## 概述

### 相关插件

传感器类：

◆Drill\_EventPressureSwitch 物体 - 重力开关

◆Drill\_EventGatherSwitch 物体 - 聚集开关

◆Drill\_MouseTriggerEvent 鼠标 - 鼠标触发事件

◆Drill\_EventAutoTrigger 物体触发 - 固定区域 & 玩家接近 & 条件触发

触发器类：

◆Drill\_EventMutiSwitch 物体 - 计数开关

◆Drill\_EventSequentialSwitch 物体 - 序列开关

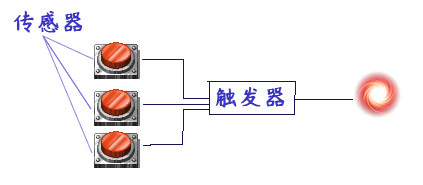
如果要设计复杂精巧的关卡，可能需要运用一定的 **数字电路** 知识。（╭(°A°`)╮没想到几个破开关，支撑起的是整个数电的知识大厦……）

不过。实际游戏设计中并不需要设计过于复杂的开关电路，这里只对一些有特点的开关进行解析。

### 快速区分

**传感器**：遇到某些情况就会自动开启的事件，比如鼠标开关、压在上面的重力开关等。

**触发器**：连接开关与门的中间体，用于控制开关如何开启门。



（二者从结构原理上**没有区别**，都是根据条件自动触发独立开关，但是在设计开关与门路时，需要区别对待，因为从构造上属于两种不同的元件。）

**物体触发**：触发条件更为复杂，对区域、标签、筛选器有非常严格的要求。虽然原理也是触发独立开关，但是功能上与传感器触发器区别很大。



### 插件关系

多数开关插件都是独立的个体，可以单独使用：



## 传感器

### 一次性开关

**1) 结构**

基于rmmv自带的独立开关与事件页。

一次性开关是最基本的开关。

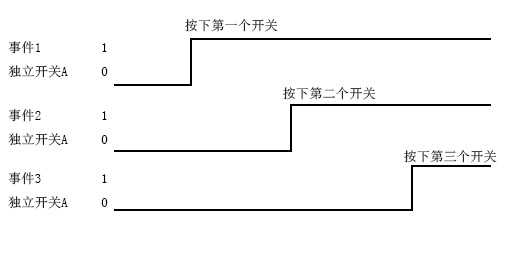




|  |  |  |  |
| --- | --- | --- | --- |
| 事件页 | 条件 | 描述 | 指令 |
| 1 | 无 | 开关未按状态 | 脚踩就开启独立开关A |
| 2 | 独立开关A | 开关按下状态 | 无 |

**2) 信号图**

这里**介绍一下时钟信号图**，假设3个开关控制着门，0表示独立开关 OFF，1表示独立开关 ON，那么就有下图：



根据时钟信号图，第三个开关为1时，三个开关都是on的，也就是说，门这时候是开启状态。由于开关都是一次性的，所以门开启后，也不需要考虑门重新关闭的状态。

### 计时开关

**1) 结构**

基于rmmv自带的独立开关与事件页。

接下来，一次性开关进行一些修改，可以设计成 计时开关：

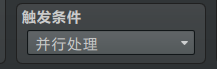
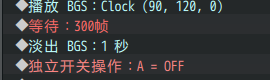


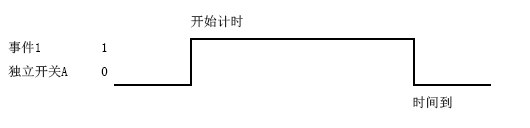


|  |  |  |  |
| --- | --- | --- | --- |
| 事件页 | 条件 | 描述 | 指令 |
| 1 | 无 | 开关未按状态 | 脚踩就开启独立开关A |
| 2 | 独立开关A | 开关按下状态 | 并行控制计时，  时间到则关闭独立开关A |

