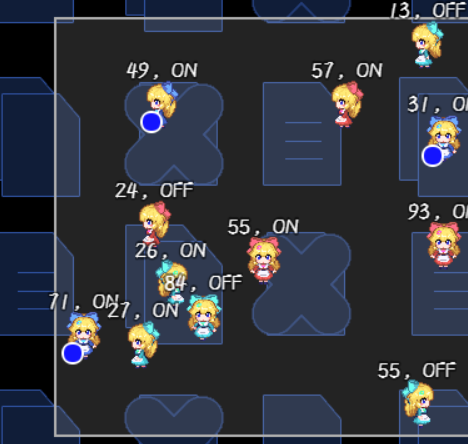
## 概述

### 基础要求

复制事件的批量管理 如下图。（标记地图中所有蓝色小爱丽丝）



|  |  |
| --- | --- |
| 先尝试回答下面的问题：  什么是变量、开关、变量数组？  什么是事件的缓存数据？  什么是同步删除？  什么是批量变量、批量槽位、批量事件？ | 需要先了解基础知识哦！ |

如果你对上述问题有疑问，那么说明你还不了解 变量数组、缓存数据 的用法。

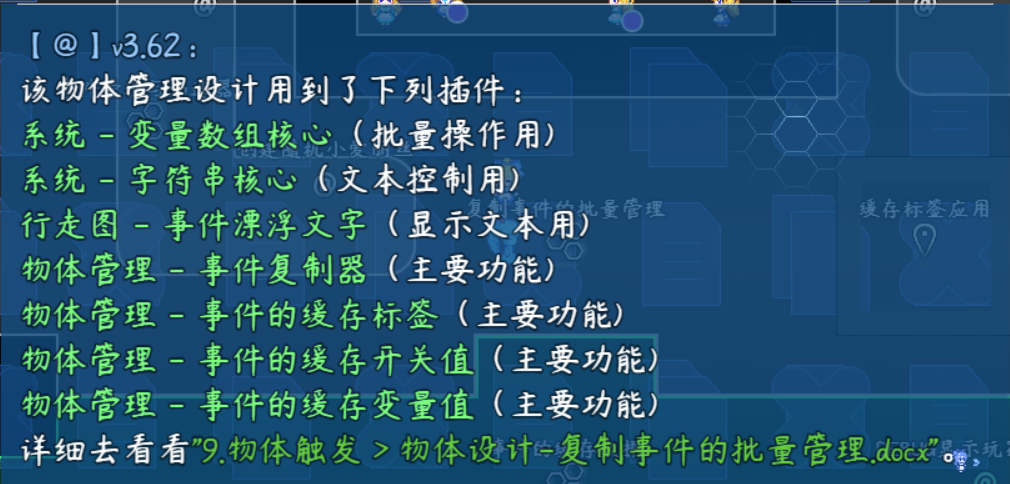
需要去看看：”1.系统 > 关于变量数组核心.docx”。

需要去看看：”28.物体管理 > 关于事件的缓存数据.docx”。

先把两个文档的内容都过目一遍，留个印象。

### 基础插件

具体设计要用到的插件如下图。



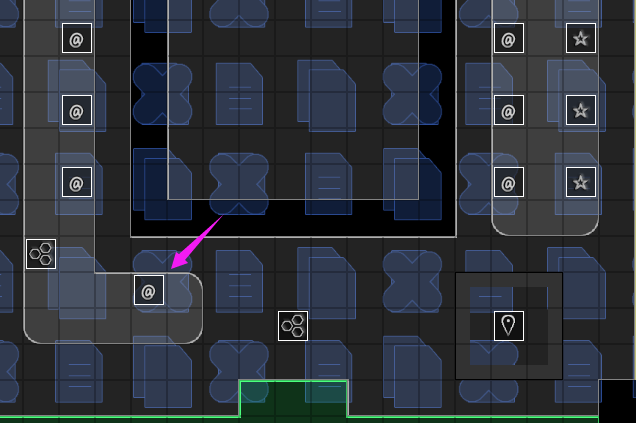
### 基础配置

具体注意示例下面的地方：

**1）地图事件**

需要留意 物体管理管理层 中的事件，

以及 模板管理层 复制用的小爱丽丝。





**2）插件配置**

需要留意在使用 字符串、变量数组 前，

先清空值，或者直接 获取覆盖 旧值，

确保其它情况对 临时数组 赋的值不会影响当前的执行内容。



## 管理说明

基于这两个插件，我们可以复制无数个事件。

◆Drill\_CoreOfEventManager 物体管理 - 事件管理核心

◆Drill\_EventDuplicator 物体管理 - 事件复制器

但是，如果复制了事件，就不管它了，

那么长时间不离开地图，会造成事件大量积压从而游戏卡死问题。

比如大量弹丸事件。



因此，为了管理复制的事件，通常有下面几种方式：

1）固定执行流程

复制的事件中为固定的执行流程，流程执行完之后，执行自我销毁。

2）定时的生命

