

THE NVL MAKER

新手教程

(六)

地图与养成面板的制
作

一、必备的基础知识

因为这是 THE NVL Maker，而不是吉里吉里/KAG 的教程，所以之前的新手教程里没有提供多少吉里吉里/KAG 基础知识。而且凡是讲到基础知识的部分，即使看不懂，也都可以无视……反正只要理解了什么是标签和跳转，再加上会制作选择按钮，自制一个多路线的电子小说就完全没有问题了。

不过如果想要深入研究的话，有些东西还是必须要说的。

很多图形化游戏制作软件都喜欢用“不需要程序知识”之类的做宣传。

如果说程序知识指的是对 C#或者 JAVA 诸如此类的代码有了解，那当然是不需要了。

但是，对游戏制作者来说，最重要的从来都不是对代码知识的了解，而是“逻辑”。

你需要严密地考虑当游戏进行到一个步骤，玩家一共有多少种选择，这每种选择又会导致怎样的下一步。

这点来说，一定的程序知识还是需要的。如果能尝试理解以下的概念的话……

(0) “层” 和 “页”

KAG，或者说吉里吉里的画面，是全部由所谓的「层」叠起来组成的（和 Photoshop 等软件的层差不多）。

KAG 的每一层又可以分成两部分，那就是被叫做表页 **fore**、里页 **back** 的东西。所有表页的内容组成了画面上显示的东西，而里页的内容，画面上是看不到的。里页主要是在使用 [trans]（画面切换效果）时，用于预载想要显示的内容的。（转自 KAG 官方文档汉化版）
层分成“消息层”和“图片层”两种。前者用于显示文字、文字连接[link]和按钮，后者用于显示背景图片、角色图片、界面底板图片等等。

(1) 脚本&指令

用一行一行的指令（TAG）来描述游戏内容的文本文档。游戏开始时，会首先执行名叫 **first.ks** 的脚本，然后再根据 **first.ks** 里的指令，“跳转”或者“呼叫”其他的脚本，以让游戏继续进行。

而指令则是以@开头，或者是由括号[]包裹起来的内容。每一个指令都让游戏做一件事。例如，显示一张图片，播放一段音乐，显示一个按钮，等等。

(2) 变量（变数）

记录了数字、字符串等内容。可以对它进行操作、显示它记录的值，或者根据它记录的值，来判断接下来应该执行什么指令（条件分歧）。

（3）跳转&标签

跳转，中断一个脚本段落的执行，跳到另外一个段落继续。（不会返回原来的脚本）。

标签：脚本中的一个“书签”，可作为跳转到的位置标记。

效果：A->B

（4）呼叫

在脚本 A 里，假如执行了呼叫指令[call storage="脚本 B.ks"]，那么就会中断脚本 A，而继续执行脚本 B，一直到脚本 B 执行到返回指令[return]。

