本型多人讲述控制器实际上就是把两个基本型单人讲述控制器写在同一行。基本型多人讲述控制器用于对话情境，或者是在一个画面上调用两个立绘的情境。不过需要注意的是，如果是两个人物的对话，那么最多只能有一个基本型单人讲述控制器允许拥有台词。两个人不能同时拥有台词。

例 4.2.1

```
/SPOL0.6.0
:TE-EX-2
:教学示范 2
[黑场]
>>>凯尔希/1/1:也许我们不该插手这件事>>>博士:
>>>博士:但我心意已决。>>>凯尔希/1/1:
```

在上面的例子中，第五行与第六行对应的两个画面中，会同时出现凯尔希和博士的立绘，并且根据顺序显示——第五行对应的画面凯尔希在左，博士在右；第六行对应的画面博士在左，凯尔希在右。请务必注意，即使一个人没有台词，

符号“:”也不能略去。在讲述控制器中，符号>>>和符号:是重要的解释器识别对象，一旦缺少就会造成无法识别。

### 4.2.2 用作对齐手段

在解释器中，由于立绘是根据键入姓名自动调度，而没有姓名会导致立绘不显示，这样的特性决定着基本型多人讲述控制器有一个其他功能，即用作单人讲述时立绘的对齐手段。

例 4.2.2-1

```
/SPOL0.6.0
:TE-EX-2
:教学示范 2
[黑场]
>>>博士:我也曾窥探自己的内心>>>:
>>>博士:它告诉我我应当这么做>>>:
```

在上面的例子中，第五行与第六行对应的两个画面中，都会只出现博士一个人的立绘，但是由于这本质上仍然为多人讲述控制器，因此这个立绘都是位于左侧的，它会空出右半个空间。如果要一个人的立绘位于右侧，空出左半个空间，那么就可以按下面的例子键入：

例 4.2.2-2

```
/SPOL0.6.0
:TE-EX-2
:教学示范 2
[黑场]
>>>:>>>凯尔希:那我也便不再拦你
>>>:>>>凯尔希:去走上你自己的道路
```

### 4.3 分离型讲述控制器

我们考虑这样一种情况。某位角色刚刚上场，但这个上场的过程需要保持一定的神秘性，于是我们考虑给立绘加一个上隐的效果：

例 4.3-1

```
/SPOL0.6.0
:TE-EX-2
:教学示范 2
[黑场]
>>>:(开门声)
>>>煌///1:你好啊博士！
>>>煌初次见面，请多多关照！
```

但是我们很容易发现，即使立绘被上隐效果盖得严严实实，下面显示姓名的地方还是会暴露出干员的真实姓名。这是由于 Name 字段同时与显示名称、调用立绘名称挂钩的特性导致的。它的本意原来是想方便用户快速编写，但是在这样

的时候反而显得非常帮倒忙。

于是我们希望有一种手段，能在正常调用立绘的同时显示其他名称，这便是分离型讲述控制器，它的标准定义如下：

```
>>>Name/countenance/turn/effect/fade_in_time/fade_out_time:show_name:Words
```

和基本型的讲述控制器相比，我们注意到，在姓名和相关设置之后，人物台词之前的位置，新增了一个字段，这个字段即为：实际显示名称字段。

例 4. 3-2

```
/SPOL0. 6. 0
:TE-EX-2
:教学示范 2
[黑场]
>>>:(开门声)
>>>煌///1:???:你好啊博士！
>>>煌:初次见面，请多多关照！
```

我们把例 4. 3-1 的第六行改成如上的第六行，这个时候，解释器就会正常调用煌的立绘，加上上隐效果，但是显示的名称为“???”

于是我们可以很快的想到，如果我们结合考虑上文介绍的旁白用法和分离型讲述控制器本身的特性，那么在其他的一些特殊情况下，就能够做到正常显示人物姓名，但是不显示人物立绘。例如下面这个例子：

例 4. 3-3

```
/SPOL0. 6. 0
:TE-EX-2
:教学示范 2
[黑场]
>>>:(路过的干员):(小声)那就是传说中的精英干员煌么？
>>>路过的梓兰:没错，是她。
```

路过的干员可能没有自己立绘，但是我们需要在姓名的位置显示“路过的某干员”，路过的梓兰可能也并没有出现在视野当中，故只需要显示姓名，但不需要显示她的立绘，那么这个时候就可以考虑采用分离型讲述控制器来完成这个操作。

最后一点是，分离型讲述控制器与基本型讲述控制器在组合上等价，**除了下文介绍的空场控制器的两种用法外**，分离型讲述控制器可以任意的代替基本型集讲述控制器——例如可以采用分离型单人讲述控制器组成分离型多人讲述控制器，从而完成和一般的基本型多人讲述控制器一样的操作。