**2) 信号图**

这里需要设置事件页为并行处理，等待5秒之后，独立开关关闭。



由图可知，计时开始时，独立开关ON，时间到后，独立开关OFF。

### 重力开关

**1) 结构**

可基于Drill\_EventPressureSwitch 物体 - 重力开关。

也可基于Drill\_MouseTriggerEvent 鼠标 - 鼠标触发事件。

重力开关需要插件注释来驱动，踩着会按下，离开会弹起，鼠标触发事件也一样。





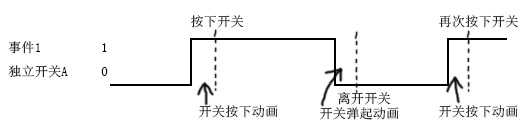




|  |  |  |  |
| --- | --- | --- | --- |
| 事件页 | 条件 | 描述 | 指令 |
| 1 | 无 | 开关按下动作 | 脚踩则立即开启独立开关A，  离开则立即关闭独立开关A。 |
| 2 | 独立开关A | 开关按下状态 | 无 |

**2) 进阶结构**

如果你考虑要附加按下开关时，开关按下的动画，离开开关时，开关弹起的动画，过程入下图：

****



|  |  |  |  |
| --- | --- | --- | --- |
| 事件页 | 条件 | 描述 | 指令 |
| 1 | 无 | 开关弹起状态 | 脚踩则立即开启独立开关A，  离开则立即关闭独立开关A。 |
| 2 | 独立开关B | 开关弹起动作 | 弹起声音。关闭B。 |
| 3 | 独立开关A | 开关按下动作 | 按下声音。开启B、C。 |
| 4 | 独立开关C | 开关按下状态 | 如果A被关了，立即关闭C，  这样就跳转到了事件页2。 |

### 脉冲开关

**1) 结构**

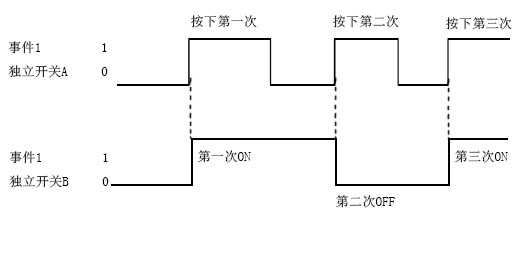
可基于Drill\_EventPressureSwitch 物体 - 重力开关。

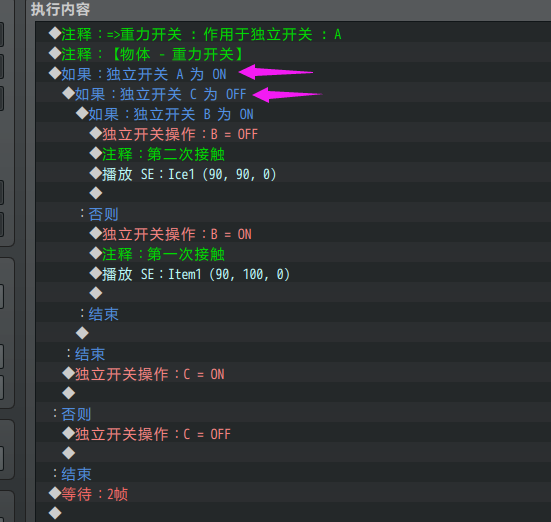
也可基于Drill\_MouseTriggerEvent 鼠标 - 鼠标触发事件。

脉冲开关，是重力开关的**二次迭代**的结果，需要独立开关A、B两个，A为重力开关控制，B为A的迭代。

脉冲开关的原理是：按第一次的时候ON，按第二次的时候OFF，第三次ON，如此往复。







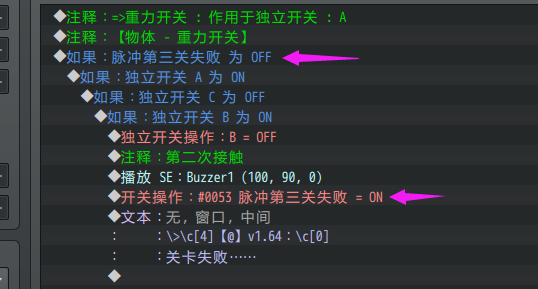
注意，这里的实现重点在于**信号图**。**重力开关反复切换A的开启与关闭，C用于记录上一次A的状态，当A开C关时，说明此时是按下动作，B在这时候开始切换ON/OFF**。B的OFF和ON正好对应事件页1和事件页2。



