

pymo (python memories off)

使用说明

V1.2.0

– By chen_xin_ming

目录

第一章	pymo 简介	5
第二章	AVG 的组成部分和移植方法简介	6
第三章	pymo 的目录结构	7
第四章	游戏配置文件解析 (gameconfig.txt)	8
第五章	pymo 中的坐标、颜色和时间表示方法	9
一、	坐标	9
二、	颜色	9
三、	时间	9
第六章	pymo 支持的图像、音频、视频格式及转换方法	10
一、	图像	10
二、	塞班版本图像的透明	10
三、	安卓、PC 版本图像的透明	13
四、	音频	13
五、	视频	14
六、	pymo 推荐的图像、音频格式及参数	15
七、	系统图像的参数和要求	15
第七章	制作你的第一个游戏!	17
一、	素材	17
二、	脚本	17
三、	运行	18
第八章	一些脚本范例	19
一、	游戏开始界面	19
二、	选择支示例	20
三、	每次游戏开始界面显示不同的图片 (使用全局变量)	21
第九章	鉴赏系统的编写	23
一、	CG 鉴赏	23
二、	自动生成 CG 鉴赏缩略图	24
三、	音乐鉴赏	24
第十章	游戏资源的打包和发布	26
一、	资源打包	26
二、	游戏发布	26
第十一章	pymo android 版、PC 版注意事项	27
第十二章	脚本指令说明书	28
一、	对话文字显示指令	28
1.	#say	28
2.	#text	28
3.	#text_off	29
4.	#waitkey	29
5.	#title	29
6.	#title_dsp	30
二、	图像相关操作指令	30
1.	#chara	30
2.	#chara_cls	31

3. #chara_pos	31
4. #bg.....	31
5. #flash.....	32
6. #quake	32
7. #fade_out.....	32
8. #fade_in.....	33
9. #movie.....	33
10. #textbox.....	33
11. #chara_quake.....	33
12. #chara_down	34
13. #chara_up.....	34
14. #scroll.....	34
15. #chara_y.....	35
16. #chara_scroll	36
17. #anime_on.....	36
18. #anime_off	37
19. #chara_anime	38
三、 变量、选择、跳转类指令	38
1. #set	38
2. #add.....	38
3. #sub.....	39
4. #label.....	39
5. #goto	39
6. #if...goto.....	40
7. #change	40
8. #call.....	40
9. #ret	41
10. #sel	41
11. #select_text.....	41
12. #select_var	42
13. #select_img	42
14. #select_imgs.....	43
15. #wait.....	44
16. #wait_se	44
16. #rand	44
四、 声音类指令.....	45
1. #bgm	45
2. #bgm_stop.....	45
3. #se	45
4. #se_stop.....	46
5. #vo.....	46
五、 系统类指令.....	46
1. #load.....	46
2. #album.....	47

3. #music	47
4. #date	47
5. #config	48

第一章 pymo 简介

pymo 全称 Python Memories Off，是由 chen_xin_ming 开发的一款 AVG 游戏引擎。因其基于 python 平台，且适合于创建“秋之回忆”风格的 AVG 而得名。

pymo 专为手机平台开发，注重运行效率和针对手机屏幕的优化。目前可以运行于 S60v3、S60v5，安卓和 Windows 平台，即使在低端的 S60v3 也可以稳定流畅地运行，是为手机开发 AVG 游戏的首选。

嗯，正式的介绍说完了，下面是关于 pymo 的来历八卦。

pymo 诞生于 2011 年 5 月，那时我刚刚完成了 NDS 版本秋之回忆的移植，正在意犹未尽的时候，想到了把秋之回忆移植到手机上的想法，那时我用的还是一款老旧的 S60v3 手机，上面的 Galgame 引擎只有一个 bug 多多、一点都不流畅且不支持中文的 ONS。这时掌叔的一个 demo 启发了我，可以使用 pyS60 来自己开发一个引擎。于是经过 3 个月的编写，第一个版本的 pymo 和秋之回忆 S60 版一起问世了。

受益于 python 开发的便捷，引擎的开发一开始没什么难度。但是后期要考虑到多游戏共存、不同屏幕分辨率的适配、配置和存档的向前兼容、不同用户环境的安装、向开发者开放哪些 API 等，还是很麻烦的。pymo 经过一年 7 个版本的演化，出了 18 款游戏，也因其流畅的效果、良好的操控性收获了良好的口碑。

因为之前使用谢叔的 AVG MAKER DS 的开发经历，pymo 的设计理念和 AMDS 很像，都是在受限的机能里，开发出特效简单但尽可能流畅的游戏。pymo 的很多指令设置都参考了 AMDS 的指令，甚至目录结构都和 AMDS 基本一样，因此有 AMDS 开发经验的移植者可以很快上手。

下面是 pymo 的一些优点：

- 配置要求极低，运行十分流畅
- 支持 jpg、bmp、png 等多种图片格式和 wav、mp3、amr、aac、midi 等多种音频格式
- 图片支持 256 级透明度
- 音频文件大小和长度不限
- 支持 BGM、语音和音效同时播放
- 立绘大小随意，支持同屏显示任意多的立绘，可以设置立绘的位置和图层顺序
- 选择肢支持动画提示
- 自适应横屏和竖屏设备。
- 100 个存档，记忆上次用户存读档位置
- 用户可设置字体和字号
- 支持多个游戏共存
- 使用资源预取机制，提升流畅度

当然，再好的引擎，没有优秀游戏的加持，玩家也不会安装。使用 pymo 制作游戏并不需要编程基础，游戏的存档、鉴赏系统这样复杂的地方都由引擎完成。游戏制作者只需要有基本的图片、音频转换能力，进行简单的脚本编写就可以完成一部游戏的开发。希望对 AVG 开发、移植感兴趣的朋友们，在阅读完这份文档之后，能够成功开发出自己的游戏。相信我，你将从中收获一份非凡的成就感。

第二章 AVG 的组成部分和移植方法简介

pymo 仅适合于 AVG 的制作。所谓 AVG 就是 Adventure Game 的缩写，以故事性剧情展开为主，基本特征是对话框式的对话、背景图片和人物立绘的组合表现剧情，以选择肢操纵的剧情分支，还有 CG 及音乐鉴赏系统等。AVG 一般包含如下 9 种游戏资源：

背景图片、人物立绘、背景音乐、音效、人物语音、视频、脚本、系统图片、存档。这些资源在 pymo 里都有对应的文件夹。具体在下一章介绍。

一般来说，AVG 的移植需要经过 4 个步骤：

一、将原游戏的资源包拆开，将特殊格式转换为通用格式。

这一步说难也不难，一般各种引擎都有大牛为你写了解包工具和转换工具。

二、将原游戏的脚本解密成文本格式，并且读懂它的语法，了解每一个指令是在干什么。这一步是最难的，如果是常见的几种引擎比如 NScripter、kirikiri 还好，都有脚本解密工具以及说明书给你用。如果是二进制的脚本比如 EntisGLS 引擎的游戏，就很难解出文本类型的脚本。另一方面，即使有文本类型的脚本，要读懂它，了解整个游戏的结构，也需要时间。

三、将图片、音频等资源转换成你用的那个引擎（这里就是 pymo）支持的格式。这个很简单，有时可能需要你用 photoshop 裁剪一下图片什么的。一般的引擎都会提供详细的转换教程，在下面几章也会介绍如何为 pymo 转换图片和音频格式。

四、使用你用的那个引擎的脚本语法，撰写新的脚本。





这一步虽然不是最难，但是是最费时间的，如果对游戏重现度要求高的话往往还需要一句一句地改特效参数。

当然，上面所说的只是游戏主体，除此之外还有系统图片的修改、鉴赏系统的移植等，这些可以稍后考虑。

因为 AVG 使用的引擎众多，这份 pymo 使用说明只涉及到上面的第 3、4 步。要想了解 1、2 步如何进行，请自行针对各引擎搜索相关资料。

一点忠告是，永远都不要使用蛮力干活，要选择偷懒的办法。比如图片和音频一定要学会批量处理。脚本的转换，简单的可以用 Ultraedit 的正则表达式替换，稍微有技术含量一点的可以自己写一个 python 小程序来处理文本，都是很快捷的方法，熟练以后一两天移植一款游戏也不是不可能。如果手工一行一行改，不仅容易出错，而且让移植变成了一件很痛苦的事情。

第三章 pymo 的目录结构

 bg	2011/12/30 12:22	文件夹	
 bgm	2011/12/30 12:22	文件夹	
 chara	2011/12/30 12:22	文件夹	
 save	2011/12/30 12:28	文件夹	
 script	2011/12/30 12:23	文件夹	
 se	2011/12/30 12:23	文件夹	
 system	2011/12/30 12:29	文件夹	
 video	2011/12/30 12:23	文件夹	
 voice	2011/12/30 12:22	文件夹	
 gameconfig.txt	2011/12/30 12:31	文本文档	1 KB
 icon.png	2011/12/29 18:08	ACDSee Photo ...	3 KB

首先我们了解 pymo 目录结构，上图是一个典型的 pymo 游戏文件夹，下面列出了 pymo 运行所必须拥有的文件夹和文件，如果缺少这些文件，pymo 将无法运行。

Python 目录

|----游戏目录

- |-----bg (用于放置背景文件或背景资源文件)
 - | |----- logo1.jpg (游戏运行显示的第一个 logo, 扩展名需要与 gameconfig.txt 中的配置符合)
 - | |----- logo2.jpg (游戏运行显示的第二个 logo, 扩展名需要与 gameconfig.txt 中的配置符合)
- |-----bgm (用于放置背景音乐文件)
- |-----chara (用于放置立绘文件)
- |-----save (用于放置存档文件)
- |-----script (用于放置脚本文件)
- |-----se (用于放置音效文件)
- |-----system (用于放置系统图片, 如文字框、选项框、鉴赏背景等, 图片必须是 png 格式)
 - | |----- alumbg_0.png、alumbg_1.png…… (CG 鉴赏系统背景)
 - | |----- cvThumb.png (CG 鉴赏系统中代表未解锁 CG 的图标)
 - | |----- menu.png (系统菜单的背景)
 - | |----- message.png、message_mask.png (对话框及其遮罩)
 - | |----- name.png、name_mask.png (名字框及其遮罩)
 - | |----- option.png、option_mask.png (选项框及其遮罩)
 - | |----- sel_highlight.png、sel_highlight_mask.png (选项高亮条目及其遮罩)
 - | |----- message_cursor.png、message_cursor_mask.png (对话末尾光标及其遮罩)
- |-----video (用于放置视频文件)
- |-----voice (用于放置语音文件)
- |-----gameconfig.txt (游戏配置文件)
- |-----icon.png (游戏图标)