即复制的事件的生命是定时的，时间到了之后，自我销毁。

3）数组记录

复制后，事件id会标记到一个数组中，在需要统一销毁时，才执行销毁。

4）缓存标签记录

复制的事件会被一个特定的缓存标签进行标记，根据标签直接统一销毁。

|  |
| --- |
| 另外，离开地图后，所有事件必然销毁，  因此只要切换了地图，就不需要担心之前复制的任何事件了。 |

## 管理 - 复制事件

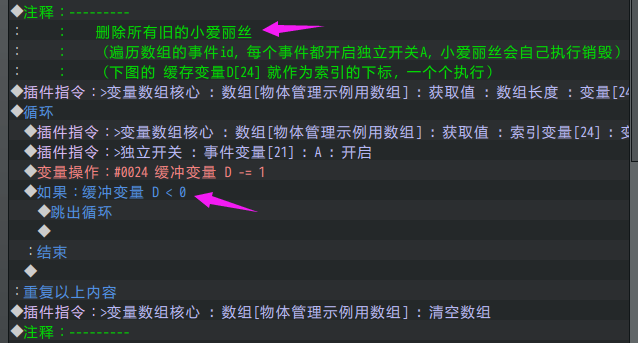
以前面章节 [管理说明](#_管理说明) 中 数组记录 的方式为例。

复制事件的管理方式如下。

### 1）删除旧事件

数组记录了事件id，那么数组要执行复制新一组事件之前，要考虑先销毁旧的事件。

（注意，这个数组在游戏初始化时需要为空，不然会删错事件）

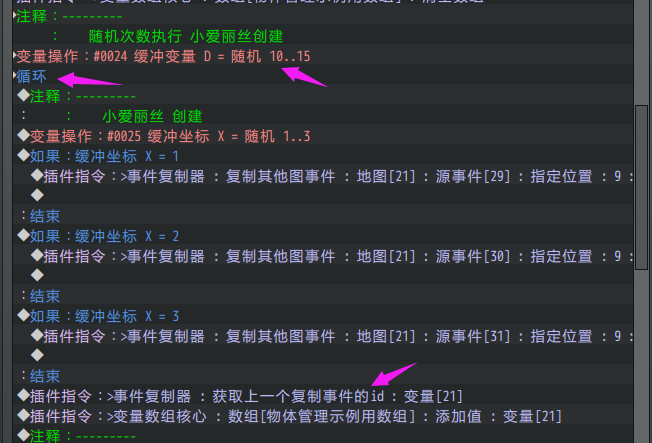


### 2）复制事件

如下图，如果需要复制多个事件，必须添加一个循环，在循环里面一个个复制事件。

使用事件复制器之后，需要使用变量获取“获取上一个复制事件的id”，

然后将此id添加的数组中。



循环执行完后，可以用debug查看数组，看看复制事件的id情况。

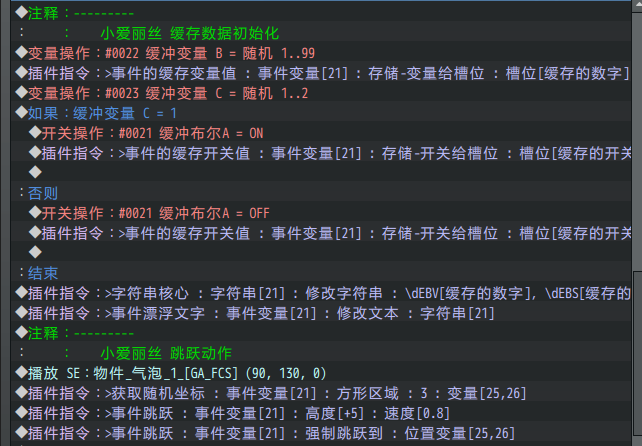


### 3）事件初始化

前面步骤将事件进行了复制，并且事件id暂存在 变量[21] 中。

那么，接下来就可以写 复制的事件 初始化的功能了，使用 事件变量[21] 即可。

如下图，可以对缓存数据初始化，可以执行跳跃动作。



|  |
| --- |
| 注意，复制事件的初始化，必须都在同一帧内执行，  也就是说，不能等待，否则变量[21]可能会被其他事件使用赋值，从而错位。 |

### 4）离开地图时

离开地图时，所有事件会被全部销毁。

因此，数组中记录的事件id，也应该在离开地图时，及时清空。