遇到[return]以后，将会停止 B 的执行，返回 A，从刚刚 A 被中断的地方开始接着往下走。

效果：A->B->A

（5）条件分歧

用[if]和[endif]包裹起来的一段指令，只有当满足某个条件（例如某个变量大于或者小于 10）的时候才会被执行，不满足条件的时候会被忽略。

（6）等待玩家选择

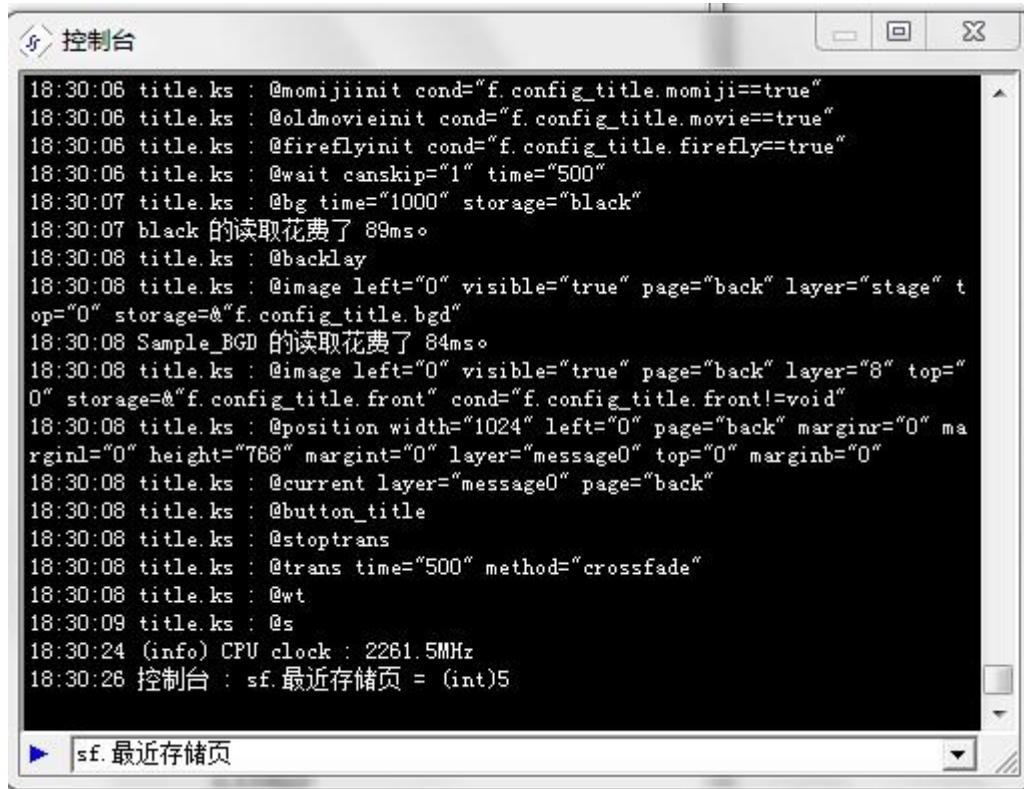
中断脚本的执行，一直到玩家点下选项按钮、按钮、鼠标左键、右键等等。

中断脚本时请一定保证画面上有玩家可以操作的东西，否则的话游戏就会卡住。

（7）调试窗口

点下游戏窗口上的 调试->控制台，可以打开的黑色窗口，会显示出游戏当前的执行状况。也就是“脚本走到了什么地方”。

你可以试着在下面输入变量的名字等，它会告诉你变量现在的值。（你也可以对这个变量进行操作……相当于使用了游戏修改器调整游戏数值 [www](#)）



```
18:30:06 title.ks : @momijiinit cond="f.config_title.momiji==true"
18:30:06 title.ks : @oldmovieinit cond="f.config_title.movie==true"
18:30:06 title.ks : @fireflyinit cond="f.config_title.firefly==true"
18:30:06 title.ks : @wait canskip="1" time="500"
18:30:07 title.ks : @bg time="1000" storage="black"
18:30:07 black 的读取花费了 89ms。
18:30:08 title.ks : @backlay
18:30:08 title.ks : @image left="0" visible="true" page="back" layer="stage" top="0" storage="@f.config_title.bgd"
18:30:08 Sample_BGD 的读取花费了 84ms。
18:30:08 title.ks : @image left="0" visible="true" page="back" layer="8" top="0" storage="@f.config_title.front" cond="f.config_title.front!=void"
18:30:08 title.ks : @position width="1024" left="0" page="back" marginr="0" marginl="0" height="768" margint="0" layer="message0" top="0" marginb="0"
18:30:08 title.ks : @current layer="message0" page="back"
18:30:08 title.ks : @button_title
18:30:08 title.ks : @stoptrans
18:30:08 title.ks : @trans time="500" method="crossfade"
18:30:08 title.ks : @wt
18:30:09 title.ks : @s
18:30:24 (info) CPU clock : 2261.5MHz
18:30:26 控制台 : sf.最近存储页 = (int)5
```

(8) 宏 (macro)

由多个指令简化而成的一行指令。每当使用时，则相当于执行了这多个指令的效果。
使用宏时需要先使用[macro][endmacro]定义并注册。
注册的方法是让系统载入（通常是呼叫）包含有宏定义的脚本。

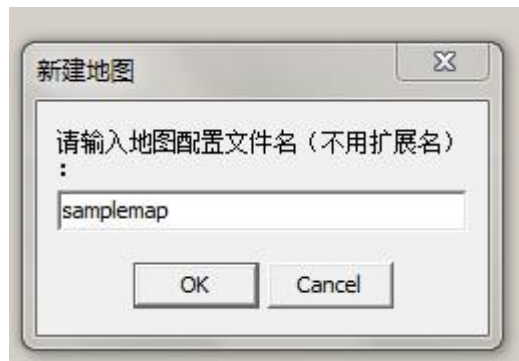
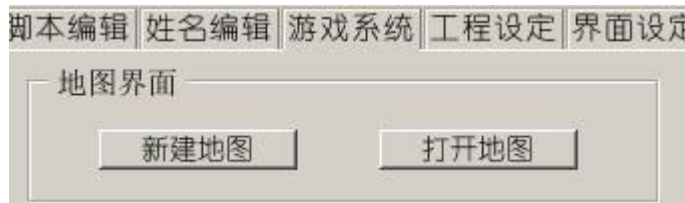
(9) 插件 (plug-in)