第四章 游戏配置文件解析 (gameconfig.txt)

gameconfig.txt 必须为 utf-8 编码，文件中保存着游戏的配置，配置名称和数值由逗号隔开，**严格区分大小写，逗号前后不允许加入空格。**

配置条目	说明
gametitle, 秋之回忆 \n 作者 : chen_xin_ming	游戏名称，可以用\n 表示换行
platform, s60v5	运行平台，塞班版本是 s60v3 或 s60v5, android 和 PC 版是 pygame
engineversion, 0.4	开发此游戏使用的 pymo 版本，若用户使用的 pymo 版本比这个低，引擎将拒绝运行
scripttype, pymo	脚本格式，固定为 pymo
bgformat, .jpg	背景图格式，注意扩展名前要加上“.”
charaformat, .png	立绘格式，注意扩展名前要加上“.”
charamaskformat, .png	立绘遮罩格式，注意扩展名前要加上“.”
bgmformat, .mp3	背景音乐格式，注意扩展名前要加上“.”
seformat, .wav	音效格式，注意扩展名前要加上“.”
voiceformat, .mp3	人物语音格式，注意扩展名前要加上“.”
font, -1	字体，-1 表示由引擎自动选择， 发布游戏时记得将此值改为-1
fontsize, 26	字体大小，单位为像素
fontaa, 1	是否字体平滑，1：是，0：否
hint, 1	是否开启选项提示，1：是，0：否
prefetching, 1	是否资源预取，1：是，0：否
grayselected, 1	是否已选项变灰，1：是，0：否
playvideo, 1	是否播放视频，1：是，0：否
textspeak, 3	对话文字显示速度，0~5，数字越大越快
bgmvolume, 4	bgm 和 se 音量，0~10
vovolume, 5	语音音量，0~10
imagesize, 540, 360	资源包里的背景图的尺寸，s60v3 一般是 320, 240。s60v5 一般是 540, 360
startscript, start	游戏开始时调用的第一个脚本的文件名
nameboxorig, 0, 7	名字框和对话框的相对位移(x, y)，单位为像素
cgprefix, E	CG 前缀，以此为前缀命名的背景图在游戏里显示之后，会被鉴赏系统解锁。如果游戏的 CG 文件名没有特征明显的前缀，逗号后面可以什么都不写
textcolor, #ffffff	对话和名字使用的文字颜色
msgtb, 4, 0	对话框内的文字上下边距，单位为像素
msglr, 12, 12	对话框内的文字左右边距，单位为像素
namealign, middle	名字框内名字的对齐方式，left, middle, right 三种取值

第五章 pymo 中的坐标、颜色和时间表示方法

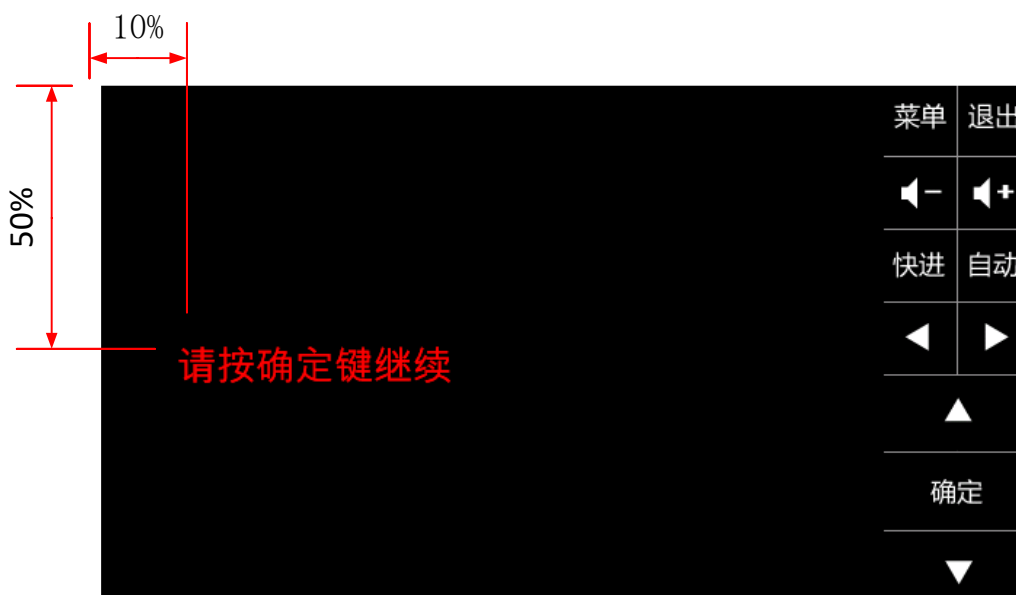
一、 坐标

因为 pymo 需要支持不同屏幕分辨率的手机，为了统一参数，脚本中使用的坐标都是屏幕长宽的百分比，可以使用小数。对于 S60v5，屏幕的宽度指的是除去虚拟键盘以外的画面宽度。例如这个命令：

```
#text 请按确定键继续,10,50,#FF0000,16
```

表示在横向坐标为屏幕宽度 10%，纵向坐标为屏幕宽度 50%的地方，用红色，16pt 大小写一个字符串“请按确定键继续”。

但有两处例外，#bg 和 #scroll 指令中的坐标使用的是相对于图像文件尺寸的百分比，具体请查看指令说明。



二、 颜色

颜色代码以#开头，后面跟 6 位 16 进制代码，分别表示 R、G、B 的值，大小写随意。例如#ff0000 表示红色，#FFFFFF 表示白色。

三、 时间

脚本编写时牵涉到的时间单位都为毫秒，例如这个命令：

```
#wait 2000
```

表示延时 2000 毫秒。

第六章 pymo 支持的图像、音频、视频格式及转换方法

一、 图片

pymo 支持 jpg、bmp、png 格式的图片。

二、 塞班版本图片的透明

塞班版本的 pymo 不会读取 bmp 和 png 图片中的 alpha 通道，图片的透明需要由两张图片实现，一张是图片本身（带或不带 alpha 通道均可，为了体积和读取速度，建议使用不带 alpha 通道的 256 索引色 png 图片），另一张是一张大小相等的 256 级灰度图，称为 " 遮罩 "，下面是一张立绘图和它的遮罩的示例。

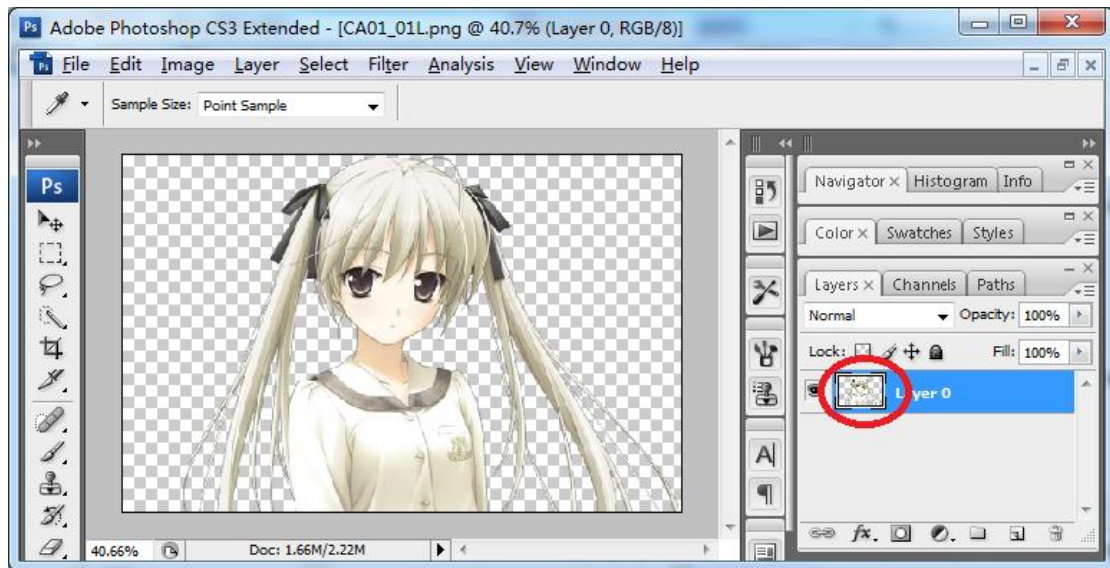


可以看到，遮罩上像素的灰度值决定了其对应图片的透明度，遮罩上的像素越黑，表示原图对应的像素越透明；遮罩上的像素越白，表示原图对应的像素越不透明。使用遮罩，就可以表示 256 级的透明度了。

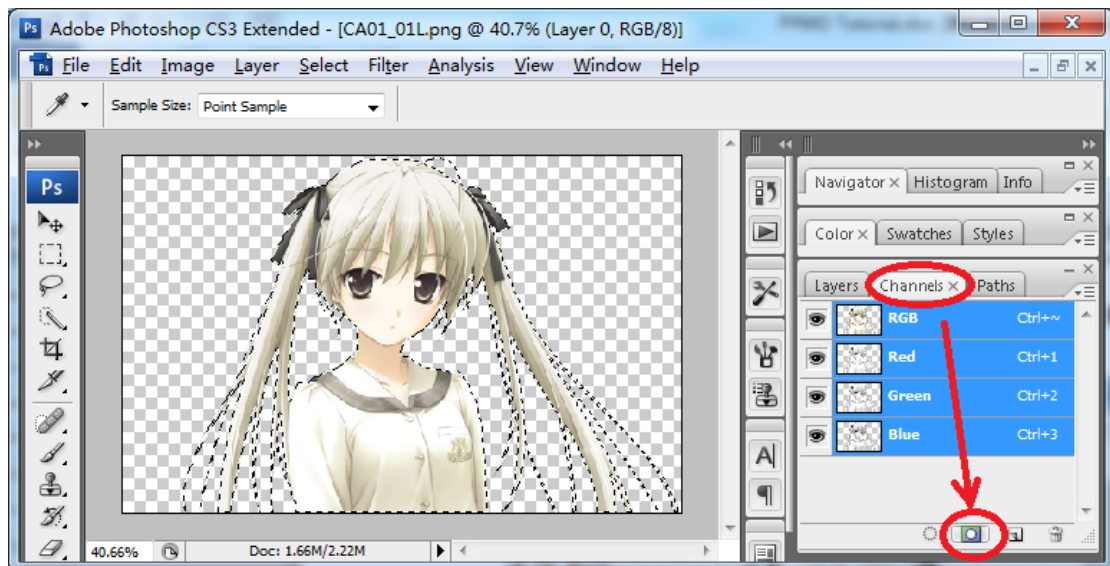
pymo 中的遮罩命名方式必须是原图片名+_mask+扩展名的命名方式，例如一张立绘的文件名为 A.png，那么它的遮罩就需要命名为 A_mask.png。