不然再次返回地图，数组会给出错误的事件id值。



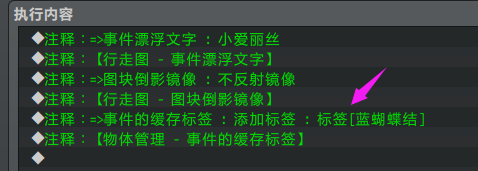
变量数组核心中提供了自动清空的设置，使用插件指令开启设置即可。

## 管理 - 筛选事件

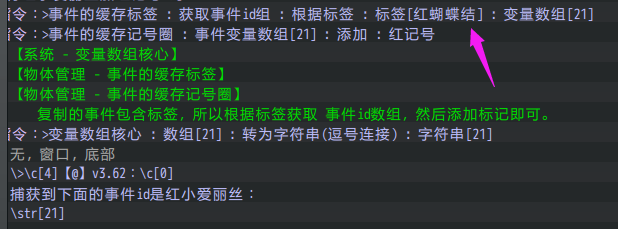
### 获取指定标签的事件

模板管理层中，三个小爱丽丝分别自带了“蓝蝴蝶结”、“红蝴蝶结”、“青蝴蝶结”三个不同的标签。





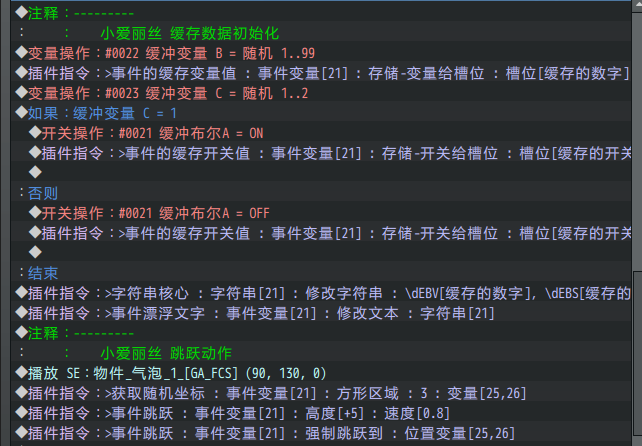
直接使用插件指令“获取事件id组”即可。



### 获取缓存数字大于50的事件

首先，要确保缓存数据中真的有值。

下图为在 [管理 - 复制事件](#_管理_-_复制事件) 时，缓冲变量B 进行的槽位赋值。



然后，准备两个临时数组：

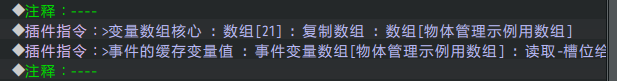
》数组[21]为 事件id，

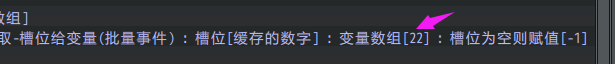
（复制事件的时候，就记录了数组，复制那个数组）

》数组[22]为 缓存数字。

（插件指令有点长，就是根据实际变量数组，获取批量事件的 槽位[缓存的数字] ）

（此长插件指令详细介绍可以去看看：“28.物体管理 > 关于事件的缓存数据.docx”）





接着，依据变量数组的 同步删除功能，

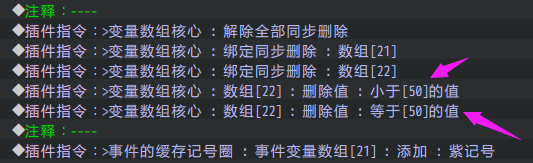
删除缓存数字，就能同步删除对应的事件id。

（删除同步详细介绍可以去看看：“1.系统 > 关于变量数组核心.docx”）



注意，我们要的是“大于50”的事件，

因此要删除不符合条件的事件，所以需要删除 小于[50]和等于[50] 的事件。



最后，数组[21] 留下的就是“大于50”的事件id。

使用此事件id进行记号添加，或者其他操作即可。

