

THE NVL MAKER

新手教程

(五)

存取界面

# 一、工程設定

之前已經在系統設定有關的教程裡大致講了下 Config。

裡面存取相關的東西主要就是這些。

全局设定 档案与通过记录 菜单设定 文字层设定 字体设定 图形与音声设定 历史

**存档文件设置**

存档目录

文件前缀

档案格式

存档总数

☐ 在存档中保存宏信息

**通过记录**

☒ 自动记录已读标签

记录模式

记录数量

**自动标签 (AutoLabel) 模式**

☒ 使用自动标签 (按行存档)

标签模式

**缩略图**

☒ 存档时保存缩略图

缩略图宽度

色彩模式

**特殊存档模式**

☐ 只读模式

☐ 自由存档

资料ID

檔案總數：

假設你預計存儲界面一頁顯示 5 個檔案，一共打算 5 頁，那麼這裡就設定 25，是很簡單的計算。

保存縮略圖：

推薦不用去動它，就一直勾著。

圖片寬度：

具體的大小會顯示在界面編輯裡。

假如是使用懸停效果，那麼值可以設大一點，如果是每個按鈕上都要放小截圖，那當然是小一點的好。

## 二、存儲、讀取圖形設定

因為存取部分要的元素比較多，所以把所有圖形有關的都單獨出來，界面編輯器裡只負責設定排版位置。因此有的時候可能需要在這幾個設定中間來回切換以查看效果了。

## 三、界面編輯器

### （1）存檔按鈕數量、位置設定

每個按鈕對應的就是一個存檔位置。每頁的檔案個數可以在這裡設定：



档案设定			
每页档案数	3		设定
档案1	x	320	y 120
档案2	x	320	y 270
档案3	x	320	y 420

當輸入數值以後點“設定”，增加或減少每頁的存檔按鈕數量。

接著返回上層界面，就可以拖動新的存檔按鈕的位置了。

不過注意，每頁的檔案數肯定不能超過工程設定裡的“檔案總數”。要是設定了太大的值，編輯器會自動調整。

通過拖動調整新的存檔按鈕位置：



## (2) 存檔按鈕上的信息顯示



在“内容设定”裡可以調整存檔按鈕上顯示的信息。

勾選必要的信息並設定在按鈕上的相對位置，就可以顯示不同的文字或截圖。

### （3）新檔標記

設定以後，玩家最後一次存檔的欄位附近會顯示一張圖片標記。

### （4）懸停效果



所謂“懸停效果”，是指當鼠標移動到某個存檔按鈕上時，會進一步在畫面其他地方顯示這個存檔的詳細信息。而當鼠標移開時，信息就被隱藏。

支持顯示的內容和存檔按鈕上可以顯示的內容差不多。

## 四、繼續自己寫系統

相信上次的系統設定裡加入“恢復默認值”按鈕的體驗並不是很嚇人。這次講的依然是通過修改腳本來增加編輯器裡沒有的功能，準備好了嗎？不過，因為這次有存儲、讀取兩個系統，所以一共要改兩個文件。當然就是 macro 文件夾下的 save.ks 和 load.ks 了。

### （1）宏

首先來說說之前一直提的“宏”吧。

所謂“宏（macro）”，作用是把一直重複用到的幾個指令合併成一個新的指令，然後使用這個新的指令來節省工作量。

在 KAG 腳本裡，定義了[macro name=宏名稱]和[endmacro]中間的內容後，就可以用[宏名稱]來調用指令。

比如說

```
[macro name=hero]
[emb exp="f.姓"]
[emb exp="f.名"]
[endmacro]
```

那麼每次使用[hero]就代表在對話裡顯示了 f.姓的值和 f.名的值。

可能一開始寫腳本的時候並不會感到不用宏有什麼問題，不過，遊戲的某個特定效果總是要頻繁出現，哪怕重複的部分只有兩三行也相當煩人了。如果能用一行來解決，總是比兩行要快一些。