由 TJS 语言写成的辅助脚本，通常也包含宏。用于扩展吉里吉里的功能。
使用对应插件的话，即使完全没有 TJS 知识，也可以达成想要的效果。
例如游戏里的樱花、红叶、萤火虫、旧电影等等画面效果。
使用插件之前同样需要让系统载入插件脚本。

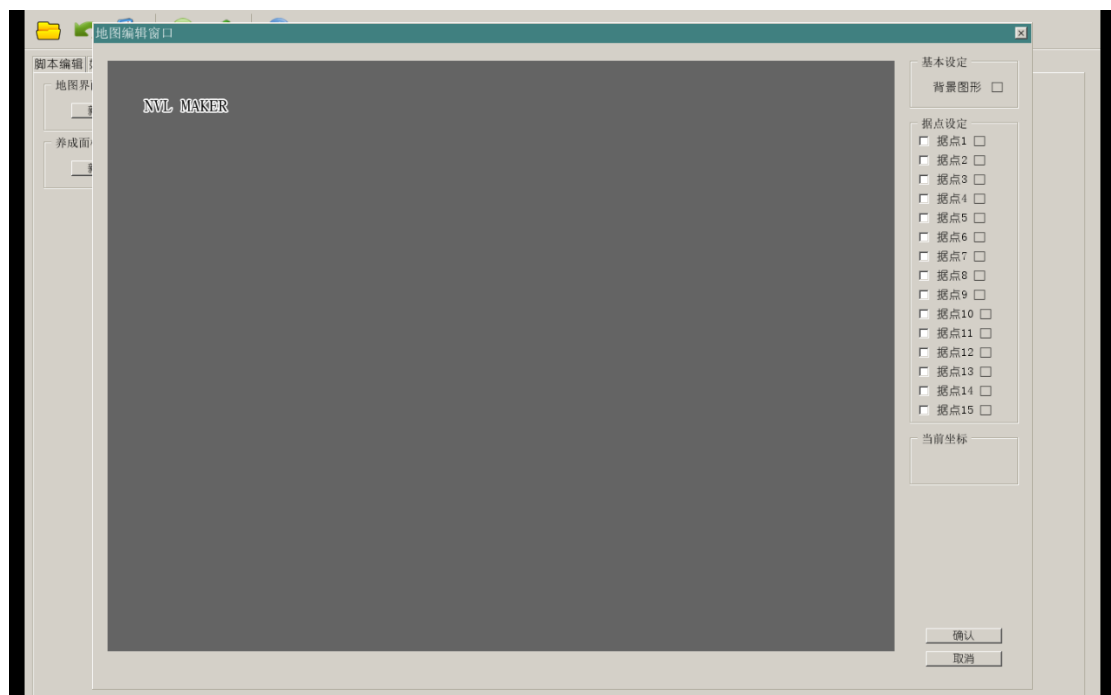
二、选择的扩展——地图

无论是选择还是地图，根本都是一个东西，那就是“按钮”。
在一个界面上显示很多个按钮，等待玩家选择，并且在选择之后跳转到不同的脚本继续剧情。
地图编辑器的操作和其他界面编辑器是类似的。不管怎样，先来试着制作一下吧？

（1）新建地图



（2）编辑地图



打开新建的地图，右边是据点设定。所谓“据点”就是地图上一个可以去的地方。目前最多可以设定 15 个。（通常来说，据点的数量太多，会导致地图画面很混乱，所以不推荐设置太多。）

