RPGMV脚本API速查

- 信息
  - 显示文字（Show Text）
    - 设置脸图（可选）
    - 设置背景（可选）
    - 设置窗口位置（可选）-
    - 文本内容
  - 显示选项（Show Choices）
    - 选项框背景
    - 窗口位置
    - 选项本体
    - 选项的回调函数
  - 数值输入处理（Input Number）
  - 物品选择处理（Select Item）
  - 显示滚动文字（Show Scrolling Text）
- 游戏进程
  - 开关操作（Control Switches）
  - 变量操作（Control Variables）
  - 独立开关操作（Control Self Switch）
  - 计时器操作（Control Timer）
- 流程控制
  - 分支条件（Conditional Branch）
  - 循环（Loop）
  - 跳出循环（Break Loop）
  - 终止事件处理
  - 公共事件
  - 标签（Label）
  - 转到标签（Jump to Label）
  - 注释（Comment）
- 队伍
  - 增减金币（Change Gold）
  - 增减物品（Change Items）
  - 增减武器（Change Weapons）
  - 增减护甲（Change Armors）
  - 队伍管理（Change Party Member）
  - 获取领队信息*
- 角色
  - 增减HP（Change HP）
  - 增减MP（Change MP）
  - 增减TP（Change TP）
  - 更改状态（Change State）
  - 完全恢复（Recover All）
  - 增减经验值（Change EXP）
  - 增减等级（Change Level）
  - 增减能力值（Change Parameter）
  - 增减技能（Change Skill）
  - 更改装备（Change Equipment）
  - 更改名字（Change Name）
  - 更改职业（Change Class）
  - 更改昵称（Change Nickname）
  - 更改简介（Change Profile）
- 移动
  - 场所移动（Transfer Player）
  - 设置载具位置（Set Vehicle Location）
  - 设置事件位置（Set Event Location）

- 滚动地图（Scroll Map）
- 设置移动路线（Set Movement Route）
- 载具乘降（Getting On and Off Vehicles）

人物

- 更改透明状态
- 更改队列行进
- 集合队列成员
- 显示动画（Show Animation）
- 显示气泡图标（Show Balloon Icon）
- 暂时消除事件
- 设置人物位置*
- 查看队伍朝向*

图片

- 显示图片（Show Picture）
- 移动图片（Move Picture）
- 旋转图片（Rotate Picture）
- 更改图片色调（Tint Picture）
- 消除图片（Erase Picture）
- 清除所有图片*

计时

- 等待

画面

- 淡入淡出
- 更改画面色调
- 闪烁画面
- 震动屏幕
- 设置天气（Set Weather Effect）

音频&视频

- 播放BGM（Play BGM）
- 淡出BGM（Fadeout BGM）
- 保存BGM（Save BGM）
- 还原BGM（Resume BGM）
- 播放BGS（Play BGS）
- 淡出BGS（Fadeout BGS）
- 播放ME（Play ME）
- 播放SE（Play SE）
- 停止SE（Stop SE）
- 播放影像（Play Movie）
- 播放系统音效*

场景控制

- 战斗处理（Battle Processing）
- 商店处理（Shop Processing）
- 名字输入处理（Name Input Processing）
- 打开菜单画面（Open Menu Screen）
- 打开存档画面（Open Save Screen）
- 游戏结束（Game Over）
- 返回标题画面（Return to Title Screen）

系统设置

- 更改战斗BGM（Change Battle BGM）
- 更改胜利ME（Change Victory ME）
- 更改战败ME（Change Defeat ME）
- 更改载具BGM（Change Vehicle BGM）
- 启用/禁用存档（Change Save Access）
- 启用/禁用菜单（Change Menu Access）
- 启用/禁用遇敌
- 启用/禁用整队
- 更改窗口颜色（Change Window Color）
- 更改角色图像