還有一個好處就在於，當你需要修改這個效果的時候——如果用的是宏，只要修改一個地方，如果你在每個地方都複製粘貼了這幾行代碼，就準備全文搜索吧……

總的來說，用 KRKR 的有三種人……

## (2) 普通青年&牛逼青年&213 青年

	腳本一	腳本二	介紹
普通青年的腳本	<pre>[iscript] //函數 Function abc() {   Xyz; } [endscript] ;宏 [macro name=def] [ghi] [jkl] [opq] [rst] [endmacro] ;----- [return]</pre>	<pre>;定義宏 [call storage="腳本一"]  ;XX 界面 *start [rclick] [eval exp="abc()"] [def] [s]  ;返回遊戲 *return [return]</pre>	<p>使用 KS 的時候比用 TJS 的時候更多。</p> <p>函數和宏放在一個腳本，實際的執行邏輯放在另外一個腳本裡。</p> <p>總的來說中規中矩。</p>
牛逼青年的腳本	<pre>Class abc{   Var ghi;   Var jkl;    function abc(..)   function finalize(...)    Property xyz{   } }</pre>	<pre>[button exp="f.ui=new abc()"]</pre>	<p>除了劇情腳本，幾乎找不到 KS 腳本的存在。</p> <p>大部分功能都封裝成 TJS 的類。</p> <p>調用的時候直接用一個 TJS 式搞定。</p>
213 青年的腳本	<pre>*start [rclick] [ghi] [jkl] [opq] [jump target=*label3] *label1 [rst] [ghi] [jkl] [opq] [jump target=*start] （……因為太長了所以接右邊）</pre>	<pre>*label2 [rst] [ghi] [jkl] [opq] [jump target=*label4] *label3 [rst] [s] *label4 [ghi] [jkl] [return]</pre>	<p>沒有第二個腳本</p> <p>所有的東西都寫在一起。</p> <p>凡是需要重複使用的，都靠複製粘貼。</p> <p>大量使用跳轉蹦到奇怪的地方，搞到最後自己也不知道到底執行邏輯是什麼樣。</p> <p>而且，完全沒有注釋……=_=</p>

### (3) 確定要添加內容的位置

THE NVL Maker 作者屬<sup>2</sup>標準的普通青年，所以遊戲模板工程基本就是個宏和函數的大集合。  
存取系統相關的函數和宏，放在 `macro_sl.ks` 裡。  
從而保證 `save` 和 `load` 的腳本內容依然非常簡單。

```
;-----  
;非常懶惰的 save 顯示  
;-----  
*start  
[locksnapshot]  
[tempsave]  
;-----  
*window  
;載入配置文件  
[iscript]  
f.config_sl=Scripts.evalStorage("uisave.tjs");  
[endscript]  
[history enabled="false"]  
  
*firstpage  
[locklink]  
[rclick enabled="true" jump="true" storage="save.ks" target=*返回]  
[backlay]  
[image layer=14 page=back storage=&"f.config_sl.bgd" left=0 top=0 visible="true"]  
  
;隱藏系統按鈕層  
[hidesysbutton page="back"]  
;用 message4 描繪  
[current layer="message4" page="back"]  
[layopt layer="message4" visible="true" page="back" left=0 top=0]  
[er]  
[eval exp="drawslbutton('back')"]  
;這裡是我們要添加內容的地方  
  
[trans method="crossfade" time=500]  
[wt]  
[s]  
;-----  
*刷新畫面  
[current layer="message4"]  
[er]  
[eval exp="drawslbutton()"]  
;這裡是我們要添加內容的地方
```



```
[s]
```

```
;-----
```

(以下省略)

## (4) 加入“頁數選擇”按鈕

THE NVL Maker 的界面編輯器裡提供了“向上翻頁”和“向下翻頁”兩個按鈕。可以在不同的存檔頁中間切換。但是如果存檔頁比較多，從第一頁切到第五頁等等，可能就要多點很多次按鈕。玩家也並不是很清楚一共有多少頁存檔。

