THE NVL MAKER
新手教程
（四）
歷史紀錄與系統設定界面

一、歷史記錄界面

（1）界面編輯器

首先你一定已經知道怎麼打開這個界面，
以及如何給它換背景，換各種按鈕，換滾動條樣式。
並且還知道可以用“文字範圍”設定歷史記錄顯示在畫面上的什麼位置。
如果真的這些都不知道，咳咳，試一下肯定知道了。
不過其他的一些詳細設置，就需要用到 Config.tjs 了。
這是吉裡吉裡的一個通用設置文件，當然，NVL 裡也可以用圖形化工具來編輯它。

（2）Config.tjs

Config.tjs 與歷史記錄相關的設置如下。
在界面設定下點擊“Config.tjs”即可自動打開編輯工具。
假如你真的完全不知道它們是幹嘛的，請慎重修改。= =b

（3）腳本裡操作歷史記錄

只有在有對話，並且開啟歷史記錄的情況下，這個界面才會顯示內容。因此遊戲開始時，請在腳本中添加這個指令。

否則就不要問我為什麼打開這個界面是一片空白啦。

另外關於“允許開啟”，在普通對話框和大對話框的情況下，使用滾輪就可以自動打開歷史記錄界面，而透明全屏對話框下則禁止了這個操作。
基本上這些設定只是作者的習慣而已，並沒有什麼特別意義。
如果你覺得需要修改的話，可以修改 Data/macro/macro_play.ks。
[macro name=dia][macro name=scr][macro name=menu]三個宏裡的[history]指令，將 enabled 的值改為 true 或者 false。

二、系統設定界面

（1）界面編輯器

你一定在想我到底還貼這個東西出來幹嘛，只要看了教程怎麼可能到現在還不知道怎麼填這些東西呢哈哈哈……

嗯沒錯是這樣的，所以，接下來要講的東西就是，怎麼不通過界面編輯器，往遊戲系統裡加新的內容。

今天主要講的內容就是怎麼在系統設定界面裡加入“都設為默認值”按鈕。

（2）macro 文件夾

哎呀不要擺出“=口=”的表情，這些就算看不懂也沒關係的。
這個還是很有需求的吧，為什麼界面編輯器裡沒有呢？
……因為作者也忘了，後來就懶得加了（喂）。
所以乾脆就利用這個機會，講解一下“在你不知道的時候，THE NVL Maker 都在做什麼”吧。


<=這個按鈕你肯定戳過了對不對，沒戳過的話就趕緊去戳一下。

<=然後我們再打開這個文件夾。也就是 project/你的遊戲文件夾/Data/macro

這裡面有兩種類型的文件，一種是.ks，另外一種是.tjs。

不管哪一種都可以用最普通的文本編輯器比如記事本打開。

.tjs 部分是用 THE NVL Maker 的界面編輯器來修改的，用來記錄你的界面背景，按鈕圖片，位置，字體等等信息。就不要隨便動它了。

至於.ks，不就是之前已經見過的腳本嗎。

只不過 scenario 文件夾下的部分是劇情腳本，這裡則是系統腳本。

所有以“macro_”開頭的都是宏設置。而其他的那些就是每個界面的具體代碼了。

至於“系統設定”這個界面，就在 option.ks 裡。

（3）自己來修改 option.ks！

因為其他複雜的東西都寫在了 macro_ui 裡，所以這個腳本很短。

我相信你不會覺得這個東西看著很嚇人的。

而且其他的部分都不要在意，我們只是需要增加一個按鈕。

這時候請重點把眼光放在“;描繪各種 ABC”這行囧注釋上面。

只要在這個注釋下面加入按鈕，就可以顯示在界面上了。

（4）設定按鈕位置

假設我們把這個按鈕添加在（200,50）的位置。

那麼就在兩個“;描繪各種 ABC”下面都插入一行：

[locate x=200 y=50]

```
;-----------------------------------------------------------
;系統設定
;-----------------------------------------------------------
*start
[locksnapshot]
[tempsave]
;-----------------------------------------------------------
*window

[history enabled="false"]

[locklink]

[rclick enabled="true" jump="true" storage="option.ks" target=*返回]


[backlay]
[image layer=14 page=back storage=&"f.config_option.bgd" left=0 top=0
visible="true"]

;隱藏系統按鈕層

[hidesysbutton page="back"]

[current layer="message4" page="back"]
[layopt layer="message4" visible="true" page="back" left=0 top=0]
[er]

;描繪各種 ABC

[locate x=200 y=50]

[button_option page=back]
[trans method="crossfade" time=500]
[wt]

[s]


*刷新畫面

[current layer="message4"]
[er]

;描繪各種 ABC
```

```
[locate x=200 y=50]

[button_option page=fore]
[s]


*返回

[jump storage="main_menu.ks" target=*返回]
```