勾选“据点 1”，画面上显示了一个按钮。接着点“据点 1”右边的□，开始设定内容。

地图据点设定

显示设定

地名 据点1

条件 ☐

按钮图形

x 50

y 100

一般 sample_off ☐

选中 sample_on ☐ ☐

按下 sample_on ☐ ☐

执行操作

剧本 ☐

标签

表达式

确认

取消

这里要注意的是地名，修改这里并不会影响游戏的内容，但是会被显示在地图编辑器侧边栏上，可以方便管理。

至于“条件”，则是这个地图什么时候可以去的设置。和选择按钮一样，你可以设定只有在特定条件下，这个地图据点才开放。

所以，在游戏里，要显示一个据点，有两个条件，第一是在侧边栏上勾选了这个据点，第二是满足了这个触发条件。

当然，条件留空的话，就是任何情况下都会显示。

按钮图形和其他界面按钮是一样的，可以设定位置，一般选中按下的图像等。

剧本、标签等等就不用说了，和选择按钮是完全一样的。

(3) 调用地图

编辑保存地图后，就可以在游戏里使用到了。

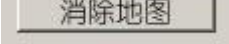
脚本编辑器里，调用地图的指令是这个：





打开文件选择，选择刚刚制作的那张地图。然后点确认。

另外，和选项一样，当玩家选择完毕，跳转到某个脚本段落继续的时候，请记得先清理画面。

和清除选项一样，这里使用的是 。

同样是每个跳转到的地方，都要加上这句哟。

养成按钮设定

x

50

y

100

一般

sample_off

☐

选中

sample_on

☐ ☐

按下

sample_on

☐ ☐

名称

按钮1

剧本

☐

标签

条件

☐

执行

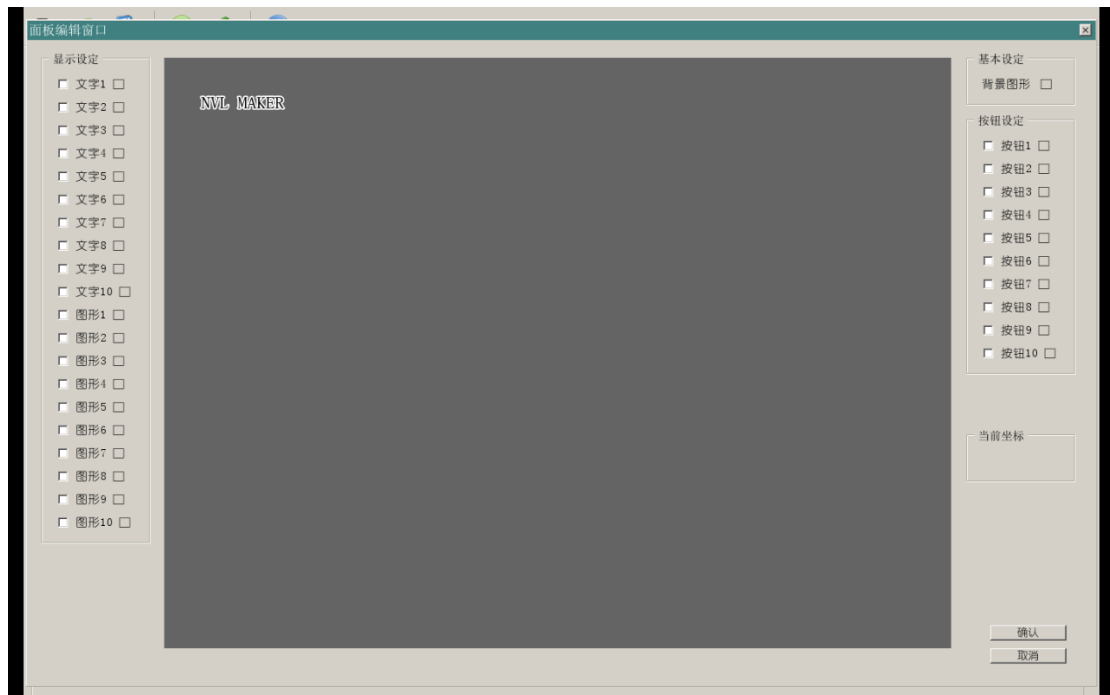
确认

取消

（最下方的“执行”就相当于选择按钮里的“执行-TJS”栏，可以填入变数操作。）

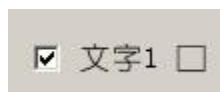
三、养成面板

(1) 文字、图形、按钮



和新建地图一样，新建一个养成面板，然后打开。可以看到界面复杂了一些。左边是显示“文字”和“图形”的地方，右边则是“按钮”。右边的按钮也可以当做地图的据点来使用。而左边的文字、图形可以用来在面板上显示数据。假如你需要做一个带有“几月几日”日期框的地图，那么就可以用养成面板来实现。