所以有的時候，會隱藏翻頁按鈕，而直接加入頁數選擇按鈕。

假設存檔一共有 5 頁，那麼就在界面上添加【1】~【5】的 5 個數字按鈕。

這裡的按鈕圖片我們就先借用一下 image 文件夾裡的圖形數字吧。

我想在系統設定 (option.ks) 裡，[locate]和[button]的使用，各位還是記得的？

這回也不過是設定按鈕，只不過一個變成了 5 個。

首先在剛打開界面的時候進行描繪。

```
;用 message4 描繪
```

```
[current layer="message4" page="back"]
```

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

```
[er]
```

```
[eval exp="drawslbutton('back')"]
```

;這裡是我們要添加內容的地方

```
[locate x=50 y=150]
```

```
[button normal="num-1"]
```

```
[locate x=50 y=190]
```

```
[button normal="num-2"]
```

```
[locate x=50 y=230]
```

```
[button normal="num-3"]
```

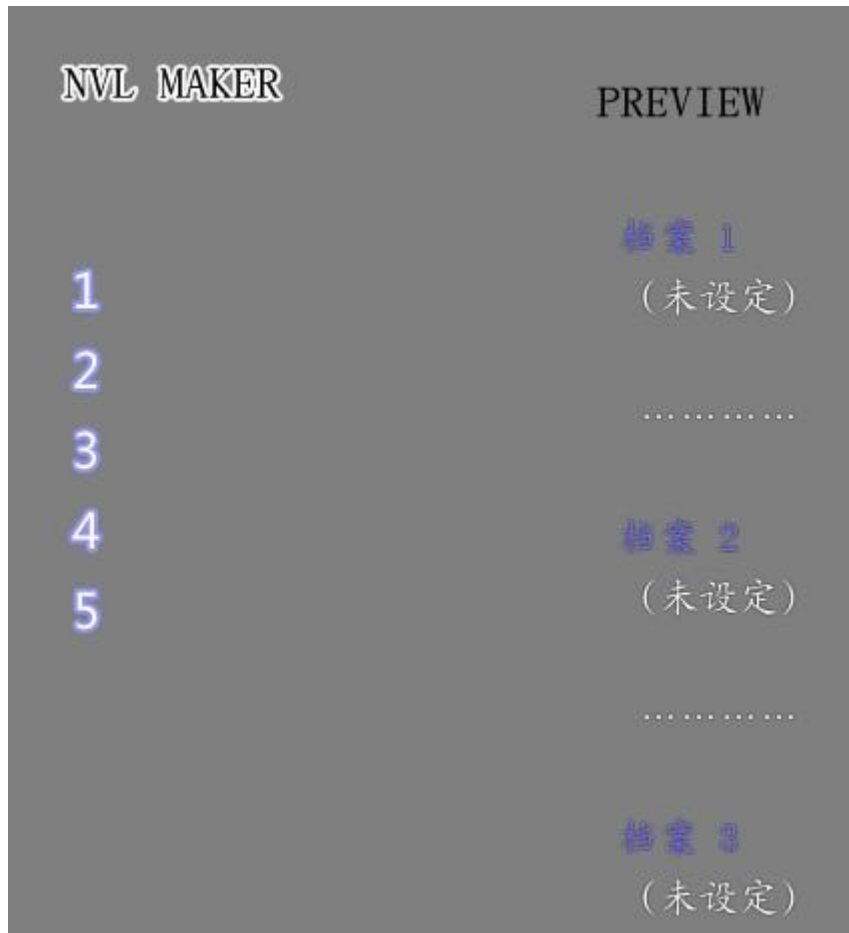
```
[locate x=50 y=270]
```

```
[button normal="num-4"]
```

```
[locate x=50 y=310]
```

```
[button normal="num-5"]
```