## 4. 4 空场控制器、行进驻留控制器

在节 4. 1. 1 中我们提到，“>>>:”也是一种固定用法，现在我们就来介绍这种用法。

当基本型单人讲述控制器形如“>>>:”的时候，称之为有遮罩空场控制器。在有的时候，场上并不需要有人在对话，这个时候我们就考虑使用一个有遮

罩空场控制器。之所以称之为“有遮罩”，是因为场上虽然没有人发言，但是保留下半部分和上半部分的黑色渐变遮罩效果。如若需要完整的展示背景，不需要任何遮罩效果，那么就需要无遮罩空场控制器，即“>>>:>>>:”。无遮罩空场控制器实际上就是基本型多人讲述控制器的最简形式。

但是即使是这样，读者可能还是想问，我怎样才能控制空场的时间呢？我才学了那么多讲述控制器，我怎么才能控制文本行进速度和每一次台词播放完成后的停留时间呢？

于是我们要引出行进驻留控制器，它的标准定义如下：

(speed, stop)

行进驻留控制器只有两个字段。

第一个字段决定着每个字弹出之后的停留秒数（出字延迟），它的默认值是 0.15（请注意，以前是 0.1）

第二个字段决定着每一页播放完毕之后的停留秒数，它的默认是 1.5

行进驻留控制器允许按位略去和后方略去的操作

行进驻留控制器能且只能写在拥有讲述控制器或者下文介绍的拥有自由文本控制器的行的末尾，不允许写在其他位置（准确来说，在其他位置上即使写了行进驻留控制器，也会被当做普通文本）。

例 4.4

```
/SPOL0.6.0
:TE-EX-2
:教学示范 2
[乌萨斯冰原]
>>>:>>>:(,3)
>>>:???:♪来自远方的人啊~♪(0.8,2)
```

上面的例子中，第五行表示的是一个长达三秒的无遮罩空场效果，第六行为体现了某种未知来源的吟唱效果，则把出字延时从 0.2 的默认值改到了 0.8，并且把出字结束后的等待时间从默认值 1.5 改到了 2，充分体现出了乌萨斯冰原浩瀚缥缈与歌声神秘悠扬的效果。

## 4.4 章节习题

1. 用现有的知识，复现视频 [BV1qi4y1P7eT](https://www.bilibili.com/video/BV1qi4y1P7eT) 所展示的剧情。

## 第五章：自由文本控制器

在明日方舟的剧情当中，我们会注意到有些时候会有部分文字出现在屏幕的中间或者其他某个并不属于下方对话区域的位置。对于这种情况，我们的讲述控制器是无能为力的，于是我们需要一种新的控制器，能够控制文本自由的出现在屏幕上的某一个位置。

自由文本控制器标准定义如下：

>>>X/Y/align:Words

自由文本控制器一共有四个字段，隔开它们的字符均为英文（半角）字符。

前两个字段决定着文本的位置，这里的 X, Y 坐标分别是屏幕自左上角起的长度坐标和高度坐标的百分比。也就是说，居于屏幕左上角就是坐标 0/0，右上角就是 1/0，左下角就是 0/1，右下角就是 1/1。居中就是 0.5/0.5。居中就是默认值情况。

第三个字段决定了文本的对齐方式。如若填写值 L，意味着文本左对齐，如若填写值 M，意味着文本居中对齐，如若填写值 R，意味着文本右对齐。默认值为 M

第四个字段则是文本内容。

自由文本控制器允许[按位略去](#)和[后方略去](#)的操作。自由文本控制器的最简形式为 “>^>:”

例 5.1

```
/SPOL0.6.0
:TE-EX-3
:教学示范 3
[罗德岛舰桥]
>^>:她时常一个人静静的回想以前发生的事情
>^>:“那些都是真的么” (0.5, 2)
[黑场]
```

在上面的例子中，两行文字都没有设置任何坐标和对齐操作，这是自由文本控制器的最简形式，也是游戏中出现对话外文本时最常见的居中对齐的形式。

自由文本控制器同样支持使用行进驻留控制器来控制出字速度。不过与讲述控制器下的行进驻留控制器有所不同的是，自由文本控制器下的行进驻留控制器的出字延迟（第一个字段）的默认值是 0。

在讲述控制器下的行进驻留控制器的第一个字段即使设置值为 0，它也会因为程序运行逻辑而导致有个最小延迟，这个延迟小到好像整段文字是一起出现的，但事实上是一个字一个字蹦出来的。不过在自由文本控制器下的行进驻留控制器的第一个字段设置为 0 的时候，程序会强制整段文字一起出现。