下面讲解一下如何使用 Photoshop 将带 alpha 通道的图片转化为一张遮罩（这里用的是英文版的 photoshop，用中文版的请自行对照）。

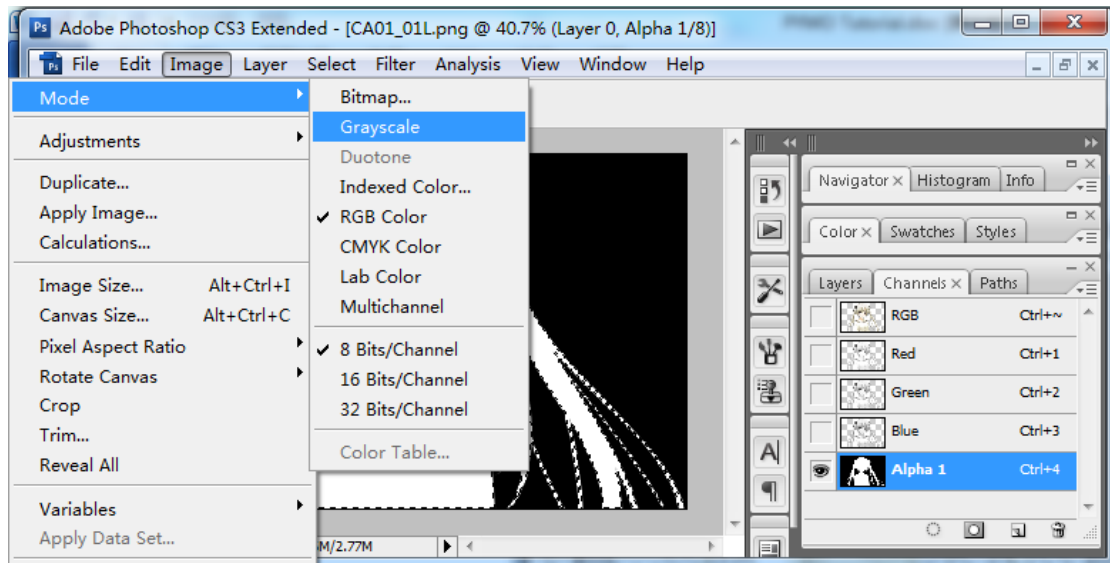
首先打开一张带 alpha 通道的 png 图片，按住 ctrl 键单击图层前面的缩略图，会根据透明度生成一个选区。



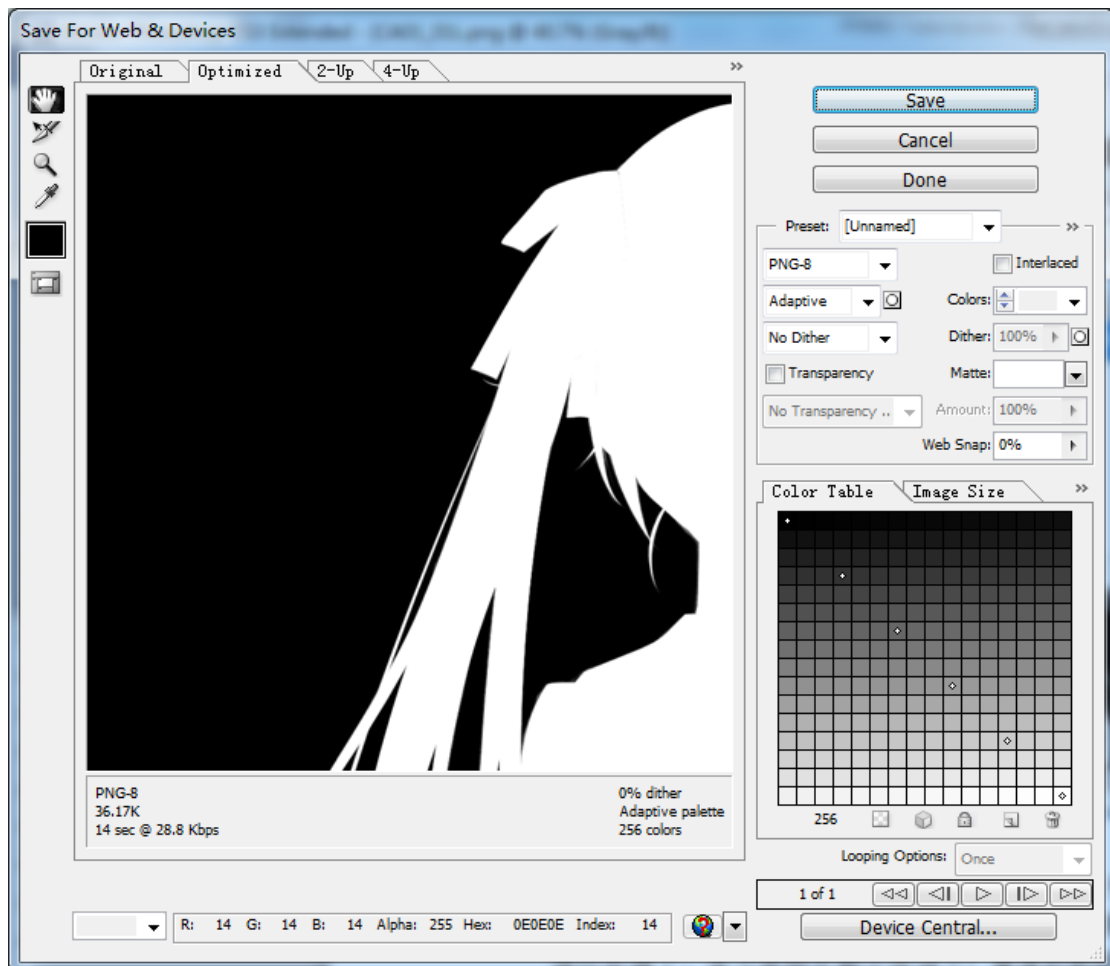
然后点击 Channels 选项卡，点击“Save selection as channel”按钮，会生成一个新的通道。



选中新通道前面的“眼睛”，同时取消其余通道前面的“眼睛”。单击新的通道使其高亮，然后选择菜单 Image->Mode->Grayscale。会出现 2 个提示，点 OK 即可。现在你已经获得了一个遮罩图！



选择菜单 File->Save for Web & devices，按照下图配置存储格式为 256 索引色，无透明，点 Save 即可。



上面仅仅是一张图片的处理方式，当我们有上千张图片需要处理的时候该怎么办呢？没关系，Photoshop 提供了批量处理的功能，只要把刚才的处理步骤录成 Action，就可以批量进行了，多少张图片都不在话下！具体的教程可以看这里：

<http://www.douban.com/note/72083205/>

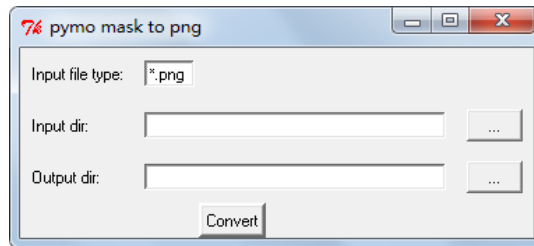
另外，处理图片时常常需要批量重命名，可以使用 ACDSee 进行批量重命名。具体可以看下面这篇日志里的“搜索替换命名法”一节

<http://space.30edu.com/0237012/ReadArticle.aspx?ID=23276258-ee87-4210-a2ab-59d77f22fa1e>

三、 安卓、PC 版本图片的透明

安卓和 PC 版使用带 alpha 通道的 png 图片来实现透明（即普通的带透明通道的 png）。

教程所附的 pymo_mask_to_png 工具可以将塞班格式的遮罩批量转换为带 alpha 通道的 png 图片。使用方法如下：



Input file type 填写塞班版的图片格式，一般是*.png。

Input dir 选择要处理的文件夹。

Output dir 选择存放处理结果的文件夹。

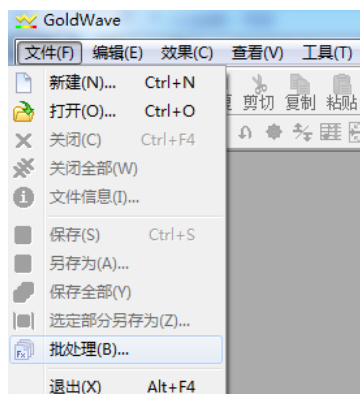
点 convert 即可批量转换。

四、 音频

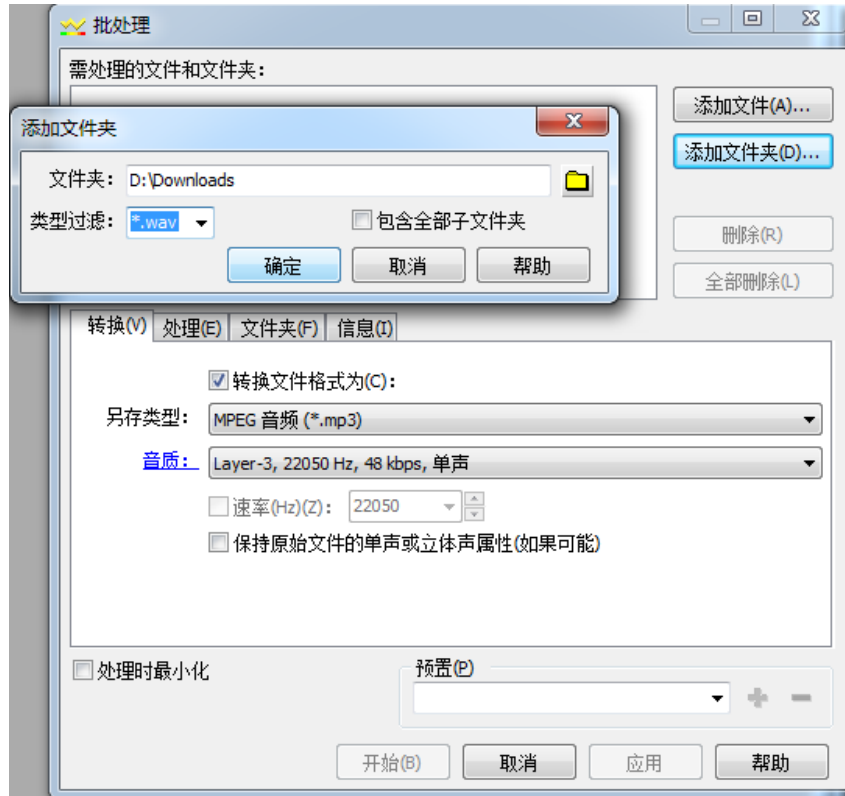
pymo 支持 wav、mp3、amr、aac、midi 格式，音频文件的大小和长度不限。

如果有较多的音频文件需要处理，推荐使用教程所附的 goldwave 软件进行批处理。下面简单介绍一下：

打开 goldwave，选择菜单“文件”——批处理。



添加录音文件所在文件夹——勾选转换文件格式以及选择另存类型和音质即可。例如我们需要把语音文件转换为 22050Hz，48kbps，单声道的 mp3 文件，可以如下设置。



选择输出文件夹后，点击开始即可。

五、 视频

Symbian 手机自带播放器可以播放的 mp4 视频参数比较严格，需要视频编码为 XVID,音频编码为 LC-AAC，帧率最好不要超过 15fps，mp4 封装类型也有特殊要求。下面讲解如何使用教程所附的 winmenc 编码 pymo 可用的 mp4 文件。

1. 执行 winmenc；点选 "About" 选项卡在 "language" 中选择 "schinese.ini"，"save config" 重新启动 winmenc 就出现中文版本界面了
2. 额外-载入配置 -S60v3.ini 或 S60v5.ini
3. 视频大小选项可以按照源文件比例自己选择参数调整，其余参数请不要调整。