現在測試一下：



左邊顯示了 5 個按鈕。似乎是成功了。

接下來，為了避免成為 213 青年，把它封裝成宏。

## （5）封裝

所謂“封裝”，這裡的意思就是定義一個宏。

而當系統讀過這個宏定義以後，你就可以在以後的腳本裡使用這個定義了。

假設把這個宏的名字叫做[存取系統\_翻頁按鈕]。

接下來找個地方，加入自己的宏定義吧？

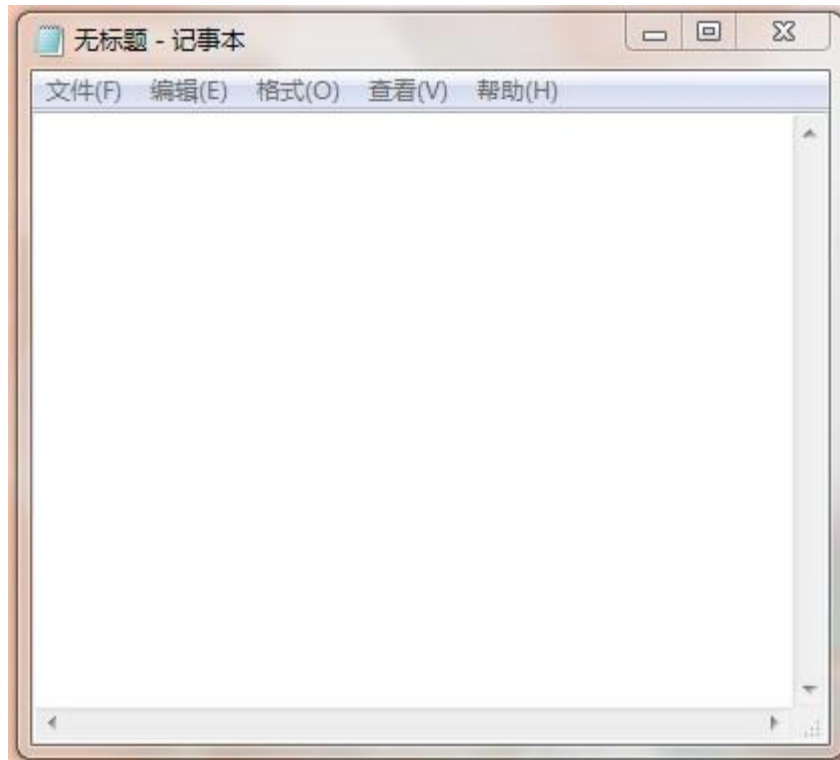
要做的事情是，

一，新建一個自己的腳本用來保存宏定義

二，把宏定義寫進腳本

三，讓系統讀入這個腳本一次

首先，打開記事本，



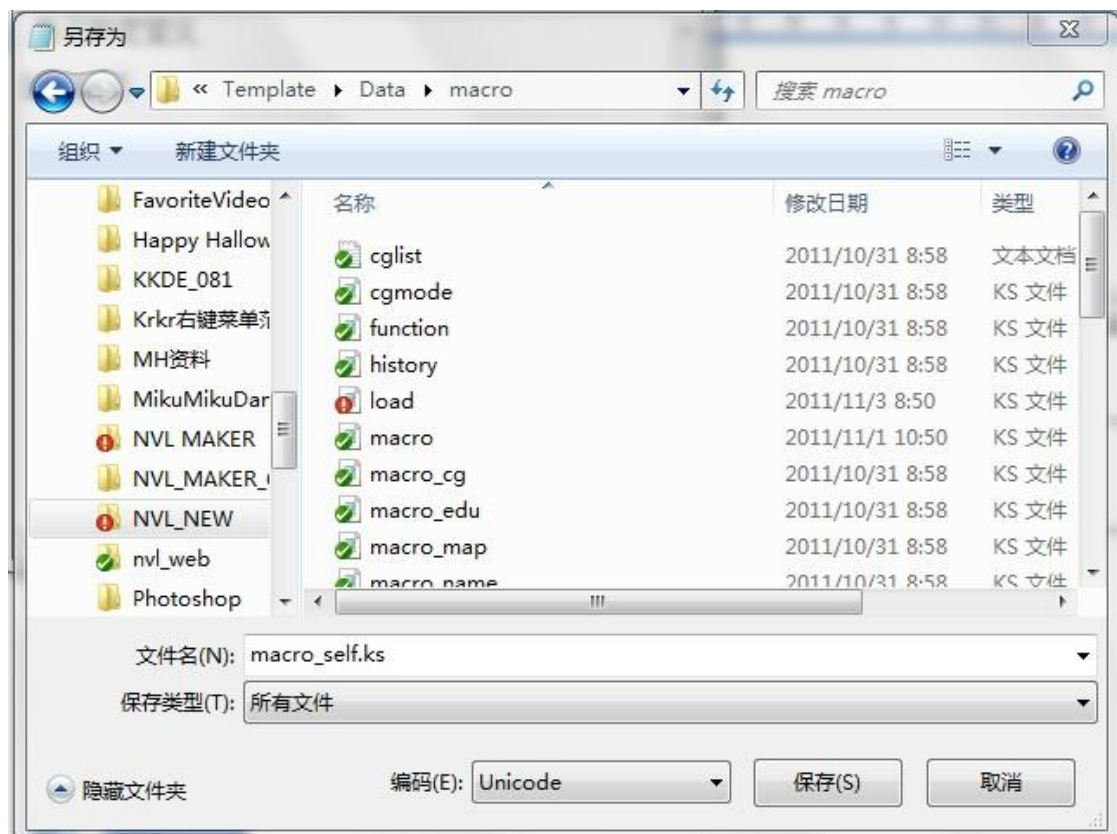
填進去以下內容：

;自己的宏定義

[return]

然後選擇保存。

不要忘記[return]哦。



這裡有兩個地方要注意，第一個是保存類型選擇“所有文件”，然後填寫文件名

macro\_self.ks。第二個則是這裡：

編碼(E): Unicode

，必須要設定成

Unicode。

假如你的 windows 系統不提供自動轉存 Unicode 格式的話，可以用 THE NVL Maker 的“腳本編輯-新建腳本”功能，存下一個空白腳本，再打開進行編輯。

將所有的腳本存成 Unicode（Unicode-LE）編碼可以保證你的遊戲在任何語言版本的操作系統下都可以正常執行而不報錯，所以非常重要。

接下來，繼續修改腳本的內容：

;自己的宏定義

[macro name=存取系統\_翻頁按鈕]

[endmacro]

[return]

現在直接把剛剛填寫的按鈕位置和設定直接剪切過來貼到這裡。變成這樣：

;自己的宏定義

```
[macro name=存取系統_翻頁按鈕]
```

```
[locate x=50 y=150]
```

```
[button normal="num-1"]
```

```
[locate x=50 y=190]
```

```
[button normal="num-2"]
```

```
[locate x=50 y=230]
```

```
[button normal="num-3"]
```

```
[locate x=50 y=270]
```

```
[button normal="num-4"]
```

```
[locate x=50 y=310]
```

```
[button normal="num-5"]
```

```
[endmacro]
```

