## 注释参数

### 文本

在插件注释中，通过@param注解，可以添加配置参数到插件中。

至少要@param、@desc两个配置。

所有注释参数在rmmv中配置之后的值，都是字符串类型，所以在变量获取时，要转类型。

具体转类型方法可以看后面章节 变量获取 。

|  |
| --- |
| \*  \* @param 最大存档数量  \* @desc 最大存档数量的控制参数。  \* @default 20  \* |

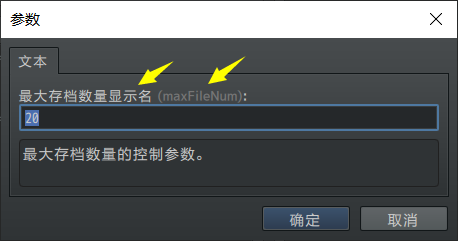
如果不写@type，则默认为text。这里写与不写区别不大。

|  |
| --- |
| \*  \* @param 最大存档数量  \* @type text  \* @desc 最大存档数量的控制参数。  \* @default 20  \* |

这里需要提及一下，@text就是编辑器中显示的名字，有了@text，@param就可以输入英文了。

**但是，由于作者自己的习惯，通过@text来设置参数实属多此一举的行为，因为后期一样要通过parameter来转参数格式到全局变量，所以不建议你们使用@text。**

|  |
| --- |
| \*  \* @param maxFileNum  \* @text 最大存档数量显示名  \* @type text  \* @desc 最大存档数量的控制参数。  \* @default 20  \* |



### 数字

通过设置@type number，可以将输入锁定为数字。

注意，锁定后，只能输入整数数字，无法输入小数。（如果要输入小数、负数，还是改回@type text吧）

|  |
| --- |
| \*  \* @param 最大存档数量  \* @type number  \* @min 1  \* @max 100  \* @desc 最大存档数量的控制参数。  \* @default 20  \* |

### 备注

通过设置@type note，可以将输入变为多行的文本域输入。

|  |
| --- |
| \*  \* @param 内容  \* @type note  \* @desc 最大存档数量的控制参数。  \* @default "描述"  \* |

注意，通过该字符串得到的结果，将会被”\”转义，并且添加了””双引号。

后期如要解析，需要先将内容进行转义。

|  |
| --- |
| var temp = String(DrillUp.parameters['内容']);  temp = temp.substring(1,temp.length-1); //去掉双引号  temp = temp.replace(/\\\\/g,"\\"); //转义双斜杠  DrillUp.g\_XXX\_context = temp; |

### 布尔

通过设置@type boolean，可以将输入变为两个单选，只能选择true和false。

注意，变量获取到的值是字符串”true”和”false”。

|  |
| --- |
| \*  \* @param 布尔  \* @type boolean  \* @on 使用  \* @off 不使用  \* @desc true - 使用，false - 不使用  \* @default true  \* |

### 下拉框

通过设置@type select，可以将输入变为下拉框，只能选择指定的字符串结果。

常用的如下：（option是显示的值，value是真实填入的值）

|  |
| --- |
| \*  \* @param 移动类型  \* @type select  \* @option 匀速移动  \* @value 匀速移动  \* @option 弹性移动  \* @value 弹性移动  \* @option 不移动  \* @value 不移动  \* @desc 初始的移动方式。  \* @default 匀速移动  \* |

也有不常用的写法，但是也可以用：

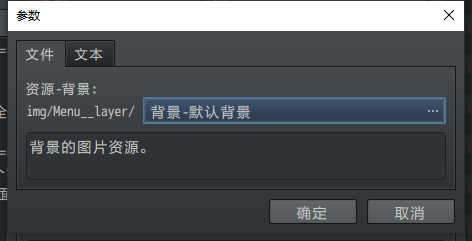
|  |
| --- |
| \*  \* @param 任务类型  \* @type combo  \* @option 初级任务  \* @option 中级任务  \* @option 高级任务  \* @option 特级任务  \* @desc 任务的类型等级。  \* @default 初级任务  \* |