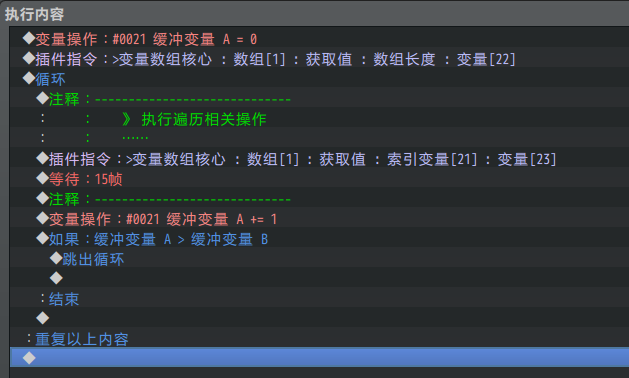
## 管理 - 遍历事件

目前遍历事件的功能暂时没被用到，

因为使用 同步删除+数组操作 功能，就能满足大部分功能。

但这里还是要提一下。（使用事件指令写一个for循环，有点麻烦）





上图的循环遍历，用到了3个变量：

缓冲变量A（id 21）： 用于表示当前的索引。

缓冲变量B（id 22）： 用于表示当前数组的长度。

缓冲变量C（id 23）： 用于表示数组中取到的数字，取到即用。

详细内容去看看：”1.系统 > 关于变量数组核心.docx” 的 遍历数组 章节。

|  |
| --- |
| 举个例子，如果要设计数组内的所有元素值+10，  使用插件指令 ”>变量数组核心 : 数组[4] : 修改值 : 全数组值[+10]” 即可，  先看看 数组插件 有没有提供直接批量操作的功能，如果实在没有，才使用循环遍历。 |

## 常见问题（FAQ）

### 刷菜单后，数字全部变相同了

|  |  |
| --- | --- |
| **问题名称** | 刷菜单后，数字全变相同了 |
| **问题图示** |  |
| **问题描述** | 1. 能正常执行事件的复制，并且事件复制后数字是对的。  2. 刷菜单之后数字就全部变相同了。 |
| **原理解析** | 此问题非常常见，作者我在设计时遇到了，这里标记一下。  这是因为使用字符串时，显示用的 \v[22] 临时变量。    复制事件时，这个临时变量每次都在赋值，所以字符串正常显示值。  但是进入菜单返回地图后，会执行贴图销毁重建。  这时候事件漂浮文字会重新寻找 \v[22] 的值，并刷新文本。  而这时候 缓冲变量22中已是最后一次的值，所以就全变成相同值了。 |
| **解决方案** | 使用缓存的变量值来存储 缓冲变量22的值，  然后使用 \dEBV[] 窗口字符显示 缓存值 即可。 |