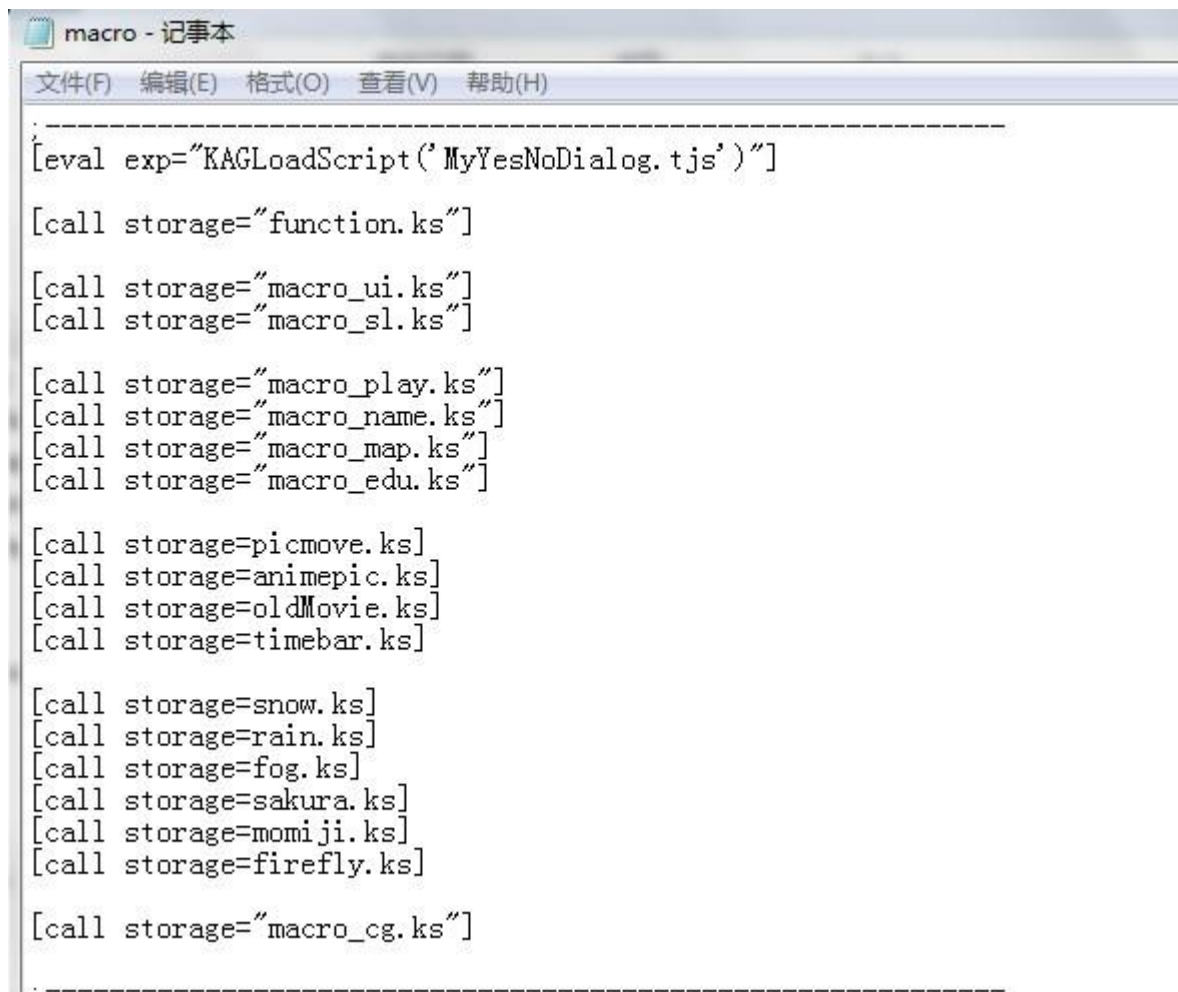
```
[return]
```

定義完畢。

現在，新建的這個腳本從來沒被系統讀取過，因此這個宏雖然寫完了，卻還沒人知道。我們需要在某個已經存在的腳本裡“呼叫”一次這個腳本。

這裡 **macro.ks** 是最好選擇，因為 **THE NVL Maker** 工程所有的宏腳本都是在這裡讀取的。

打開 **macro.ks**



```
macro - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

[eval exp="KAGLoadScript('MyYesNoDialog.tjs')"]

[call storage="function.ks"]

[call storage="macro_ui.ks"]
[call storage="macro_sl.ks"]

[call storage="macro_play.ks"]
[call storage="macro_name.ks"]
[call storage="macro_map.ks"]
[call storage="macro_edu.ks"]

[call storage="picmove.ks"]
[call storage="animepic.ks"]
[call storage="oldMovie.ks"]
[call storage="timebar.ks"]

[call storage="snow.ks"]
[call storage="rain.ks"]
[call storage="fog.ks"]
[call storage="sakura.ks"]
[call storage="momiji.ks"]
[call storage="firefly.ks"]

[call storage="macro_cg.ks"]
```