（5）加上按鈕

一般的按鈕，使用[button]指令就可以了。重要的參數主要有以下幾個：

Normal 按鈕“一般”狀態下的圖片。

Over　按鈕“選中”狀態下的圖片。

On　　按鈕“按下”狀態下的圖片。

Target 按鈕點下後，跳轉到的標簽。

因為現在沒有別的圖片，先隨便代替一下吧。

[button normal=sample_off over=sample_on]

現在兩個地方的內容應該都變成了這樣：

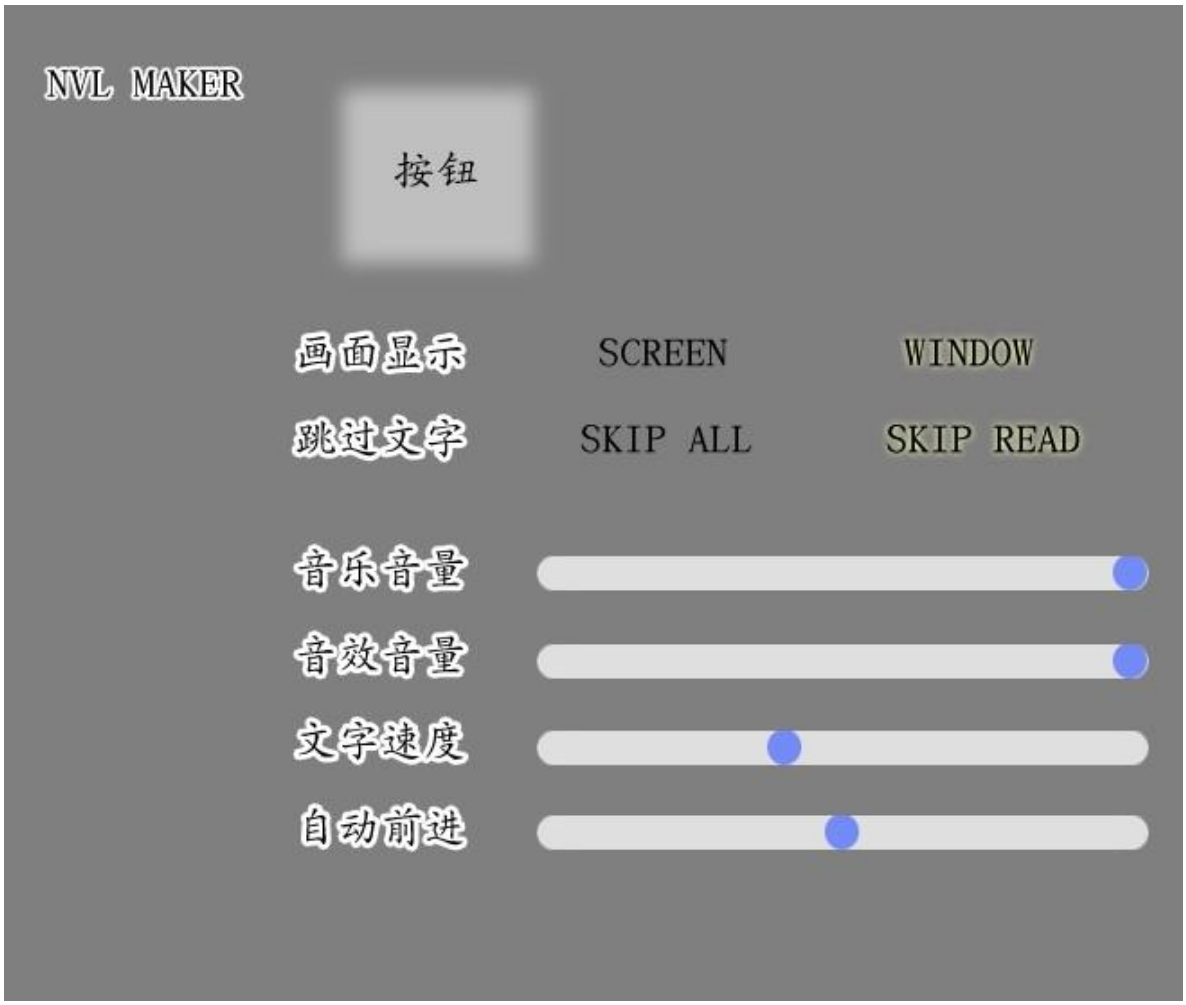
```
;描繪各種 ABC

[locate x=200 y=50]
[button normal=sample_off over=sample_on]
```