**\*上述强制整段文字一起出现的功能暂未实现（也许没有实现？我已经也记不清了，最近没有测试这一项目。）**

# 第六章：选择分支控制器

## 6.1 剧情分支控制器（大分支控制器）

在明日方舟的剧情中，会有不少时刻需要用户选择一些对话选项。于是 B 站的网友在创作剧情的过程中，充分利用了 B 站互动视频的特性，为了还原游戏的交互体验，制作了一些在用户选择不同选项后会有截然不同的剧情走向的同人剧情。本节即介绍能够实现这个过程的剧情分支控制器。

大分支控制器的每个独立部件标准定义如下：

-->Label1:Text1:filename1

每个独立部件共有三个字段

第一个字段是选项的标签，这个可以自由设置，没有默认值。在 Spawn 模式下，用户需要输入标签内容才能选择对应选项。如若多个标签名称重复，则默认选择所有重复项中的第一个。

第二个字段是显示文本，即选项的内容，没有默认值。

第三个字段是该选项对应需要打开的剧情文件名称，**注意不要加 .spol 后缀**。

例 6.1.1

```
/SPOL0.6.0
:TE-EX-3
:教学示范 3
[罗德岛舰桥]
>>>:她时常一个人静静的回想以前发生的事情
>>>:“那些都是真的么”(0.5,2)
-->A:我已经记不清了:TE_EX_3_A-->B:我真切的经历过那段时光:TE_EX_3_B
```

在上面的例子中，最后一行定义了两个选项。在 Spawn 模式下，控制器会逐行列出选项标签和内容，并且在用户输入了一个标签之后打开对应的文档。比如说用户输入 B，那么解释器就会在 story 文件夹下查找名为 TE\_EX\_3\_B.spol 的文档并尝试打开。

需要指出的是，**一旦解释器遇到大分支控制器，就会认为遇到了这个文档的最后一行**，也就是说，在完成大分支控制器的解释和文件打开尝试后，解释器会立即退出这一文档。在大分支控制器所在行之后的任何内容都不会被继续读取。

在一些特殊情况下，可能需要若干剧情文件通过大分支控制器的种种调度来完成一个完整的大型剧情，那么在这种情况下把所有的相关剧情文件直接塞到 story 文件夹是一件非常辣眼睛的事情，于是我们希望能够使用一个文件夹包裹一组剧情所有的相关文件。

例 6.1.2

```
/SPOL0.6.0
:TE-EX-3
:教学示范 3
[罗德岛舰桥]
>>>:她时常一个人静静的回想以前发生的事情
```



```
>>>: “那些都是真的么” (0.5, 2)
-->A: 我已经记不清了:TE_EX_3\A-->B: 我真切的经历过那段时光:TE_EX_3\B
```

在上面的例子中，A 选项会查找 Story 文件夹下的 TE\_EX\_3 文件夹中的文件 A. spol，B 选项会查找 Story 文件夹下的 TE\_EX\_3 文件夹中的文件 B. spol。这就是说，可以在剧情文件的名字之前加上从 story 文件夹开始（不含 story 本身）的绝对文件路径，来告诉程序剧情文件的位置。这样一来，就可以通过建立文件夹来包裹一组剧情文件，易于整理，且能被解释器正确调用。

## 6.2 对话分支控制器（小分支控制器）

我们在上节的引入中提到了明日方舟剧情本身的对话选择，每一个看过剧情的玩家都知道，游戏内的这些对话选择实际上并不会像 B 站网友制作的互动视频那样强烈的影响剧情走向，只是会简单改变在选择之后紧跟的几句话的内容。对于这样的分支情况，使用剧情分支控制器来在文件之间来回跳转读取就极其不划算，所以我们这一节介绍能够实现游戏内对话选择的对话分支控制器。

之所以先讲解类似 B 站互动视频的大分支控制器，把小分支控制器放在后面，是因为小分支控制器有些难度——

它拥有 SPOL 最独一无二的特性：它需要若干行互相关联的控制器构成。

对话分支控制器的对话分支**选项**控制器的每个独立部件标准定义如下：

```
|||Label:Text
```

每个独立部件共有两个字段，他们的定义和大分支控制器每个独立字段的前两个字段的定义一样，不再赘述。

对话分支控制器的对话分支**标签**控制器的每个独立部件标准定义如下：

```
||Label
```

对话分支**标签**控制器的每个部件只有一个字段，那就是已经出现在对话分支选项控制器里面的标签字段。

对话分支控制器的对话分支**内容**控制器的每个独立部件标准定义如下：

```
|Other
```

对话分支**内容**控制器实际上可以看做是一种行首标识符。

对话分支控制器的全结构（以三个选项为例）标准定义如下：

```
|||Label_1:Text1|||Label_2:Text2|||Label_3:Text3
||Label_1
|Other1
||Label_2
|Other2
||Label_3
|Other3
|||
```