在[call storage="macro\_cg.ks"]下面加入一句

[call storage="macro\_self.ks"]。

保存，關閉。——現在你在任何地方使用[存取系統\_翻頁按鈕]，就等於建立了這五個按鈕。

## （6）開始使用宏

之前已經確定了 save.ks 裡有兩個需要添加按鈕的地方，而 load.ks 裡也同樣。

現在這四個地方都修改成這樣。

```
.....
[eval exp="drawslbutton('back')"]
;這裡是我們要添加內容的地方
[存取系統_翻頁按鈕]
.....
```

沒有錯，只要一行！四個地方都只要一行！

親愛的觀眾朋友們，你還在等什麼呢，現在打電話訂購宏，還贈送函數……（咦）

## （7）繼續修改宏的內容

定義了宏以後最大的好處就是這個了，只要修改宏的內容，四個地方的效果都會改變。

接下來當然是為按鈕添加點下以後的操作了。

之前在製作選擇的時候有說過，可以讓按鈕點下之後跳轉到同一個地方，同時設定一個數值。這裡所有的翻頁按鈕，都是跳轉到\*刷新畫面，而修改的數值就是“現在翻到第幾頁”。

在 THE NVL Maker 的存取系統裡，“sf.最近存儲頁”代表的就是存取系統現在翻到的頁數。點下按鈕 1，就設定 1，點下 2，就設定 2，以此類推。（當然你要保證你在工程設定裡的確設定的存檔總數夠分成這麼多頁的……默認是 12 個，這裡至少需要 15 個。）

因此我們現在可以把 macro\_self.ks 裡的[macro name=存取系統\_翻頁按鈕]這個宏改成下面這樣：

```
[macro name=存取系統_翻頁按鈕]

[locate x=50 y=150]
[button normal="num-1" target=*刷新畫面 exp="sf.最近存儲頁=1"]
[locate x=50 y=190]
[button normal="num-2" target=*刷新畫面 exp="sf.最近存儲頁=2"]
[locate x=50 y=230]
[button normal="num-3" target=*刷新畫面 exp="sf.最近存儲頁=3"]
[locate x=50 y=270]
[button normal="num-4" target=*刷新畫面 exp="sf.最近存儲頁=4"]
[locate x=50 y=310]
[button normal="num-5" target=*刷新畫面 exp="sf.最近存儲頁=5"]
[endmacro]
```

如果你有兩個不同狀態的數字圖片，還可以用 cond 配合數字圖片，用不同按鈕來表示當前翻到哪一頁。例如：

```
[button normal="num-1" over="num-1_over" target=*刷新畫面 exp="sf.最近存儲頁=1"
cond="sf.最近存儲頁!=1"]
[button normal="num-1_over" target=*刷新畫面 exp="sf.最近存儲頁=1" cond="sf.最近存儲頁
==1"]
```

## 五、從翻頁按鈕改成頁數顯示

其實單純的“向上翻頁”和“向下翻頁”也不錯，但是有時候並不知道現在翻到的是哪一頁也有點麻煩。

假如可以直接在畫面上有個地方顯示現在翻到的頁數，不是比做很多個按鈕更簡單嗎。

所以現在乾脆去掉這些按鈕，再修改一下宏的內容。

### （1）在一個宏定義裡使用其他宏

一旦宏被定義以後，就可以像普通的 KAG 指令一樣使用，所以它同樣也可以被用在其他宏裡。所謂偷懶，就是這樣簡單……

這次可以利用到養成部分的宏裡的一個，就是“顯示圖片數字”，在 macro\_edu.ks 裡。名

字叫[drawnum]。

現在，macro\_self 裡的宏變成了這樣：

```
[macro name=存取系統_翻頁按鈕]
[draw_num num=&"sf.最近存儲頁" pic="num-" x=50 y=150 layer=15 page=%page | back]
[endmacro]
```