第一個地方，是系統界面剛剛打開時顯示的內容（漸變顯示），第二個地方，則是點下任何一個按鈕修改設定之後顯示的內容（瞬間刷新）。

假如少加了其中一個地方，就會導致按鈕在那個情況下不顯示。

保存，測試下遊戲。



現在系統設定界面上多了一個按鈕。

雖然你怎麼戳它都沒有反應……

（6）添加標簽

按鈕戳下去以後沒有反應，當然是因為沒有設定接下來應該跳轉到哪裡。也就是沒有“標簽”。

現在，在 option.ks 裡的最後，加入一個新的標簽，叫“*恢復默認值”好了。

不過在做完“恢復默認值”的操作以後，肯定還要跳回來“*刷新畫面”（把修改的值反映在界面上）。

所以，繼續加上新的一行。

[jump storage="option.ks" target=*刷新畫面]

同時按鈕的定義也改成這樣：

[button normal=sample_off over=sample_on target=*恢復默認值]

現在你的 option 腳本應該是這樣的：

```
;-----------------------------------------------------------
;占位用的系統設定
;-----------------------------------------------------------
*start
[locksnapshot]
[tempsave]
;-----------------------------------------------------------
*window

[history enabled="false"]

[locklink]
[rclick enabled="true" jump="true" storage="option.ks" target=*返回]


[backlay]
[image layer=14 page=back storage=&"f.config_option.bgd" left=0 top=0
visible="true"]
;隱藏系統按鈕層
[hidesysbutton page="back"]

[current layer="message4" page="back"]
[layopt layer="message4" visible="true" page="back" left=0 top=0]
[er]
;描繪各種 ABC
[locate x=200 y=50]
[button normal=sample_off over=sample_on target=*恢復默認值]


[button_option page=back]
[trans method="crossfade" time=500]
```

```
[wt]

[s]


*刷新畫面

[current layer="message4"]
[er]

;描繪各種 ABC

[locate x=200 y=50]

[button normal=sample_off over=sample_on target=*恢復默認值]


[button_option page=fore]
[s]


*返回

[jump storage="main_menu.ks" target=*返回]


*恢復默認值


[jump storage="option.ks" target=*刷新畫面]
```