|  |  |  |  |
| --- | --- | --- | --- |
| 事件页 | 条件 | 描述 | 指令 |
| 1 | 无 | 开关弹起状态 | 接触ON，再接触OFF。 |
| 2 | 独立开关B | 开关按下状态 | 接触ON，再接触OFF。 |

**2) 只能踩一次的脉冲开关**

脉冲开关，在只能踩一次开关的解谜游戏用的非常多，开关一旦踩了第二次，谜题失败，需要重新开始。



**根据这个原理，只需要在B开关OFF的时候，控制关卡失败就可以了，因为B为OFF时，肯定已经接触了一次**。

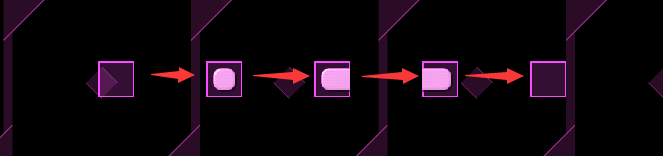
### 多状态开关

**1) 结构**

可基于Drill\_EventPressureSwitch 物体 - 重力开关。

也可基于Drill\_MouseTriggerEvent 鼠标 - 鼠标触发事件。

多状态开关不一定仅仅局限于ON和OFF两种状态，可以设置为N种状态。



脉冲开关是这样的：ON->OFF->ON->OFF->ON->OFF->……

多状态开关是这样的：1->2->3->1->2->3->1->2->3->……

（示例中未写 可以脚踩 的多状态开关，你可以结合逻辑图形关卡的模板和脉冲开关的模板参考。）

**需要注意的是，逻辑图形关卡中，空也是一种状态，图中一共有4种状态**。除去鼠标悬停/动作的环，多状态就如下图所示：



|  |  |  |  |
| --- | --- | --- | --- |
| 事件页 | 条件 | 描述 | 指令 |
| 1 | 无 | 空状态 | 点击后，开A。 |
| 2 | 独立开关A | 状态1 | 点击后，开B关A。 |
| 3 | 独立开关B | 状态2 | 点击后，开C关B。 |
| 4 | 独立开关C | 状态3 | 点击后，关C。 |

**2) 复合结构**

而把鼠标动作的环加进来，看起来就有些复杂了。

图中的结构是示例中，设计-逻辑图形 的多状态开关，实际上有两个环，第一个环（蓝）是状态切换的流程，第二个环（淡蓝）是鼠标点击的流程。



|  |  |  |  |
| --- | --- | --- | --- |
| 事件页 | 条件 | 描述 | 指令 |
| 1 | 无 | 空状态 | 悬停开M1，点击开M3。 |
| 2 | 独立开关M1 | 鼠标悬停动作 | 开M2。 |
| 3 | 独立开关M2 | 鼠标悬停状态 | 如果离开悬停，关M2。 |
| 4 | 独立开关A | 状态1 | 无。 |
| 5 | 独立开关B | 状态2 | 无。 |
| 6 | 独立开关C | 状态3 | 无。 |
| 7 | 独立开关M3 | 鼠标按下动作 | 关M3。根据情况控制状态：  1.开A  4.开B关A。  5.开C关B。  6.关C。 |

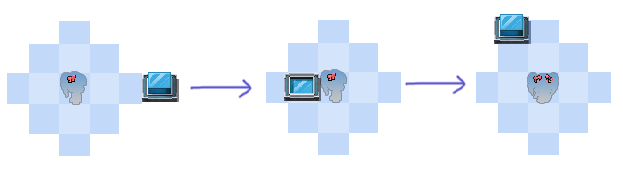
实际上，要添加更多的状态，只需要在6后面加内容，然后修改最后一个事件页所对应的控制就可以了。