先解釋一下，layer15 是存儲、讀取界面用來描繪縮略圖等等的專用圖層，因此這裡只要將圖片數字描繪在層 15 上，就可以顯示出來了。

吉裡吉裡中，層是很重要的概念。

圖片需要顯示在圖片層上，而文字則需要顯示在文字層（也可以叫做對話框層、消息層或者 message 層）上。

最後這些圖層、文字層按照一定順序疊加起來，就變成了完整的畫面。

而 pic 則是設定了圖片數字的前綴名，這裡默認是 num-，代表圖片 num-0~num-9，你可以設定自己的數字圖片。

x 和 y 自然是顯示在層上的坐標了。

至於 num，這裡是用來顯示數字的。但是我們需要它顯示的是一個變數的值，而不是單純的數字。所以使用了&"變數名"來表示“取得這個變數的值”。

&是 KAG 裡獨有的對變數取值的符號。在 TJS 式和 TJS 代碼裡有其他的寫法，請不要照搬誤用。

至於最後這個 page=%page|back 是什麼意思先不管它，首先來測試一下。



上次打開的時候是翻到第五頁，所以這次系統自動翻到了第五頁，並且正常顯示了頁碼。



現在可以按一下“preview”和“next”……  
……頁碼數字不見了。  
但是假如關閉這個界面再打開，數字會再次顯示出來。  
為什麼只有第一次打開才有效呢？

## （2）page 的“表”和“裡”

之前在選擇的製作一章裡，有講過下面的內容。

文字和圖片，設定在“表頁”的時候，可以立即顯示。  
而設定在“裡頁”的時候，可以通過“執行切換”和“等待切換”來做出各種各樣的效果。  
（比如剛剛的選擇按鈕，就是漸入顯示的。）

“裡頁”可以看做是舞臺的“後臺”，當一切準備都做好以後，才通過“切換”拉開幕布，把效果顯示到前臺來。  
而“表頁”就是立竿見影的了。  
每個圖層、文字層都有“表頁”和“裡頁”，可以單獨設定。

為什麼之前的按鈕宏就沒有必要指定 page 呢，原因就是在存取系統之前的代碼裡，已經用過[`current layer=message4 (page="fore")`]和[`current layer="message4" page="back"`]  
來指定按鈕是顯示在 message4 的“表”或者“裡”了。這是只有 message 層才有的福利。  
圖片層的話，就只好一層一層手動指定了……

存取界面第一次打開的時候，所有的內容也是漸入顯示的，當翻頁的時候，就變成瞬間顯示了。第一次打開正常，翻頁不正常，一定是瞬間顯示的部分出了問題。  
試著通過修改代碼來解決這個問題吧。

## （3）在宏裡使用參數

宏就像普通指令一樣，有名字，還有具體的參數內容。  
這些參數也是在宏定義裡設定的。  
比如說：

```
[macro name=特殊文字]
[font size=%size color=0xFF0000]
[endmacro]
```

這樣當使用[特殊文字]的時候，不但可以顯示紅色字體，還可以同時用 size 指定這個特殊文字的大小。比如說[特殊文字 size=20]

**%用來指定宏的參數名，只會在宏定義裡出現，請不要在其他地方誤用。**

之前提到的 `page=%page` 就是用來設定叫 page 的參數的，這個參數指定了頁碼數字應該顯示在表頁還是裡頁的。

後面的|back 代表不填寫參數時候的默認值。

因為默認的是“裡頁（back）”，所以什麼參數都不填寫的時候，數字就被描繪在這裡。如果想要瞬間顯示，就需要指定為描繪在“表頁（fore）”。

所以，save.ks 和 load.ks 裡，各有一個地方需要做出小小的修正，就是加上紅字部分的參數。

```
*刷新畫面  
[current layer="message4"]  
[er]  
[eval exp="drawslbutton()"]  
[存取系統_翻頁按鈕 page=fore]  
[s]
```

保存測試。