（7）添加功能

加上標簽以後，依然沒有任何其他操作，因此點下按鈕以後還是和原來一樣。

接下來要做的就是在

```
*恢復默認值


[jump storage="option.ks" target=*刷新畫面]
```

的中間插入內容。

既然說是恢復默認值，那麼先來假定幾個默認值吧。

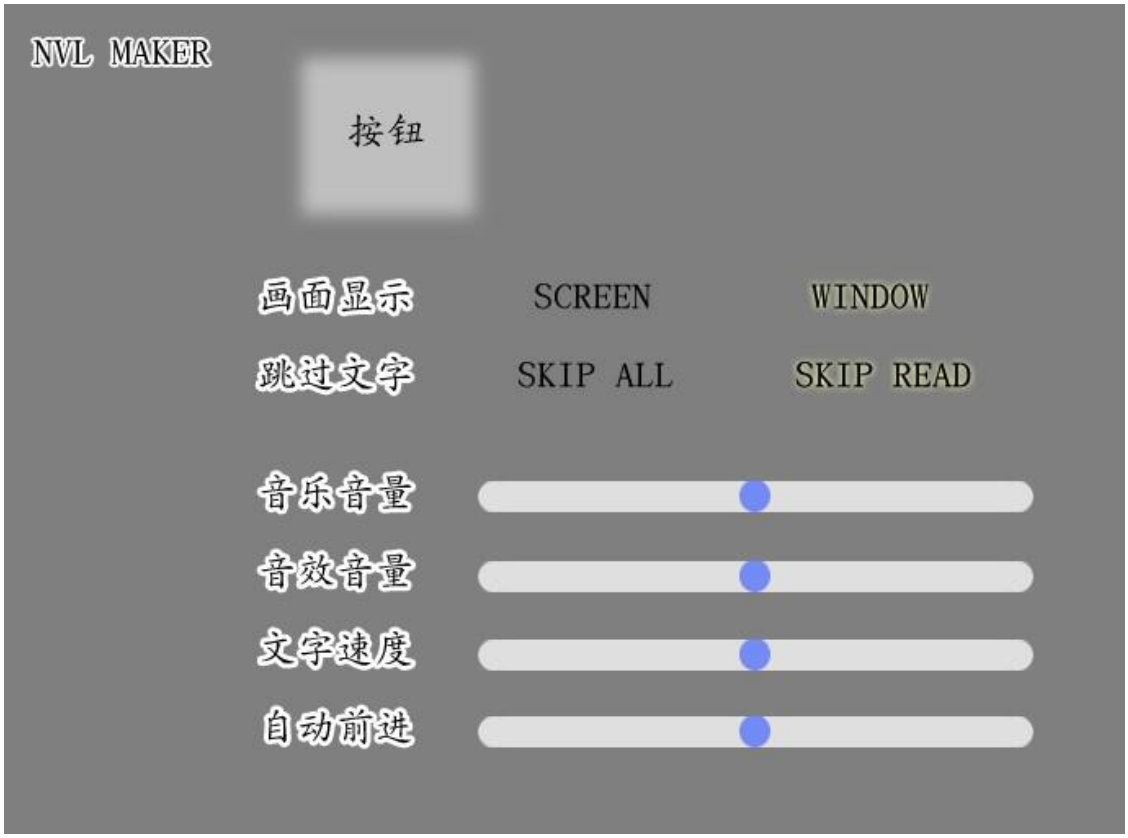
默認遊戲是窗口顯示，只跳已讀內容，並且音樂和音效音量都是 50。

文字速度和自動前進速度都在中間檔（0~10）的 5。

把這些東西加上看看：

```
*恢復默認值
[eval exp="kag.fullScreen=false" cond="kag.fullScreen"]
[eval exp="kag.allskip=false"]
[eval exp="kag.bgmvolume=50"]
[eval exp="kag.sevolume=50"]
[eval exp="kag.textspeed=5"]
[eval exp="kag.autospeed=5"]
[jump storage="option.ks" target=*刷新畫面]
```

現在，測試下遊戲。



畫面是不是變得很整齊了？www

三、擴展內容

（1）用 eval 執行 TJS 式

其實剛剛的各種[eval exp="TJS 式"]，就跟之前點下選擇按鈕時候執行的 TJS 式是一樣的，可以對變數的值進行操作。

對應腳本編輯器裡的這個指令：

（2）cond 屬性和條件分歧

[eval exp="kag.fullScreen=false" cond="kag.fullScreen"]，這裡的 cond="TJS 式"]，則相當於一個條件分歧。只有當“kag.fullScreen”的值為真時才執行。

（如果本來就是窗口，還是強制執行 kag.fullScreen=false 的話，在這個版本裡會導致畫面卡住，所以加上了這個判斷。）

也就是說，

[eval exp="kag.fullScreen=false" cond="kag.fullScreen"]

等價於下面的三行，

```
[if exp="kag.fullScreen==true"]
[eval exp="kag.fullScreen=false"]
[endif]
```

（PS：kag.fullScreen 和 kag.fullScreen==true 是一個意思。）

cond 屬性在判斷是不是要執行單條指令的時候比 if，endif 輕鬆得多。

在自己撰寫腳本的時候，請多利用。

（3）TJS 段落

不過剛剛這樣一行一行的[eval exp="TJS 式"]是不是也挺煩人的？

其實，它也可以寫成下面的樣子：

```
*恢復默認值
[iscript]
if (kag.fullScreen) kag.fullScreen=false;
kag.allskip=false;
kag.bgmvolume=50;
kag.sevolume=50;
kag.textspeed=5;
kag.autospeed=5;
[endscript]
[jump storage="option.ks" target=*刷新畫面]
```

[iscript]和[endscript]標起來的段落，是書寫大段 TJS 代碼的地方。

在.ks 腳本裡插入 TJS 部分的時候經常會用到。

（4）“kag.”開頭的變數

沒有錯，這些 kag.開頭的東西，和 f.開頭的東西都一樣是變數。

只不過他們是 kag 系統提供的內置變數。

許多無法通過 KAG 指令實現的東西都可以通過修改這些變數來達成效果。

修改系統設定的值時，最常用的已經在上面列出來了。就是這些：

kag.fullScreen 是否全屏
kag.allskip 是否跳過未讀部分
kag.bgmvolume 音樂音量
kag.sevolume 音效音量
kag.textspeed 文字速度
kag.autospeed 文字自動前進速度