4、在“批量”选项卡中，添加要转换的视频文件和输出目录，点“开始编码”即可。

六、 pymo 推荐的图片、音频格式及参数

虽然 pymo 对于图片和音频的格式要求并不严格，但是综合体积和质量考虑，最好按照下面的格式准备素材。

素材	推荐格式	推荐参数
背景图	jpg	大小：S60v3 为 320*240; S60v5 为 540*360
立绘及其遮罩	png	立绘本身为 256 索引色 png 图片，遮罩为 256 索引色灰度 png 图片。每个立绘的大小可以不一样，不过要记得根据背景图的大小进行同比例缩放。
背景音乐	mp3	44100Hz, 192Kbps, 立体声
音效、语音	mp3	22050Hz, 48Kbps, 单声

七、 系统图片的参数和要求

icon.png 和 system 文件夹下的图片大小有着特殊的要求，必须是 png 格式，大小要满足下面的条件：

素材	大小
icon.png	必须为 57*57
alumbg_0.png alumbg_1.png……	和背景图相同
cvThumb.png	随意，推荐 120*90
menu.png	S60v3: 240*144; S60v5: 360*216。当然别的大小也可以，但 pymo 不会根据不同屏幕大小进行缩放。菜单会被居中显示。
message.png message_mask.png	推荐 480*86，当然别的长宽比也可以，pymo 会根据不同屏幕大小进行缩放，并且根据长宽比智能地调整对话框的排版。

message_cursor.png 、 message_cursor_mask.png	推荐 24*24，接近方形就可以，pymo 会根据字体大小进行缩放
name.png 、 name_mask.png	推荐 140*52，当然别的长宽比也可以，pymo 会根据字体大小进行缩放，并且将人名居中显示。
option.png 、 option_mask.png	S60v3: 260*70; S60v5: 390*105，当然别的大小也可以，但 pymo 不会根据不同屏幕大小进行缩放。选项会被居中显示。
sel_highlight.png 、 sel_highlight_mask.png	推荐 260*48，当然别的长宽比也可以，pymo 会根据不同屏幕大小进行缩放。

第七章 制作你的第一个游戏！

经过上一章头晕目眩的格式讲解，是不是已经不知所云了呢？其实，pymo 开发包里提供了必要文件的范例，一开始使用这些默认文件就可以制作一个简单的游戏了。接下来我们将制作一小段游戏 demo。

一、 素材

教程附带了 2 个模板，首先根据自己的手机系统，选择相应的模板，模板里预先放好了如下素材：

bg 图片 logo1.jpg、logo2.jpg、BG_BLACK.jpg、BG001_H.jpg、BG018_H.jpg、title.jpg

立绘图片 AY04BA.png、AY04BA_mask.png、AY03BA.png、AY03BA_mask.png

背景音乐 BGM00.mp3

音效 SE02.wav

人物语音 PRO000.wav、PRO001.wav

以及必需的系统图片，这里不再列举。

二、 脚本

资源准备好后，就到了最关键的写游戏脚本了。

脚本的指令在第十章中作了详细说明，这里只用到一些基本的指令。

我们用记事本建立一个文本 start.txt，添加如下内容。

```
#bg BG_BLACK
#say 很久以来，一切都未曾改变。
#say 沐浴着和煦的阳光，伴随着四季悄然的轮回。
#say 从未怀疑过什么，平凡的日子就这样度过……
#say 这一切，你几乎感觉不到，因为，它是那样的从容……
#say ……忽然间才发现……在不经意的时候，夏天已悄然而去……
#say 我已深深地感觉到……秋天，来了。
#bg BG001_H
#se SE02
#say 生活，一如往常。
#say 被住在隔壁的伙伴叫醒后，我拖着一副慵懒的身子走出家门。
#say 智也，「…久等了……」
#say 她用甜甜的笑脸，回应着我的话。
#say 又一个『今天』开始了。
#bg BG018_H
#bgm BGM00
#say 这是一段非常紧张的时间。
#say 感受着早晨沁人心脾的清凉空气，我们走在上学的路上。
#chara 0,AY04BA,50,0,300
#vo PRO000
```

```
#say 彩花,「哎, 智也。这个星期天……有时间吗?」  
#say 智也,「时间倒是有……怎么了?」  
#chara 0,AY03BA,50,0,300  
#vo PRO001  
#say 彩花,「呵呵呵……我呀……」  
#say 她突然转过身,神秘地笑着。  
#bgm_stop  
#change start
```

将文本另存为 utf-8 编码（一定要记住 pymo 只接受 utf-8 编码的文本文件哦，否则会直接报错退出），然后放入 script 文件夹。

三、 运行

把 Python 文件夹置于存储卡的根目录下，在手机上运行 pymo 即可体验这一小段剧情。

怎么样，是不是很简单？更多高级指令的应用，请见第十二章里的指令列表。要想找范例请参考悠之空、沙耶之歌的脚本。

第八章 一些脚本范例

一、 游戏开始界面

```
;播放背景音乐 BGM00
#bgm BGM00
;创建名为 START 的标签
#label START
;插入背景图 title
#bg title,BG_NOFADE,BG_VERYFAST

;创建 4 个选项，名为开始,继续,鉴赏,结束，使它们在点 (3,67) 到点 (50,100) 所画成的矩形内居中排列，
按钮颜色为#3E8BF4，5 个选项的值保存为 FSEL，大小由 0 至 4 依次递增
#select_text 4,开始,继续,鉴赏,结束,3,67,50,100,#3E8BF4,1
;如果 FSEL 的值为 0（就是选了第一个选项），就跳转至这个脚本内的 NEW_GAME 标签
#if FSEL=0,goto NEW_GAME
;如果 FSEL 的值为 1（就是选了第二个选项），就跳转至这个脚本内的 CONTINUE_GAME 标签
#if FSEL=1,goto CONTINUE_GAME
;如果 FSEL 的值为 2（就是选了第三个选项），就跳转至这个脚本内的 YOUR_MEMORIES 标签
#if FSEL=2,goto YOUR_MEMORIES
;如果 FSEL 的值为其他值，就跳转至这个脚本内的 END_GAME 标签
#goto END_GAME

;创建名为 NEW_GAME 的标签
#label NEW_GAME
;停止播放背景音乐
#bgm_stop
;插入背景图 BG_BLACK，插入方法为屏幕变暗然后淡入，时间 3000 毫秒
#bg BG_BLACK, BG_FADE, 3000
;在这张背景图停留 2000 毫秒
#wait 2000
跳转至 script 文件夹下名为 start.txt 的脚本
#change start
#goto START

;创建名为 CONTINUE_GAME 的标签
#label CONTINUE_GAME
使用 load 指令进入读档画面
#load
;若没有选择任何档，跳转至这个脚本内的 START 标签
```

```

#goto START

;创建名为 YOUR_MEMORIES 的标签
#label YOUR_MEMORIES
;插入背景图 title
#bg title,BG_NOFADE,BG_VERYFAST
#select_text 3,图片,音乐,返回,3,67,50,100,#3E8BF4,1
#if FSEL=0,goto ALBUM
#if FSEL=1,goto MUSIC
#goto START

#label ALBUM
;停止播放背景音乐
#bgm_stop
;使用 album 命令进入 CG 鉴赏画面
#album
#goto YOUR_MEMORIES

#label MUSIC
;停止播放背景音乐
#bgm_stop
;使用 music 命令进入音乐鉴赏画面
#music
#goto YOUR_MEMORIES

;创建名为 END_GAME 的标签
#label END_GAME
;因为运行到脚本末尾，所以退出了

```

二、 选择支示例

```

#bg BG018_H
#chara 0,AY03BA,50,0,300
#say 彩花,「太好了。其实,我今天在里面放了好多智也喜欢吃的东西……知道吗?」
#say 智也,「我喜欢吃的东西?」
#say 呀……
#sel 3
咸菜
豆沙
咖喱

#if FSEL=0,goto IF_LABEL_3

```

```

#if FSEL=1,goto IF_LABEL_4
#goto IF_LABEL_5

#label IF_LABEL_3
#say 智也,「放咸菜了吗?」
#chara 0,AY04BA,50,0,300
#say 彩花,「我想,在三明治里面是不能放咸菜的……」
#goto IF_LABEL_2_END

#label IF_LABEL_4
#say 智也,「放豆沙了吗?」
#chara 0,AY04BA,50,0,300
#say 彩花,「我想,在三明治里面是不能放豆沙的……」
#goto IF_LABEL_2_END

#label IF_LABEL_5
#say 智也,「好像放咖喱了吧」
#chara 0,AY04BA,50,0,300
#say 彩花,「我想,在三明治里面是不能放咖喱的……」
#goto IF_LABEL_2_END

#label IF_LABEL_2_END
#say 智也,「是,是吗……」
#say 彩花,「……真的不知道吗?」
#say 智也,「啊,开个玩笑。我说不知道,只不过是故意气气你的」
#say 我急忙辩解。

```

三、 每次游戏开始界面显示不同的图片（使用全局变量）

;以 S 开头的变量会作为全局变量处理，这种变量保存在 golbal.sav 中，在游戏开始时就载入内存，并且不受读档的影响，适用于保存通关记录，或者用于游戏开始界面的一些变化花样，例如这个例子。

```

#bgm BGM00
#if S02=0, goto SHOW_BG_01
#if S02=1, goto SHOW_BG_02
#goto SHOW_BG_03

#label SHOW_BG_01
#bg BG001_H
#goto INC_VAR

#label SHOW_BG_02
#bg BG018_H

```

```
#goto INC_VAR
```

```
#label SHOW_BG_03
```

```
#bg title
```

```
#goto INC_VAR
```

```
#label INC_VAR
```

```
#add S02,1
```

```
#if S02<3, goto START
```

```
#set S02,0
```

```
#label START
```

```
#select_text 4,开始,继续,鉴赏,结束,3,67,50,100,#3E8BF4,1
```

第九章 鉴赏系统的编写

一般的 AVG 都会有 CG 和音乐鉴赏系统，以便于玩家在通关后浏览游戏中的精彩画面。pymo 的鉴赏系统只需要编写 2 个列表文件，然后配合相应的背景图就可以制作完成。

一、CG 鉴赏

pymo 内建了 CG 图片的解锁功能，即玩家在游戏中玩到了某张 CG，pymo 会将其记录在 global.sav 中，在鉴赏系统中这张 CG 就会变得可见。

CG 和普通 bg 图片的区别是通过 gameconfig.txt 里的 cgprefix 条目区分的。举个例子，假如你要移植的游戏的 CG 文件名都是以“E”开头的，那么可以写为

```
cgprefix,E
```