### 文件资源

通过设置@type file来变为文件选择框，通常文件资源会指定一个路径：

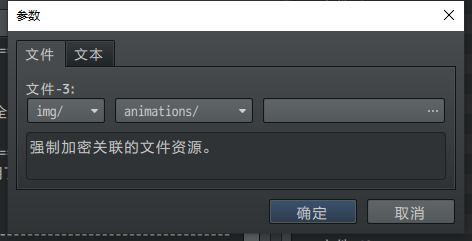
（部分插件中@desc和@default的顺序会乱，不过并不影响控制。）

|  |
| --- |
| \*  \* @param 资源-背景  \* @require 1  \* @dir img/Menu\_\_layer/  \* @type file  \* @desc 背景的图片资源。  \* @default 背景-默认背景  \* |



有时候也可以直接使用泛用路径（不指定@dir）：

|  |
| --- |
| \*  \* @param 文件-3  \* @require 1  \* @type file  \* @desc 强制加密关联的文件资源。  \* @default  \* |



通过上面两种文件资源关联的方式，可以使得在rmmv打包排除时，不被清理。

但是，这种注解最多只能延伸到第二层文件夹，再下层的文件夹，是获取不到的。

### 常用对象id

除了上述常用对象，还有其它可供选择的对象id：

其中0表示空对象。

|  |
| --- |
| \*  \* @param 对象-开关id  \* @type switch  \* @desc 开关id的设置。  \* @default 0  \* |

|  |
| --- |
| \*  \* @param 对象-变量id  \* @type variable  \* @desc 变量id的设置。  \* @default 0  \* |

|  |
| --- |
| \*  \* @param 对象-公共事件id  \* @type common\_event  \* @desc 公共事件id的设置。  \* @default 0  \* |

总结起来如下：

\* @type switch （开关）

\* @type variable （变量）

\* @type animation （动画）

\* @type actor （角色）

\* @type armor （防具）

\* @type item （物品）

\* @type class （职业）

\* @type skill （技能）

\* @type tileset （图块）

\* @type state （状态）

\* @type troop （敌群）

\* @type enemy （敌人）

\* @type weapon （武器）

\* @type common\_event （公共事件）

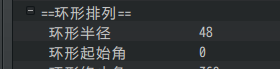
### 父类分支

你可以通过设置@parent来划分参数配置的树结构。

parent名字必须与param的名字对应。

|  |
| --- |
| \*  \* @param ==环形排列==  \* @desc  \*  \* @param 环形半径  \* @parent ==环形排列==  \* @type number  \* @min 0  \* @desc 环形排列时，角色框围绕的半径。单位像素。  \* @default 192  \*  \* @param 环形起始角  \* @parent ==环形排列==  \* @type number  \* @min 0  \* @max 360  \* @desc 环形排列时，第一个角色框所在的角度位置。  \* @default 0  \* |

通过这种方式，”环形半径”和”环形起始角”就出现在”==环形排列==”参数下了。



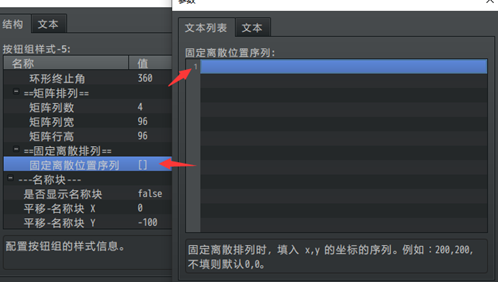
### 数组

直接在@type后面加[]即可。比如 @type text[]、@type number[]、@type note[]。

注意，@default最少要有个[]，不然后期变量获取转换时，很容易报错。

|  |
| --- |
| \*  \* @param 固定离散位置序列  \* @parent ==固定离散排列==  \* @type text[]  \* @desc 固定离散排列时，填入 x,y 的坐标的序列。例如：200,200，不填则默认0,0。  \* @default []  \* |