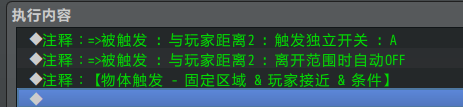
### 接近开关

**1) 结构**

基于Drill\_EventAutoTrigger 物体触发 - 固定区域 & 玩家接近 & 条件触发。

接近开关和重力开关很相似，能够远距离触发事件，只是插件只对玩家的范围有效。







|  |  |  |  |
| --- | --- | --- | --- |
| 事件页 | 条件 | 描述 | 指令 |
| 1 | 无 | 开关弹起状态 | 无。 |
| 2 | 独立开关A | 开关按下状态 | 无。 |

注意，如果没有设置”离开范围自动OFF”，那么事件页2回到1的跳转是没有的。

**2) 自定义区域触发**

接近开关也属于 物体触发类 的插件，区域可以完全自己定制，

可以去看看”关于物体触发-固定区域.docx”。

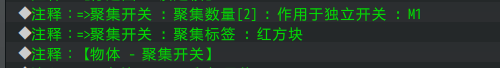
### 聚集开关

**1) 结构**

基于Drill\_EventGatherSwitch 物体 - 聚集开关。

聚集开关即多个相同的开关相邻时，所有相邻的开关都会触发独立开关。







|  |  |  |  |
| --- | --- | --- | --- |
| 事件页 | 条件 | 描述 | 指令 |
| 1 | 无 | 开关弹起状态 | 无。 |
| 2 | 独立开关M1 | 开关按下状态 | 无。 |

示例中的 消除砖块 设计，规则基于该开关。并且，相邻的方块接触后，方块粉碎消除。

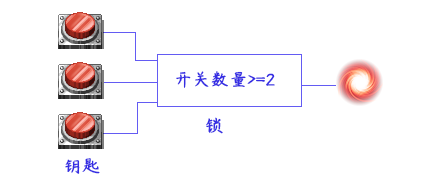
需要注意的是，相邻的硬性规定为：两个事件的距离<=1。

## 触发器

### 计数开关

计数开关顾名思义，根据开关的数量，来控制门。

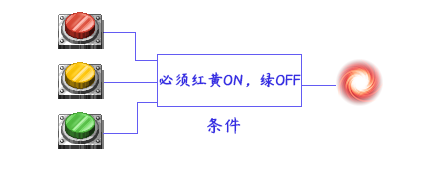
计数开关插件中，在传感器的指定事件页中，设置”钥匙”，在门事件中设置”锁”，这样门会根据”钥匙”的数量，自动开关。



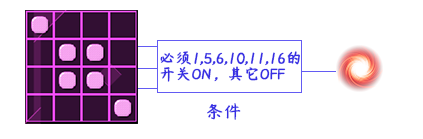
计数开关相对来说，配置比较简单，只要添加匹配的钥匙名和锁名即可。

### 序列开关

序列开关与计数开关不同，条件更加苛刻，对每个开关都有固定的要求。条件设置相对比较复杂。



**上图的三个开关，必须完全满足条件，才可开启门，只要任一条件不满足，立即关闭**。这样一来，按钮的顺序、规则就被牢牢地固定了，你可以用来制作特定图形的解谜。



但是，大多情况下的解谜中，开关数量非常多，你必须将所有事件的条件都写全才能达到想要的效果，条件配置起来会非常麻烦。

## 其他说明

### 性能影响

开关中，有的开关消耗的多，有的几乎没有消耗。不过，有一点可以确定，开关数量越多，性能消耗越大。

一般来说，控制一张地图中开关数量<150个即可。

（示例中的设计关卡，一张地图直接承载了六个关卡的开关数，也未出现明显的卡顿。如果你设计成一张地图一个关卡，那么大可不必担心性能问题。）