(2) 显示变数的内容



“文字”部分的使用和其他部件一样，勾选就能显示在面板上。□开始设定。

养成面板文字设定

显示设定

名称 文字1

条件 ☐

变数 f. 好感度

文字样式

x 64

y 129

字体 ☐

字号 20

颜色 0xFFFFFFFF ☐

☐ 加粗

☐ 阴影 ☐

☐ 边缘 ☐

确认

取消

“文字一”同样只是名称，修改之后显示在编辑器上，方便管理。
这里新鲜的应该就只有“变数”一项了。
没有错，在这里直接填入变数的名字，当调用养成面板的时候，这里就会显示出变数的内容。

例如我们可以把之前操作的那个变数显示在这里……
那么当调用养成面板的时候就可以看到这样的东西了：

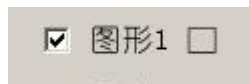


如果变量的内容是字符串，那么就会显示出字符串来。



(3) 显示数字图片

之前在存取系统里调用的那个显示数字图片的宏，其实就是在这里用的。相比单纯的显示数字，有的时候更喜欢稍微华丽一点的效果吧。也就是用 0-9 的个位数字图片，来组成多位的数字。这时候就要用到左边栏的“图形”了。





养成面板图形设定

显示设定

名称

条件 ☐

变数

坐标设定

x

y

变数值作为图形数字

前缀

间距

变数值作为图片名

演示图形 ☐

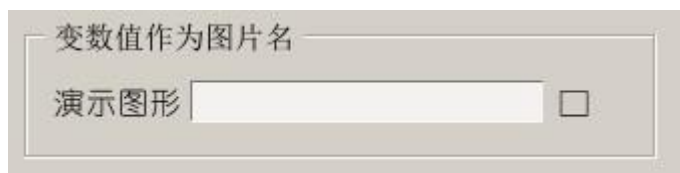
确认

取消

填写变数，则变数的值会被当做数字显示。
而数字图片的外观，是由数字图片前缀决定的，代表 0-9 数字图片前面的部分。
例如你可以把一套 0-9 的数字图片命名为 abc0~abc9，那么这里就应该填写 abc。

（4）根据变数的值（字符串）显示图片

除此之外，还有另外一个功能，



变数值作为图片名

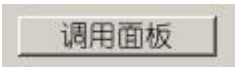
演示图形 ☐

你可以指定在这里显示固定的一张图片作为演示图片。
在指定了演示图片以后，数字图片功能将会失效，改为第二个功能。
根据变数的值显示图片。

例如你的变数里保存的不是“1~7”，而是“星期一~星期日”的字符串。
那么准备名叫“星期一~星期日”的7张图片，
并且在游戏里根据日期操作变数的值，那么调用养成面板的时候，就会读取这个字符串，然后把字符串当做图片名，显示出来。（找不到叫这个名字的图片的话，当然会报错。）

（5）调用养成面板

在脚本里调用养成面板和调用地图的方式几乎完全一样。

使用： 来进行。



注意要做地图的时候，这里的“等待玩家选择”需要勾上。
这样的话游戏才会停留在这里，等待玩家点选按钮。
否则的话，游戏就会继续前进了。

（6）不等待的话？

假如不等待玩家输入，游戏就会继续进行了。
为什么要为养成面板设定这样的功能呢？
其实也很简单。

☒ 等待玩家选择

养成游戏里经常用到的设定是一周只需要输入一次日程。这样只有周一是需要“等待玩家安排”的，其他时候都只是刷一下画面，显示一下数值的增减。这时候，养成面板就成为一个“记分牌”一样的存在了。

当然要注意，既然不需要等待，这个面板就没有必要设定按钮。你可以制作“输入日程专用”和“显示专用”的两张面板，根据条件轮番使用。

四、养成系统制作浅谈

（1）没有那么简单啦！

如果到制作养成系统还能够说“完全不研究吉里吉里/KAG”就可以搞定。那是……不可能的！死心吧！好在这只是新手教程，也没有要负责教你怎么打造一个完整的养成系统。不过，可以大概讲一下诀窍。

（2）不变的日常——游戏主循环

对一般的电子小说来说，是没有循环这种概念的，因为它的每一步都是确定发生的，从 A→B→C，并不会再返回到 A。但是养成系统则不同，它通常是以一周为单位，随着日期的前进，根据今天是周几，调用不同的面板。很可能整个游戏都是这样的，周一，安排日程，周二到周五，养成，周六日，外出。那么这整个游戏的主循环很可能就是一个简单的“一天”脚本。每天要做的事情就是根据条件分歧调用面板，以及操作时间变数，再跳回开始，继续下一天。