其他前缀以此类推。假如你要移植的游戏里的 CG 没有明显的前缀，可以把此项留空，即写为

```
cgprefix,
```

要建立一个 CG 鉴赏系统，首先在 script 文件夹下建立一个名为 album_list.txt 的文本文件，要求是 UTF8 编码，文件最后记得空一行。文件内容类似于下面：

```
1,3,prima,EV_SH01A,EV_SH01B,EV_SH01C
1,1,brio,EV_SH02A
1,2,vivace,EV_SH03A,EV_SH03B
1,1,espressivo,EV_SH04A
1,2,contenerezza,EV_SH14A,EV_SH14B
2,1,dolcezza,EV_SH13A
2,1,energiaco,EV_SH05A
2,1,stringendo,EV_SH15A
2,2,tacet,EV_SH08A,EV_SH08B
2,5,conanima,EV_SH06A,EV_SH06B,EV_SH06C,EV_SH06D,EV_SH06E,1
```

这个文件的每一行都表示鉴赏系统里的一个 CG。例如

```
1,3,prima,EV_SH01A,EV_SH01B,EV_SH01C
```

第一个数表示这张 CG 在第一页，第二个数表示这张 CG 包含了 3 张图片，第三个参数是图片的名称，如果没有可以留空，后面的 3 个参数就是 3 个图片的文件名。

每行的末尾还可以加一个参数 1，比如

```
1,3,,EV_SH01A,EV_SH01B,EV_SH01C,1
```

表示这张 CG 无需解锁就可以直接显示在鉴赏系统里。

然后需要按照页数，在 system 文件夹下面制作几张 CG 鉴赏系统的背景图片，分别命名为 albumbg_0.png、albumbg_1.png、albumbg_2.png……（注意编号从 0 开始，第一页的文件名是 albumbg_0.png）

使用 photoshop 按如下图所示的坐标放上缩略图，每页最多放 25 个。注意图上的坐标单位是 bg 尺寸的百分比，请自行根据 S60v3 和 v5 的尺寸换算成像素。（s60v3 是 320*240。s60v5 是 540*360）

最后在 system 文件夹下放一张 cvThumb.png，代表未解锁 CG 的图标。

要在脚本里调用 CG 鉴赏系统，使用如下的指令：

```
#album
```



用户退出鉴赏系统之后，会接着运行脚本的下一条指令。

Pymo 0.9 版开始，#album 指令可以指定列表文件名。例如：

```
#album album0
```

表示要载入的列表文件名为 album0.txt，列表文件格式和以前一样。相应的背景文件名为 album0_0.png, album0_1.png, album0_2.png... 用于生成背景的底图为 album0.png。

二、 自动生成 CG 鉴赏缩略图

Pymo 0.7 版加入了为开发者自动生成 CG 鉴赏图的功能，首先按照上面的格式写好一个 album_list.txt 文件，然后把一张 albumbg.png 放入 system 文件夹作为底图，运行 pymo，进入 CG 鉴赏模式，把所有页翻一遍，就会在 system 下生成全部的 CG 鉴赏缩略图。

注意：为防止开发者忘记删除 albumbg.png，缩略图生成完毕后，pymo 会自动删除 albumbg.png。

三、 音乐鉴赏

音乐鉴赏就简单多了，只需要在 script 文件夹下建立一个名为 music_list.txt 的文本文件，要求是 UTF8 编码，文件最后记得空一行。文件内容类似于下面：

BGM_SORA, 01. フタリ
BGM01, 02. 道の先、空の向こう
BGM03, 03. 優しい風が吹くあの場所で
BGM04, 04. 子守歌じゃないのに
BGM05, 05. スローライフに憧れて
BGM06, 06. 大空と紙飛行機
BGM07, 07. 膝枕で過ごす放课后
BGM08, 08. 素直と不器用と意地っ張り
BGM09, 09. 黄昏の帰り道
BGM10, 10. 頬を伝う冷たい涙
BGM11, 11. 漆黒の思惑
BGM12, 12. 気持ちの狭間
BGM13, 13. むふ～
BGM14, 14. よーしやっちまえ!
BGM15, 15. 楽しんだ今が勝ち!
BGM16, 16. 寂しい夜
BGM17, 17. 無くならない大事な心
BGM18, 18. 追い求めてきたもの
BGM19, 19. 星の滴
BGM20, 20. 未来へ踏み出す一步
BGM21, 21. 开辟新地
BGM02, 22. 夜明けのプリズム

这个文件的每一行都表示鉴赏系统里的一个音乐。例如

BGM_SORA, 01. フタリ

第一个参数是 bgm 的文件名，第二个参数是显示在列表里的音乐名称，可以使用\n 换行。就是这么简单！要在脚本里调用音乐鉴赏系统，使用如下的指令：

```
#music
```

用户退出鉴赏系统之后，会接着运行脚本的下一条指令。

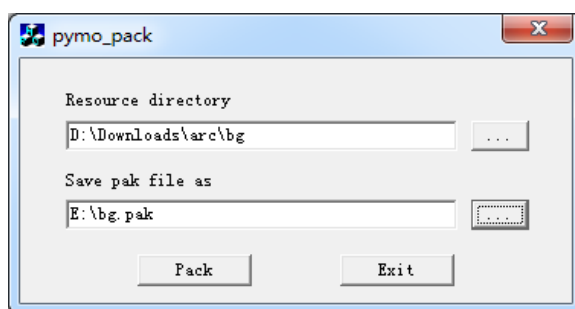
第十章 游戏资源的打包和发布

一、 资源打包

在游戏开发完毕之后，发布之前。最好进行游戏资源的打包。否则小文件拷进存储卡很慢，图片和语音也会显示在手机自带的多媒体管理里，造成手机变卡。

支持打包的资源类型是 bg, chara, se, voice。不支持打包的资源类型是 bgm, system, script, video。

使用开发包里的 pymo_pack.exe 进行打包。Resource directory 选择要打包的文件夹，文件夹下所有的文件都会被打包，不过不包括子文件夹。然后选择保存的文件名，点 Pack 就可以打包了。



打包的文件名必须和其所在的目录名一致，就像 bg.pak、chara.pak、se.pak、voice.pak 这样的，要把 pak 文件放到相应的资源文件夹下，pymo 才会认。

如果某目录下有 pak 文件，pymo 将只从 pak 文件中读取资源。所以要打包就整个文件夹一起打包，不能只打包一部分。

二、 游戏发布

游戏发布前，请仔细分别检查 S60v3 和 S60v5 的 gameconfig.txt 文件，**尤其要注意的是里面的 font 项要改成-1!**

另外要针对 S60v3 和 S60v5 分别制作不同分辨率的 bg、chara、video，以及一部分 system 图片。声音资源和脚本可以通用。

第十一章 pymo android 版、PC 版注意事项

android 版和 PC 版使用同一个数据包格式。由于它们和塞班版的游戏数据稍有区别，所以习惯了开发塞班版的移植者，在转向 android 平台时，需要注意以下几个方面：

- 1、android 版的透明图片使用单独一张带 alpha 通道的 png 图片来实现，而不是像塞班使用一张图片和一张遮罩。移植教程内附 image_converter_v1.0，可以将塞班版的图片+遮罩批量转换为 android 版的透明图片，具体请看教程。
- 2、android 和 PC 版的 gameconfig.txt 里的 platform 项应该设为 pygame。
- 3、android 版的音频只支持 mp3 格式。
- 4、android 版的 bg、chara 图片大小没有限制，推荐使用 800*600 或更高分辨率，注意 gameconfig.txt 里的 imagesize 项要相应地设为 bg 的大小。
- 5、由于 PC 版运行速度快，修改脚本方便，推荐先做好 PC 版的游戏，再制成塞班版。

第十二章 脚本指令说明书

脚本文件的扩展名是.txt，pymo 只接受 utf-8 编码的文本文件。如果文本文件使用其他编码 pymo 将直接报错退出。

每条指令均为#开头，以半角逗号,分隔各个参数，指令是大小写敏感的。

所有需要调用资源的语句都不用写资源的扩展名，扩展名在 gameconfig.txt 里指定。

pymo 会忽略一切不以#开头的行，因此要写注释什么的只要不以#开头就可以，但是为了规范，推荐注释以分号;开头。

下面的指令原型中，使用[]括起来的参数表示这个参数是可以省略的，在实际编写的时候不要写上[]。

如果开发者觉得需要更多指令，可以与我联系添加。

一、 对话文字显示指令

1. #say

1.1 功能:

人物对话显示功能，在对话框中进行显示。

1.2 脚本指令原型:

```
#say [name,] content
```

1.3 参数说明:

1. name: 说话人物名字。省略此参数表示不需要名字
2. content: 说话内容。注意内容中不要包括半角逗号。可以使用\n换行。

1.4 例:

```
#say 智也,「…久等了……」  
#say 她用甜甜的笑脸, \n 回应着我的话。
```

2. #text

2.1 功能:

文字显示功能，注意用此指令显示的文本不会记入对话记录和存档，因此不要用#text 代替#say 作为主要的文字显示手段。

#text 显示的文字会被这些指令所清除: #text_off, #bg, #scroll, #chara_cls a。除此之外不会被清除。

2.2 脚本指令原型:

```
#text content, x1,y1,x2,y2,color,size, show_immediately
```

2.3 参数说明:

1. content: 显示内容。注意内容中不要包括半角逗号。可以是变量名（但不能以变量名+文本的形式混合出现）。可以使用\n换行。

2. x1: 显示范围的左上角顶点的 x 坐标，单位是屏幕宽度的百分比

3. y1: 显示范围的左上角顶点的 y 坐标, 单位是屏幕高度的百分比

4. x2: 显示范围的右下角顶点的 x 坐标, 单位是屏幕宽度的百分比

5. y2: 显示范围的右下角顶点的 y 坐标, 单位是屏幕高度的百分比

6. color: 文字的颜色

7. size: 文字的大小, 单位是 pt, 注意在不同机型上引擎会自动根据屏幕大小显示不同的大小, 例如脚本里写上这个参数为 16, 在 S60v3 上会显示为 16 像素大小, 在 S60v5 上会显示为 $16 \times 360 / 320 = 18$, 在 800×480 的安卓机器上会显示为 $16 \times 480 / 320 = 24$ 。

6. show_immediately: 是否立即显示, 为 0 则逐字显示并在显示完之后显示光标, 为 1 则立即显示且不显示光标。

2.4 例:

```
#text 第一章,10,10,50,50,FF0000,16,1
```

```
#text F12,10,10,50,50,FF0000,16,0
```

3. #text_off

3.1 功能:

消除屏幕上所有文字, 只留下 bg 和立绘。

3.2 脚本指令原型:

```
# text_off
```

3.3 参数说明:

3.4 例:

```
# text_off
```

4. #waitkey

4.1 功能:

等待用户按键输入, 然后继续执行下一指令。如果 5 秒后还没有按键, 则继续执行下一条指令。

4.2 脚本指令原型:

```
#waitkey
```

4.3 参数说明:

4.4 例:

```
#waitkey
```

5. #title

5.1 功能:

设置章节标题, 此标题会作为存档的标题。**注意此命令不会将标题显示在屏幕上, 要显示在屏幕上, 需要在#title 命令之后使用#title_dsp 命令。**

5.2 脚本指令原型:

```
#title content
```

5.3 参数说明:

1. content: 标题内容。注意内容中不要包括半角逗号。

5.4 例:

```
#title 序章
```

6. #title_dsp

6.1 功能:

将当前的章节标题显示在屏幕上。

6.2 脚本指令原型:

```
#title_dsp
```

6.3 参数说明:

6.4 例:

```
#title 序章  
#title_dsp
```

二、 图像相关操作指令

1. #chara

1.1 功能:

显示立绘，支持任意数量立绘同屏显示。

1.2 脚本指令原型:

```
#chara charaID1,filename1,position1,layer1[,charaID2,filename2,position2,  
layer2...],time
```

1.3 参数说明:

1. charaID: 立绘 ID，可以是任意非负整数。如果之前已经有此 ID 的立绘，代替之。
2. filename: 立绘文件名，文件名后不写文件后缀。如果文件名为 NULL 的话，表示擦除对应的立绘 ID。
3. position: 立绘中心在屏幕上的 x 坐标，单位是屏幕宽度的百分比，例如 50 表示在屏幕中央显示。
4. layer: 立绘图层，可以是任意非负整数。多个立绘中，图层编号小的显示在最下面。
5. time: 立绘淡入时间，单位是毫秒。

1.4 例:

```
#chara 0,SM02AMA,25,0,400  
#chara 0,SM02AMA,25,1,1,SN01AMA,75,2,500  
#chara 0,SM02AMA,16,1,1,SN01AMA,50,2, 2,NULL,0,0,400
```

2. #chara_cls

2.1 功能:

擦除一个或全部立绘。

2.2 脚本指令原型:

```
#chara_cls charaID,[time]
```

2.3 参数说明:

1. charaID: 立绘 ID, 如果此参数为 a 则清除所有立绘。
2. time: 立绘淡出时间, 单位是毫秒, 此参数如省略默认为 300 毫秒。

2.4 例:

```
#chara_cls a  
#chara_cls 0,500
```

3. #chara_pos

3.1 功能:

更改立绘的位置, 并显示到屏幕上。

3.2 脚本指令原型:

```
#chara_pos charaID,new_x,new_y, coord_mode
```

3.3 参数说明:

1. charaID: 立绘 ID。
2. new_x: 新的立绘 x 坐标。
3. new_y: 新的立绘 y 坐标。
4. coord_mode: 坐标模式, 详见#chara_y 说明。

3.4 例:

```
#chara_pos 0,30,0,5
```

4. #bg

4.1 功能:

加载背景, 如果当前屏幕上有立绘将被清除。可以省略参数 4、5, 也可以省略参数 2、3、4、5。不能有其他省略方式。

4.2 脚本指令原型:

```
#bg filename[, transition,time[,x,y]]
```

4.3 参数说明:

1. filename: 背景图象文件。文件名后不写文件后缀。
2. transition: 渐变效果名称, 可以是: BG_NOFADE -无渐变, BG_ALPHA -Alpha 淡入, BG_FADE -淡出成黑屏后淡入, 或者是遮罩图片的文件名, 遮罩图片要放在 system 文件夹下并且为 png 格式。此参数省略后默认为 BG_ALPHA。
3. time: 淡入时间, 单位是毫秒。此参数省略后默认为 300 毫秒。
4. x: 部分显示的 x 坐标。例如一张大的 bg, 想把其中以(x,y)点为左上角的部分显示出来, 可以使用此参数。此参数省略后默认为 0。

5. y: 部分显示的 y 坐标。此参数省略后默认为 0。

4.4 例:

```
#bg BG001_H
#bg BG001_H, BG_FADE, 500
#bg BG001_H, BG_FADE, 500, 10, 20
#bg BG001_H, trans_mask, 500
```

5. #flash

5.1 功能:

屏幕闪光效果。

5.2 脚本指令原型:

```
#flash color,time
```

5.3 参数说明:

1. color: 闪光颜色。
2. time: 闪光持续时间。

5.4 例:

```
#flash #FF0000,1000
```

6. #quake

6.1 功能:

画面振动效果。

6.2 脚本指令原型:

```
#quake
```

6.3 参数说明:

6.4 例:

```
#quake
```

7. #fade_out

7.1 功能:

屏幕淡出，淡出后对背景层和立绘层的操作将不可见。可以利用此命令在后台完成更改背景和立绘、文本等操作。

注意:此指令的意义在于“暂时不显示 bg 或 chara 指令的效果,并在 fade_in 之后才显示出来”。如果你需要同时改变 bg 和多个 chara 又不希望在屏幕上显示出逐个改变的过程,可以先 fade_out, 然后改 bg 和 chara, 最后 fade_in。

记住一定要 fade_in! 否则 pymo 将停留在 fade_out 状态, 很多特效无法显示。

7.2 脚本指令原型:

```
#fade_out color,time
```

7.3 参数说明:

1. color: 淡出颜色。

2. time: 淡出时间。

7.4 例:

```
#fade_out #000000,1000
```

8. #fade_in

8.1 功能:

屏幕淡入, 淡入后背景层和立绘层将恢复为可见。

8.2 脚本指令原型:

```
#fade_in time
```

8.3 参数说明:

1. time: 淡入时间。

8.4 例:

```
#fade_in 1000
```

9. #movie

9.1 功能:

播放视频。只支持开发包里附带的 winmenc 转出来的 mp4 格式文件。

9.2 脚本指令原型:

```
#movie filename
```

9.3 参数说明:

1. filename: 视频文件名, 不加扩展名。

9.4 例:

```
#movie opening
```

10. #textbox

10.1 功能:

更换对话框图片, 因为系统默认的对话框和名字框是 message.png 和 name.png, 所以要恢复原来对话框可以用 #textbox message,name

10.2 脚本指令原型:

```
#textbox message,name
```

10.3 参数说明:

message: 对话框的图片文件名

name: 名字框的图片文件名

10.4 例:

```
#textbox message2,name2
```

11. #chara_quake

11.1 功能:

立绘左右振动效果。

11.2 脚本指令原型:

```
#chara_quake charaID1, charaID2,.....
```

11.3 参数说明:

charaID1, charaID2,.....: 立绘 ID, 可以有多个

11.4 例:

```
#chara_quake 0  
#chara_quake 0,1
```

12. #chara_down

12.1 功能:

立绘下沉效果。

12.2 脚本指令原型:

```
# chara_down charaID1, charaID2,.....
```

12.3 参数说明:

charaID1, charaID2,.....: 立绘 ID, 可以有多个

12.4 例:

```
#chara_down 0  
#chara_down 0,1
```

13. #chara_up

13.1 功能:

立绘上跳效果。

13.2 脚本指令原型:

```
# chara_up charaID1, charaID2,.....
```

13.3 参数说明:

charaID1, charaID2,.....: 立绘 ID, 可以有多个

13.4 例:

```
#chara_up 0  
#chara_up 0,1
```

14. #scroll

14.1 功能:

滚动图片, 如果当前屏幕上有立绘将被清除。

14.2 脚本指令原型:

```
#scroll filename, startx, starty, endx, endy, time
```

14.3 参数说明:

1. filename: 背景图象文件。文件名后不写文件后缀。

2. startx, starty, endx, endy: 滚动的开始坐标和结束坐标。注意这里的单位是像素坐标占此图像文件尺寸的百分比。例如一张 640*240 的图片, 希望起始点像素坐标为(0,0), 结束点像素坐标为(320,0), 那么这里的坐标要写成(0,0)和(50,0)

3. time: 滚动时间, 单位是毫秒。

14.4 例:

```
#scroll B34a,0,0,50,0,10000
```

15. #chara_y

15.1 功能:

显示立绘，可以指定立绘的纵横坐标。有 7 种坐标模式可供选用。

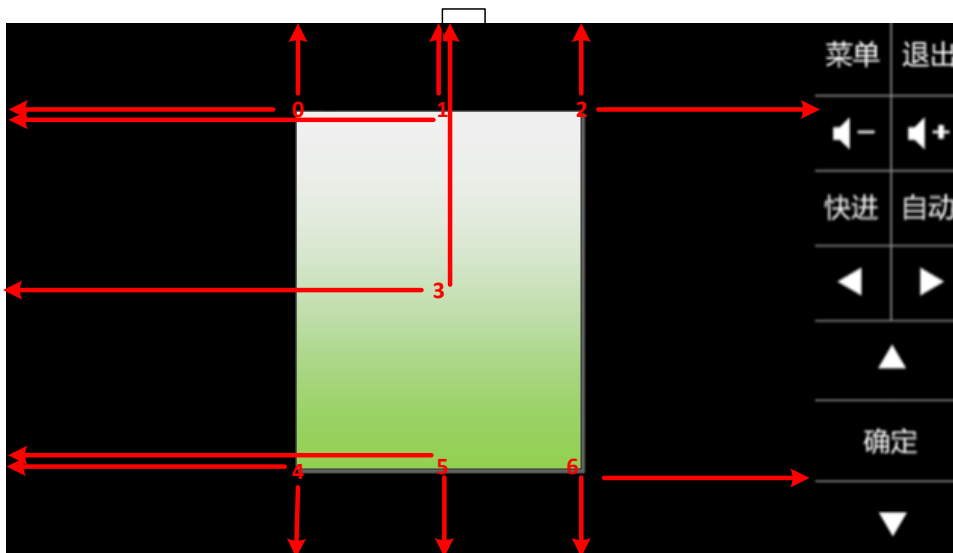
15.2 脚本指令原型:

```
#chara_y coord_mode, charaID1,filename1,x1,y1,layer1  
[,charaID2,filename2,x2,y2, layer2...],time
```

15.3 参数说明:

1. coord_mode: 坐标模式，共有 7 种:

coord_mode	x 坐标表示的含义	y 坐标表示的含义
0	立绘左沿距屏幕左沿的距离	立绘上沿距屏幕上沿的距离
1	立绘中点距屏幕左沿的距离	立绘上沿距屏幕上沿的距离
2	立绘右沿距屏幕右沿的距离	立绘上沿距屏幕上沿的距离
3	立绘中点距屏幕左沿的距离	立绘中点距屏幕上沿的距离
4	立绘左沿距屏幕左沿的距离	立绘下沿距屏幕下沿的距离
5	立绘中点距屏幕左沿的距离	立绘下沿距屏幕下沿的距离
6	立绘右沿距屏幕右沿的距离	立绘下沿距屏幕下沿的距离



2. charaID: 立绘 ID，可以是任意非负整数。如果之前已经有此 ID 的立绘，代替之。

3. filename: 立绘文件名，文件名后不写文件后缀。如果文件名为 NULL

的话，表示擦除对应的立绘 ID。

4. x: 立绘在屏幕上的 x 坐标，单位是屏幕宽度的百分比。
5. y: 立绘在屏幕上的 y 坐标，单位是屏幕高度的百分比。
6. layer: 立绘图层，可以是任意非负整数。多个立绘中，图层编号小的显示在最下面。
7. time: 立绘淡入时间，单位是毫秒。

15.4 例:

```
#chara_y 1,0,SM02AMA,25,10,0,400  
#chara_y 0, 0,SM02AMA,25,20,1, 1,SN01AMA,75,10,2,500
```