上面为按钮组中的参数 ”固定离散位置序列” 的功能，点开后效果如下：



注意，数组通过变量获取后，是一长串字符串。需要进行JSON.parse()操作。

具体见后面章节。

### 结构体

结构体是将一系列的变量封装成一个json集合。

首先，需要定义一个结构体：DrillWindowLayout。

结构体的名称完全自定义，只要在当前插件中不重名即可。

但要注意，/\*~struct~DrillWindowLayout: 这一行不能留任何空格。

|  |
| --- |
| /\*~struct~DrillWindowLayout:  \*  \* @param 布局类型  \* @type select  \* @option 默认皮肤  \* @value 默认皮肤  \* @option 单张背景贴图  \* @value 单张背景贴图  \* @option 隐藏布局  \* @value 隐藏布局  \* @desc 窗口布局的类型。  \* @default 单张背景贴图  \*  \* @param ---单张背景贴图---  \* @default  \*  \* @param 资源-贴图  \* @parent ---单张背景贴图---  \* @desc 窗口的背景贴图的资源。  \* @default 背景贴图  \* @require 1  \* @dir img/Menu\_\_self/  \* @type file  \*  \* @param 贴图位置修正 X  \* @parent ---单张背景贴图---  \* @desc 修正图片的位置用。以窗口的点为基准，负数向右，正数向左，单位像素。  \* @default 0  \*  \* @param 贴图位置修正 Y  \* @parent ---单张背景贴图---  \* @desc 修正图片的位置用。以窗口的点为基准，负数向上，正数向下，单位像素。  \* @default 0  \*  \*/ |

然后，通过@type struct<DrillWindowLayout>来设置这个结构体。

（@default中的内容默认{}。也可以先在编辑器里面配置，切换到文本，然后复制粘贴即可）

|  |
| --- |
| \*  \* @param 描述窗口布局  \* @parent ----描述窗口----  \* @type struct<DrillWindowLayout>  \* @desc 控制窗口框架与窗口背景。  \* @default {"布局类型":"单张背景贴图","---单张背景贴图---":"","资源-贴图":"信息面板A-描述窗口","贴图位置修正 X":"0","贴图位置修正 Y":"0"}  \* |

另外，结构体也支持写成数组形式：@type struct<DrillWindowLayout>[]，但是通过这种方式获取到的字符串，需要JSON.parse()两次。注意不要被嵌套层级太多给绕晕了。

### Drill最大值编辑器

插件小工具中，作者我写了最大值编辑器，通过下面注解，可以实现编辑器对该插件的注释编辑。

具体去看看 插件小工具 > 插件最大值编辑器 > 关于插件最大值编辑器.docx 。

|  |
| --- |
| \*  \* @Drill\_LE\_param "阶段-%d"  \* @Drill\_LE\_parentKey "==阶段%d至%d=="  \* @Drill\_LE\_var "DrillUp.g\_xxxx\_length"  \* |

|  |
| --- |
| \*  \* @Drill\_LE\_param "图片-%d"  \* @Drill\_LE\_parentKey "==图片【%d-%d】=="  \* @Drill\_LE\_var "null"  \* |

|  |
| --- |
| \*  \* @Drill\_LE\_param "资源-%d"  \* @Drill\_LE\_parentKey ""  \* @Drill\_LE\_var "null"  \* |

|  |
| --- |
| \*  \* @Drill\_LE\_param "背景音乐-%d"  \* @Drill\_LE\_parentKey ""  \* @Drill\_LE\_var "DrillUp.g\_xxxx\_length"  \* |

相关drill插件也有配置实例，你可以去看看并参考参考。

## 变量获取

### 字符串类型

直接赋值即可，因为全部都是字符串。

不过，为了后期阅读代码合理，这里最好加上String()。

|  |
| --- |
| DrillUp.g\_SMa\_layout\_src = String(DrillUp.parameters['资源-整体布局'] || ""); |