这就是说，对话分支控制器的第一行决定了选项的标签和显示的内容，之后解释器会根据用户的选择，在下面查找与标签对应的模块。比如用户选择了第二个选项，那么解释器就会从 Label2 开始，读取完整的 Other2 的内容。

例 6.2.1

```
/SPOL0.6.0
:TE-EX-3
:教学示范 3
[罗德岛舰桥]
>>>:她时常一个人静静的回想以前发生的事情
>>>:“那些都是真的么”(0.5,2)
|||A:我已经记不清了|||B:我真切的经历过那段时光
||A
|>>>:岁月已经抹去这段印记
|>>>:让任何人不再想起它的惨剧
||B
|>>>:没有人能让我忘却
|>>>:我所亲眼见到的景象。
|||
```

在上面的例子中，如若用户选择了“我已经记不清了”，则解释器会去解释标签 A 到标签 B 之间的内容。显示旁白“岁月已经抹去这段印记”和旁白“让任何人不再想起它的惨剧”；如若用户选择了“我真切的经历过那段时光”，则解释器会去解释标签 B 到对话分支控制器结束这之间的内容。

在一个对话分支标签控制器下可以有若干行对话分支内容控制器。您可以像正常情况一样随心所欲写剧情，只是别忘了每行的开头都是字符“|”。

特别的，对话分支控制器不支持中间存在空行，因为对话分支控制器的若干行是一个完整的整体。如果遇到了空行，解释器会认为对话分支控制器的部分已经结束。如果这个时候还有剩余的对话分支控制器未解释完毕，则会被列入报错内容。



# 第七章：剧情资源使用

由于明日方舟的完整资源包特别大，因此在程序包中直接为每一位用户提供全部资源是不可能的。我们只在程序包中放置了几个范例剧情所需要的资源文件。因此，如果您想播放来自他人的剧情，或者自己创作剧情，您就了解如何向程序源文件夹添加文件。

各类文件的文件接受格式和放置位置如下：

文件描述	放置位置	接受格式
背景文件	Visual\source\BGP	任意像素，PNG 格式
立绘文件	Visual\source\Chara	900x900+像素，PNG 格式
音频文件	Visual\source\BGM	MP3 格式
派别 Logo 文件	Visual\source\Logo	510x510 像素，PNG 格式
按钮贴图文件	Visual\source\BaseUI\Button	原图匹配像素，PNG 格式

由于在读取的时候，程序会对背景文件进行缩放，所以背景文件可以是任意像素，不过我们推荐的图片最低质量是长宽比为 16:9，且不小于 1024x768 像素。

立绘文件可以是任何正方形像素的文件，推荐分辨率 900x900（或更高）。如若您自己准备了其他立绘，请自行处理到 900x900 大小，并且做好背景扣除，程序无内置抠图算法。

派别 Logo 文件的要求和立绘文件一样，都可以在 PRTS 上获取标准文件。至于自行制作的都需要自己进行缩放与抠图处理。

按钮贴图属于程序 UI 的一部分，如无必要不要更改。如若必须更改，需要保证新的 png 文件与原始文件的像素一致。

音频文件目前只支持 mp3 格式，且必须为无损压缩的高质量 mp3 音频，任何有损压缩都不能被正常播放。

## 其他致谢名单（排名不分先后）

称谓（B 站用户名）	致谢内容
Ayano_Aishi	提出若干优化用户使用体验的建议 反馈了若干 Bug 提出了可以用于程序加密后分发的秘钥生成与验证算法
Kwlgjp	提出若干优化用户使用体验的建议 对开发者在开发过程中提出的问题给出了参考性建议 尝试给 YSP 制作总启动器程序和 SPOL 的 IDE 环境
Dw 戒	帮忙解决了光谱行动期间开发者没时间没水平打合约 18 的问题
白羽墨雪	帮忙解决了原神 1.5 版本期间开发者没时间协调部家具制作时长的问题
_低吟ゞ	对开发者在开发过程中提出的问题给出了参考性建议 反馈了若干 Bug 对程序内容进行了英文翻译
这只寰宇不想咕咕	在 YSP 播放器内测期间使用 YSP 制作了数个播放量可观的视频 反馈了若干 Bug