16. #chara_scroll

16.1 功能:

立绘一面滑动一面淡入，仅支持一张立绘，可以指定立绘的横纵坐标。有 7 种坐标模式可供选用。

此指令还有一种简化版本，作用为使当前已存在于屏幕上的立绘以当前位置为起始点，滑动到中止点。

16.2 脚本指令原型:

```
#chara_scroll coord_mode, charaID, filename, startx, starty, endx, endy,  
beginalpha ,layer, time
```

或者

```
#chara_scroll coord_mode, charaID, endx, endy, time
```

16.3 参数说明:

1. coord_mode: 坐标模式，共有 7 种。
2. charaID: 立绘 ID，可以是任意非负整数。如果之前已经有此 ID 的立绘，代替之。
3. filename: 立绘文件名，文件名后不写文件后缀。
4. startx: 滑动开始点的 x 坐标，单位是屏幕宽度的百分比。
5. starty: 滑动开始点的 y 坐标，单位是屏幕高度的百分比。
6. endx: 滑动结束点的 x 坐标，单位是屏幕宽度的百分比。
7. endy: 滑动结束点的 y 坐标，单位是屏幕高度的百分比。
8. beginalpha: 开始时的立绘透明度，0~255，0 为不透明，255 为完全透明。
9. layer: 立绘图层，可以是任意非负整数。多个立绘中，图层编号小的显示在最下面。
10. time: 立绘淡入时间，单位是毫秒。

16.4 例:

```
#chara_scroll 5,0,SM02AMA,0,0,50,0,130,1,400  
#chara_scroll 5,0,100,0, 400
```

17. #anime_on

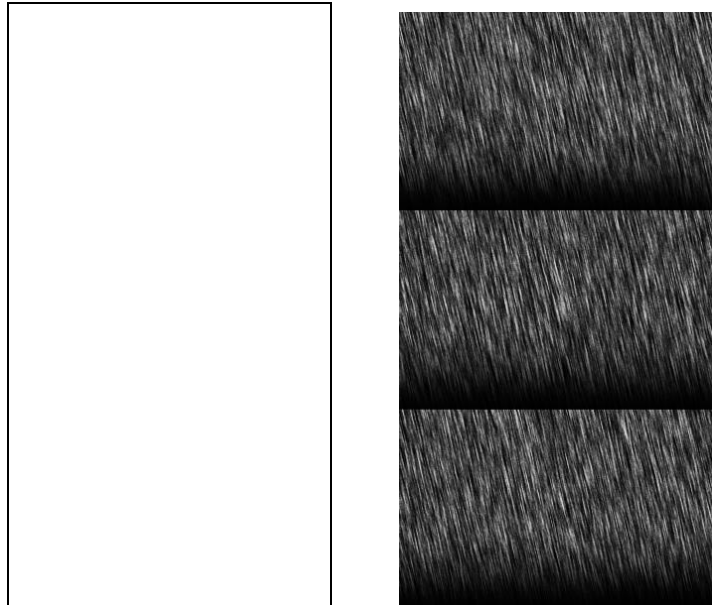
17.1 功能:

显示一个逐帧播放的动画（类似于动态 gif 图片的效果），动画显示在最顶层图层（对话框上层）。可以用于模仿动态雨、雪、樱花飘落的特效，或是像 clannad 里光

玉一样的动画效果。可以单次播放也可以循环播放。循环播放的话，要调用#anime_off 停止。同屏只能存在一个动画，如果使用了两次#anime_on 命令，后一次的动画会替代前一次的。

注意：动画图片过大的话会消耗较多内存，程序跳出的几率会增加。每帧播放间隔最好不要小于 200ms，否则会显得很卡。

所有的动画帧都集中在一张图片及其遮罩上，图片及其遮罩要放置在 system 目录下，格式为 png。下面是一张下雨效果图片的示例：图片从上到下分别是帧 1、2、3，pymo 会按照 num 的值平均分割这张图片，这也意味着每帧的大小必须一样。



17.2 脚本指令原型：

```
#anime_on num, filename, x ,y, interval, isloop
```

17.3 参数说明：

1. num: 帧数
2. filename: 动画图片文件名。
3. x: 动画左上角的 x 坐标，单位是屏幕宽度的百分比
4. y: 动画左上角的 y 坐标，单位是屏幕高度的百分比
5. interval: 每帧间隔时间，单位是毫秒。
6. isloop: 是否循环播放，0 为不循环，1 为循环。

17.4 例：

```
#anime_on 3,rain,0,0,300,1
```

18. #anime_off

18.1 功能：

停止由#anime_on 开启的动画效果。

18.2 脚本指令原型：

```
#anime_off filename
```

18.3 参数说明：

1. filename: 动画图片文件名。

18.4 例：

```
#anime_off rain
```

19. #chara_anime

19.1 功能:

自定义立绘震动效果。此语句可以指定一系列偏移量（单位为像素）的序列，并可以指定偏移的时间间隔和整体循环次数，利用此语句可以实现各种形式的立绘震动。

19.2 脚本指令原型:

```
#chara_anime charaID, period, loop_num, offset_x1, offset_y1, offset_x2, offset_y2,.....
```

19.3 参数说明:

charaID: 立绘 ID, 只能有 1 个

period: 每步偏移延迟的时间, 单位为毫秒

loop_num: 动作循环次数

offset_x1, offset_y1, offset_x2, offset_y2,.....: 每步的偏移量, 步数不限, 但最后的偏移量最好为(0,0)也就是回到原点。

19.4 例:

```
#chara_anime 1, 100, 2, 0,7, 0,16, 0,12, 0,16, 0,7, 0,0
```

三、 变量、选择、跳转类指令

系统变量说明: 本系统支持任意多个变量, 变量需为整数。变量名必须为字母开头, 区分大小写。

原则上变量名可以任意选取, 但是以下有一些特殊的变量名是系统固定使用的:

FSEL: 存储#sel 和#select_text 的选择结果

FMONTH: 游戏中的月份, 可以使用#date 显示在屏幕上, 也会显示在存档中。

FDATE: 游戏中的日期, 可以使用#date 显示在屏幕上, 也会显示在存档中。

以字母 S 开头的变量: 全局变量, 这种变量保存在 golbal.sav 中, 在游戏开始时就载入内存, 并且不受读档的影响, 可用于保存通关记录, 或者用于游戏开始界面的一些变化花样

1. #set

1.1 功能:

变量赋值。

1.2 脚本指令原型:

```
#set var_name, var_value
```

1.3 参数说明:

1. var_name: 变量名

2. var_value: 变量值或者另一个变量

1.4 例:

```
#set F58,1
```

```
#set F58,FSEL
```

2. #add

2.1 功能:

往变量上加一个数或者一个变量。

2.2 脚本指令原型:

```
#add var_name, add_value
```

2.3 参数说明:

1. var_name: 变量名, 如果此变量之前从未被赋值, 系统会先为它赋值为 0, 然后再加 add_value

2. add_value: 加上的数, 可以是数字或者变量名。如果加上的变量之前从未被赋值, 系统会不执行加法。

2.4 例:

```
#add F11,1  
#add F11,S1
```

3. #sub

3.1 功能:

从变量上减一个数或者一个变量。

3.2 脚本指令原型:

```
#sub var_name, sub_value
```

3.3 参数说明:

1. var_name: 变量名, 如果此变量之前从未被赋值, 系统会先为它赋值为 0, 然后再减 sub_value

2. sub_value: 减去的数, 可以是数字或者变量名。如果减去的变量之前从未被赋值, 系统会不执行减法。

3.4 例:

```
#sub F11,1  
#sub F11,S1
```

4. #label

4.1 功能:

行标签, 用于指定跳转的目标行

4.2 脚本指令原型:

```
#label label_name
```

4.3 参数说明:

1. label_name: 标签名, 可任意选取, 必须以英文字母开头, 区分大小写。注意同一脚本里不要有重名标签, 不同脚本里则没有关系。

4.4 例:

```
#label SEL_LABEL_1
```

5. #goto

5.1 功能:

跳转到当前脚本里指定的行标签。

5.2 脚本指令原型:

```
#goto label_name
```

5.3 参数说明:

1. **label_name**: 标签名, 必须是当前脚本文件里有的标签。系统会从脚本的当前位置往下搜寻, 到文件尾跳转到文件头往下搜寻到当前位置为止。如果没有找到标签会报错退出。

5.4 例:

```
#goto SEL_LABEL_1
```

6. #if...goto

6.1 功能:

条件执行体。条件成立则跳转到指定行。除了跳转以外不支持别的指令。

6.2 脚本指令原型:

```
#if condition,goto label_name
```

6.3 参数说明:

1. **condition**: 判断表达式, 支持判断类型: =(相等), >(大于), <(小于), >=(大于等于), <=(小于等于), !=(不等于); 左操作数必须为变量, 右操作数支持变量; 如果左操作数未被赋值过, 系统将其看作 0 来比较。如果右操作数为变量且变量未被赋值过, 系统会忽略这条 if 语句。

6.4 例:

```
#if F11=0, goto SEL_LABEL_0  
#if F11>=S1, goto SEL_LABEL_1
```

7. #change

7.1 功能:

不带返回的脚本文件跳转。直接更换脚本文件

7.2 脚本指令原型:

```
#change filename
```

7.3 参数说明:

1. **filename**: 脚本文件名, 不加扩展名

7.4 例:

```
#change script_01
```

8. #call

8.1 功能:

带返回的脚本文件跳转。目标脚本执行完毕后返回原脚本文件继续执行

8.2 脚本指令原型:

```
#call filename
```

8.3 参数说明:

1. **filename**: 脚本文件名, 不加扩展名

8.4 例:


```
#call script_01
```

9. #ret

9.1 功能:

如果是被#call调用的脚本文件，末尾一定要加上此结束标志，表示返回原脚本文件。否则运行到脚本末尾将退出游戏。

9.2 脚本指令原型:

```
#ret
```

9.3 参数说明:

无

9.4 例:

```
#ret
```

10. #sel

10.1 功能:

出现选择肢进行选择，结果存储到 FSEL 变量中。如果用户选择了第一项，FSEL=0，选择第二项，FSEL=1，以此类推

和#select_text的区别是，#sel多用在游戏主体中，支持选择中存档、已选项变灰和动画提示。这是 pymo 命令中唯一一个分为多行写的语句。

10.2 脚本指令原型:

```
#sel choice_num[,hint_pic]
```

```
choice_text_1
```

```
choice_text_2
```

```
...
```

10.3 参数说明:

1. choice_num: 选项个数

2. hint_pic: 选项提示图片的文件名，不写扩展名。图片应该放在 system 文件夹下。省略此参数表示无提示

3. choice_text_1、choice_text_2……: 选项文本。

10.4 例:

```
#sel 2
```

```
去放焰火吧
```

```
到我家里去坐坐吧？
```

11. #select_text

11.1 功能:

出现选择菜单进行选择，结果存储到 FSEL 变量中。如果用户选择了第一项，FSEL=0，选择第二项，FSEL=1，以此类推

和#sel的区别是，#select_text多用在游戏开始的菜单中，不会加载选项框背景，支持自定义文本颜色、位置。文本将在指定的矩形范围内居中显示

11.2 脚本指令原型:

```
#select_text choice_num,choice_text_1, choice_text_2,...,x1,y1,x2,y2,color,  
init_position
```

11.3 参数说明:

1. choice_num: 选项个数
2. choice_text_1、choice_text_2……: 选项文本。
3. x1: 显示范围的左上角顶点的 x 坐标, 单位是屏幕宽度的百分比
4. y1: 显示范围的左上角顶点的 y 坐标, 单位是屏幕高度的百分比
5. x2: 显示范围的右下角顶点的 x 坐标, 单位是屏幕宽度的百分比
6. y2: 显示范围的右下角顶点的 y 坐标, 单位是屏幕高度的百分比
7. color: 文字的颜色
8. init_position: 菜单初始被选中的条目, 默认设为 0 即可。

11.4 例:

```
#select_text 3,开始游戏,继续游戏,退出, 0,50,100,100,#409900, 0
```

12. #select_var

12.1 功能:

由变量控制的选择菜单, 当相应选项的变量为 1 时才会显示出来。结果存储到 FSEL 变量中。

注意: 即使在部分选项没有显示出来的情况下, FSEL 的值仍然是按照整体选项的标号进行赋值。

12.2 脚本指令原型:

```
#select_var choice_num, choice_text_1, var1, choice_text_2,  
var2, ... ,x1 ,y1 ,x2 ,y2 , color, init_position
```

12.3 参数说明:

1. choice_num: 选项个数
2. choice_text_1、choice_text_2……: 选项文本。
3. var1、var2……: 控制选项是否显示的变量或常数, 可以为常数 0、1 或变量名, 当变量为 0 时不显示, 其余情况显示。
4. x1: 显示范围的左上角顶点的 x 坐标, 单位是屏幕宽度的百分比
5. y1: 显示范围的左上角顶点的 y 坐标, 单位是屏幕高度的百分比
6. x2: 显示范围的右下角顶点的 x 坐标, 单位是屏幕宽度的百分比
7. y2: 显示范围的右下角顶点的 y 坐标, 单位是屏幕高度的百分比
8. color: 文字的颜色
9. init_position: 菜单初始被选中的条目。若设为-1, “已选项变灰”和“选项时存档”功能将可用。因此, 请在游戏开始菜单中将此项设为 0, 而在游戏过程中此项应设为-1。

12.4 例:

```
# select_var 3,去教室,var0,去保健室,var1,退出,1, 0,50,100,100,#409900, 0
```

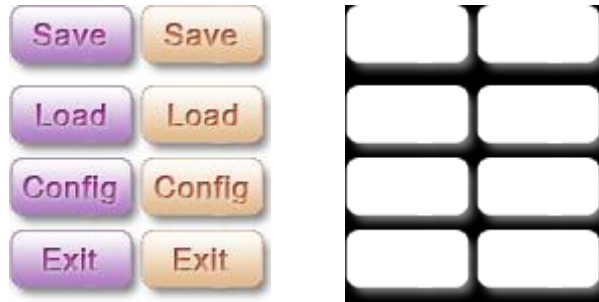
13. #select_img

13.1 功能:

出现图形选择菜单, 当相应选项的变量为 1 时才会显示出来。结果存储到 FSEL 变量中。

注意：即使在部分选项没有显示出来的情况下，FSEL 的值仍然是按照整体选项的标号进行赋值。

所有的选项都集中在一张图片上，选项图片及其遮罩要放置在 system 目录下，格式为 png。布局如下：图片分左右两列，左边为未选中状态的图片，右边为选中状态图片，从上到下分别是选项 1、2、3……pymo 会按照 choice_num 的值平均分割这张图片，这也意味着每个选项图片的大小必须一样。



13.2 脚本指令原型：

```
#select_img choice_num, filename, x1, y1, var1, x2 ,y2 , var2, ... ,  
init_position
```

13.3 参数说明：

1. choice_num: 选项个数
2. filename: 选项图片文件名。
3. x1、x2……: 相应选项图片的中心点的 x 坐标，单位是屏幕宽度的百分比
4. y1、y2……: 相应选项图片的中心点的 y 坐标，单位是屏幕高度的百分比
5. var1、var2……: 控制选项是否显示的变量或常数，可以为常数 0、1 或变量名，当变量为 0 时不显示，其余情况显示。

6. init_position: 菜单初始被选中的条目。若设为-1，“选项时存档”功能将可用。因此，请在游戏开始菜单中将此项设为 0，而在游戏过程中此项应设为-1。

13.4 例：

```
#select_img 4,button,50,40,var0,50,50,var1,50,60,var2,50,70,var3,0
```

14. #select_imgs

14.1 功能：

出现图形选择菜单，当相应选项的变量为 1 时才会显示出来。结果存储到 FSEL 变量中。

注意：即使在部分选项没有显示出来的情况下，FSEL 的值仍然是按照整体选项的标号进行赋值。

每个选项使用单独的图片，选项图片及其遮罩要放置在 system 目录下，格式为 png。图片分左右两列，左边为未选中状态的图片，右边为选中状态图片。

14.2 脚本指令原型：

```
#select_imgs choice_num, filename1, x1, y1, var1, filename2, x2 ,y2 , var2, ... ,  
init_position
```

14.3 参数说明：

1. choice_num: 选项个数

2. filename1、filename2……：选项图片文件名。
3. x1、x2……：相应选项图片的中心点的 x 坐标，单位是屏幕宽度的百分比
4. y1、y2……：相应选项图片的中心点的 y 坐标，单位是屏幕高度的百分比
5. var1、var2……：控制选项是否显示的变量或常数，可以为常数 0、1 或变量名，当变量为 0 时不显示，其余情况显示。

6. init_position: 菜单初始被选中的条目。若设为-1,“选项时存档”功能将可用。因此,请在游戏开始菜单中将此项设为 0,而在游戏过程中此项应设为-1。

14.4 例:

```
#select_imgs
4,button0,50,40,var0,button1,50,50,var1,button2,50,60,var2,button3,50,70,var3,0
```

15. #wait

15.1 功能:

等待指定时间。

15.2 脚本指令原型:

```
#wait time
```

15.3 参数说明:

1. time: 等待的时间 (单位: 毫秒)

15.4 例:

```
#wait 2000
```

16. #wait_se

16.1 功能:

等待 se 播放结束。按确定键可中断等待。

16.2 脚本指令原型:

```
#wait_se
```

16.3 参数说明:

16.4 例:

```
#wait_se
```

16. #rand

16.1 功能:

生成一个指定范围内的随机整数并把它赋给变量。

16.2 脚本指令原型:

```
#rand var_name, min_value, max_value
```

16.3 参数说明:

1. var_name: 变量名

2. min_value: 随机数范围的最小值,必须是非负整数

3. max_value: 随机数范围的最大值,必须是非负整数且大于 min_value

16.4 例:

```
#rand F11,0,3
```

四、 声音类指令

1. #bgm

1.1 功能:

播放背景音乐。如果当前已经有正在播放的背景音乐，系统会先停止正在播放的音乐，再播放此音乐。

1.2 脚本指令原型:

```
#bgm filename[,isloop]
```

1.3 参数说明:

1. filename: 音频文件名，不加扩展名
2. isloop: 是否循环播放，0 为不循环，1 为循环。此参数省略默认为循环

1.4 例:

```
#bgm BGM001  
#bgm BGM002,1
```

2. #bgm_stop

2.1 功能:

停止播放当前背景音乐。如果当前没有正在播放的背景音乐，系统会忽略此命令。

2.2 脚本指令原型:

```
#bgm_stop
```

2.3 参数说明:

无

2.4 例:

```
#bgm_stop
```

3. #se

3.1 功能:

播放音效。如果当前已经有正在播放的音效，系统会先停止正在播放的音效，再播放此音效。

3.2 脚本指令原型:

```
#se filename[,isloop]
```

3.3 参数说明:

1. filename: 音频文件名，不加扩展名
2. isloop: 是否循环播放，0 为不循环，1 为循环。此参数省略默认为不循环

3.4 例:

```
#se SE001  
#se SE002,1
```

4. #se_stop

4.1 功能:

停止播放音效。如果当前没有正在播放的音效，系统会忽略此命令。

4.2 脚本指令原型:

```
#se_stop
```

4.3 参数说明:

4.4 例:

```
#se_stop
```

5. #vo

5.1 功能:

播放人物语音。如果当前已经有正在播放的语音，系统会先停止正在播放的语音，再播放此语音。

5.2 脚本指令原型:

```
#vo filename
```

5.3 参数说明:

1. filename: 音频文件名，不加扩展名

5.4 例:

```
#vo AY00001
```

五、 系统类指令

1. #load

1.1 功能:

进入读档界面。如果用户在读档界面没有选择任何档而是退出，系统继续执行下一条指令

1.2 脚本指令原型:

```
#load [save_num]
```

1.3 参数说明:

save_num: 存档号。如果不指定，那么调出读档界面让用户选择。如果指定此项，那么不调出读档界面而是直接读指定的存档。如果此存档不存在则继续执行下一条指令

1.4 例:

```
#load
```

```
#load 0
```

2. #album

2.1 功能:

进入 CG 鉴赏系统。如果用户退出鉴赏系统，系统继续执行下一条指令
`#album album0`

2.2 脚本指令原型:

```
#album [album_list_filename]
```

2.3 参数说明:

album_list_filename: 列表文件名，格式见第九章。

2.4 例:

```
#album
```

默认载入的列表文件名为 `album_list.txt`，对应的背景文件名为 `albumbg_0.png`, `albumbg_1.png`, `albumbg_2.png`... 用于生成背景的底图为 `albumbg.png`。

```
#album album0
```

表示要载入的列表文件名为 `album0.txt`，对应的背景文件名为 `album0_0.png`, `album0_1.png`, `album0_2.png`... 用于生成背景的底图为 `album0.png`。

3. #music

3.1 功能:

进入音乐鉴赏系统。如果用户退出鉴赏系统，系统继续执行下一条指令

3.2 脚本指令原型:

```
#music
```

3.3 参数说明:

无

3.4 例:

```
#music
```

4. #date

4.1 功能:

显示游戏日期，游戏日期由变量 `FMONTH` 和 `FDATE` 指定。

4.2 脚本指令原型:

```
#date date_bg,x,y,color
```

4.3 参数说明:

1. **date_bg:** 背景图片的文件名，不写扩展名。图片应该是 `png` 格式，放在 `system` 文件夹下。省略此参数表示无提示

2. **x:** 日期文字的左上角顶点的 `x` 坐标，单位是屏幕宽度的百分比

3. **y:** 日期文字的左上角顶点的 `y` 坐标，单位是屏幕高度的百分比

4. **color:** 文字的颜色

4.4 例:

```
#date EYE_D,65,42,#000000
```

5. #config

5.1 功能:

显示游戏设置对话框。

5.2 脚本指令原型:

#config

5.3 参数说明:

5.4 例:

#config