### 数字类型

在赋值时，使用Number()函数强转即可。也可以用parseInt()。

|  |
| --- |
| DrillUp.g\_SCA\_maxFileNum = Number(DrillUp.parameters['最大存档数量'] || 20); |

### 布尔类型

布尔类型不能直接和字符串”true”进行比较，如果直接获取到对象然后比较，很容易出现if不起效的情况。

为了规避这个问题，在赋值时，强行使用等号进行字符串比较，使其变成布尔类型。

|  |
| --- |
| DrillUp.g\_SMa\_mapName\_enable = String(DrillUp.parameters['是否显示所在地图名'] || "true") == "true"; |

### 数组类型

调用时，必须先判断这个参数是否为空字符串（赋值错误情况）或直接未定义（版本更新时的新变量情况）。

|  |
| --- |
| if( DrillUp.parameters["备用角色前视图列表"] != undefined &&  DrillUp.parameters["备用角色前视图列表"] != "" ){  DrillUp.g\_SMa\_actorPicBackupTank = JSON.parse( DrillUp.parameters["备用角色前视图列表"] || [] );  }else{  DrillUp.g\_SMa\_actorPicBackupTank = [];  } |

### 结构体类型

在对结构体进行处理前，最好先写一个变量获取函数。（由于在最外层，函数必须写前面）

这个函数DrillUp.drill\_SSh\_initMenuCursor()可以由后面获取时，进行调用。

这样做的一个好处就是，函数中不需要写复杂的DrillUp.parameters了，而是只考虑dataFrom对象本身的属性数据。

|  |
| --- |
| //==============================  // \* 变量获取 - 指针与边框  // （~struct~DrillCursor）  //==============================  DrillUp.drill\_SSh\_initMenuCursor = function( dataFrom ) {  var data = {};  data['MCu\_enabled'] = String( dataFrom["是否显示菜单指针"] || "true") == "true";  data['MCu\_lock'] = String( dataFrom["是否锁定菜单指针样式"] || "false") == "true";  data['MCu\_style'] = Number( dataFrom["锁定的菜单指针样式"] || 1);  data['MCB\_rectEnabled'] = String( dataFrom["是否启用闪烁白矩形"] || "true") == "true";  data['MCB\_enabled'] = String( dataFrom["是否启用菜单边框"] || "true") == "true";  data['MCB\_lock'] = String( dataFrom["是否锁定菜单边框样式"] || "false") == "true";  data['MCB\_style'] = Number( dataFrom["锁定的菜单边框样式"] || 1);  return data;  } |

调用时，必须先判断这个参数是否为空字符串（赋值错误情况）或直接未定义（版本更新时的新变量情况）。

|  |
| --- |
| if( DrillUp.parameters['购买窗口指针与边框'] != "" &&  DrillUp.parameters['购买窗口指针与边框'] != undefined ){  var cursor = JSON.parse( DrillUp.parameters['购买窗口指针与边框'] );  DrillUp.g\_SSh\_buy\_cursor = DrillUp.drill\_SSh\_initMenuCursor( cursor );  }else{  DrillUp.g\_SSh\_buy\_cursor = null;  } |

如果你觉得DrillUp.g\_SSh\_buy\_cursor为null时，后期函数处理会很麻烦，那么写下面的情况，可以确保这个参数一定不为空：

|  |
| --- |
| if( DrillUp.parameters['购买窗口指针与边框'] != "" &&  DrillUp.parameters['购买窗口指针与边框'] != undefined ){  var cursor = JSON.parse( DrillUp.parameters['购买窗口指针与边框'] );  DrillUp.g\_SSh\_buy\_cursor = DrillUp.drill\_SSh\_initMenuCursor( cursor );  }else{  DrillUp.g\_SSh\_buy\_cursor = DrillUp.drill\_SSh\_initMenuCursor( {} );  } |

如果有 结构体和数组组合、多层嵌套 等更复杂情况，可以去看看：Drill\_SceneMain插件中的用法。