（3）地图事件

因此养成的地图，也和其他地图 AVG 的形式不太一样，通常一个地图据点下面会有很多“可能触发的事件”随机发生，而不是点下固定据点就一定会跳到那个事件去。

如果你连续 10 天每天都去保健室，每天都发现同一个粉红色头发的少女微笑着对你打招呼并自我介绍，要嘛你是鬼打墙了，要嘛你是黑长直（喂）。

所以当制作地图等面板时，跳跃到的应该也不是固定的事件，而是一个新的判断脚本。在这个脚本里，根据各种各样的变数值，来判断现在哪一个事件可以被触发。

这种时候当然是要用到“条件分歧”了。

例如，在某个地图据点下，有 10 个可能发生的事件。1-10。

那么这个判断脚本可能是这样的：

```
[if exp=条件]
呼叫事件 1
[elsif exp=条件]
呼叫事件 2
[elsif exp=条件]
呼叫事件 3
.....
[else]
呼叫事件 10
[endif]
[jump storage=主循环]
```

而被呼叫的每个事件脚本，都用[return]结尾。这样当事件执行完毕之后，就会返回判断脚本，再从判断脚本跳到主循环。不必在意这个事件具体是什么时候触发的，或者触发完毕以后接着干什么。

让每个据点都对应一个判断脚本，管理事件会方便一些。

（4）主线事件

主线事件，养成游戏里常见的“满足一定条件之后”必定触发的事件。同样也可以用和管理地图事件一样的方法来管理。只不过这部分的内容应该插入到主循环内。

例如这样的一天：

```
*主循环
【判断早上可能发生什么事件】->
根据今天是周几调用养成面板或者地图->
数值变化->
【判断下午可能发生什么事件】->
一天结束->
[jump target=*主循环]
```

而【判断可能发生什么事件】自然就是跳到一个主线事件判断脚本了。

（5）避免事件重复发生

假如一个事件已经发生过了，不希望它反复发生，那么怎么从事件判断脚本里把这个事件去掉呢？答案当然是“用变量”。

每个事件都应该有一个属于自己的变量。当这个事件执行的时候，在这个事件的脚本里，把变量设为1。

在事件判断的时候，要触发这个事件，必须满足的条件要包括“对应变量==0”。这样的话，就可以制作出不重复发生的事件了。

除了用一般的变量以外，为了方便管理，“数组”和“字典”也是不错的选择。至于具体的就请去看吉里吉里和 KAG 的说明文档了……

（6）事件的优先级

刚刚展示的事件判断脚本（伪）是这样的结构。假如事件 1 触发了的话，那么以后的事件（即使满足条件）也都不会触发了，一直到下次调用这个事件判断脚本。所以希望先发生的事件，应该写在比较前面的位置。

而在最后[else]以后的，则是优先级最低的事件，通常是在没有事件的时候，用来凑数的。例如“去 XX 地方逛了逛，什么也没发生，又回来了。”

假如每次只触发一个事件的话，就可以用这样的事件判断脚本。
但假如你希望的是一旦进行判断，就让所有满足条件的事件都触发的话。
就把事件判断脚本写成这样的形式吧。

```
[if exp=条件]
呼叫事件 1
[endif]

[if exp=条件]
呼叫事件 2
[endif]

[if exp=条件]
呼叫事件 3
[endif]

.....

[if exp=条件]
呼叫事件 10
[endif]
[jump storage=主循环]
```

（7）日期系统

本质上就是几个变数。f. 年，f. 月，f. 日，f. 星期几……等等。每天（一次主循环）给日加上 1，当满足条件的时候给月加上 1……以此类推。

不过如果觉得这段写起来麻烦的话，也有插件可以用，具体看这里：

<http://kcddp.keyfc.net/bbs/viewthread.php?tid=195>

五、后记

嗯，这个漫长的新手教程终于结束了。

在学吉里吉里的时候，总看到网上很多人喊着没有教程没有范例，不过已经存在的那么多范例和教程，认真去看的人又有几个呢？

<http://kcddp.keyfc.net/bbs/viewthread.php?tid=1188&extra=page%3D1>

做游戏，其实并不是那么难的事。

但是也没有简单到不用动脑，不用动手的程度。

不管怎样，希望你看完教程以后还愿意坚持。

打开 THE NVL Maker，开始游戏制作之旅吧……

期待着作品哟。^_^