- 更改载具图像
- 游戏时间*
- 显示帧数*
- 关闭游戏*

地图
- 启用/禁用显示地图名称（Change Map Name Display）
- 更改地图图块（Change Tileset）
- 更改战斗背景（Change Battle Back）
- 更改远景（Change Parallax）
- 获取指定位置的信息（Get Location Info）
- 地图尺寸*

战斗
- 增减敌人HP（Change Enemy HP）
- 增减敌人MP（Change Enemy MP）
- 增减敌人TP（Change Enemy TP）
- 更改敌人状态（Change Enemy State）
- 敌人完全恢复（Enemy Recover All）
- 敌人出现（Enemy Appear）
- 敌人变身（Enemy Transform）
- 显示战斗动画（Show Battle Animation）
- 强制战斗行动（Force Action）
- 中断战斗（Abort Battle）
- 获取敌人名字*
- 获取敌人对象*
- 战斗状态判断*

高级
- 脚本（Script）
- 插件指令（Plugin Command）

动画*
- 播放动画

事件*
- 获取事件注释
- 获取指定位置的事件编号
- 运行事件
- 访问事件对象
- 事件对象的移动速度

鼠标*
- 鼠标移动
- 鼠标左键是否正在被按下
- 是否切换为鼠标左键
- 鼠标左键是否连续按下
- 鼠标左键释放
- 鼠标右键点击
- 鼠标位置

键盘*
- 增加WASD控制方向
- 检测键盘事件

存档*
- 载入存档
- 载入全局配置

武器与装备*
- 查看武器信息
- 获取装备信息*

nw.js脚本API速查

窗口
- 获取当前窗口
- 打开新窗口
- 窗口的位置

RPGMV脚本API速查

信息

显示文字（Show Text）

一共有4个需要设定的地方：

设置脸图（可选）

如果想给弹出的文本框设置脸图，那么就可以用这个代码。

参数	参数类型/参数内容	默认值	说明	
faceName	字符串，内容就是脸图的名字		就是脸图文件的名字，事件编辑器那里面选择显示文字，然后点击脸图，就会有一个弹窗。弹窗里面就是脸图的文件，这个faceName就是这些文件的名字。	
faceIndex	0到8		一个脸图有8个头像，从零开始数，第一行是0到4，第二行是5到7。写几就是第几个图。	

```
$gameMessage.setFaceImage(faceName, faceIndex);
```

设置背景（可选）

参数	参数类型/参数内容	默认值	说明	
background	0：窗口 1：暗淡 2：透明		这个参数就对应着背景下拉框的属性	

```
$gameMessage.setBackground(background);
```

设置窗口位置（可选）-

参数	参数类型/参数内容	默认值	说明	
positionType	0：顶部 1：中间 2：底部		这个参数就对应着背景下拉框的属性	

```
$gameMessage.setPositionType(positionType);
```

文本内容

这个就是本指令的核心方法，效果就是弹出来文本框。如果没有配置前面的样式，那么会按照默认样式来显示。

- text：这个就是一个普通的字符串

```
$gameMessage.add(text);
```

示例：

```
$gameMessage.setFaceImage("Actor1",0);
$gameMessage.setBackground(0);
$gameMessage.setPositionType(2);
$gameMessage.add("内容");
```

显示选项（Show Choices）

选项框背景

参数	参数类型/参数内容	默认值	说明	
background	0：窗口 1：暗淡 2：透明		窗体的样式	

```
$gameMessage.setChoiceBackground(0);
```

窗口位置

positionType：有三个固定的数值

- 0：顶部
- 1：中间
- 2：底部

```
$gameMessage.setChoicePositionType(1);
```

选项本体

参数	参数类型/参数内容	默认值	说明	
choices	数组		数组的内容就是选项的内容，比如["是","否"]。**注意！元素必须是字符串类型。**	
defaultType	0，1，2，，，	0	光标默认移动到第几个选项，从0开始。就是说一进入这个选择页面。	
cancelType	0，1，2，，，	0	取消选项时，默认选择哪个选项。	

```
  $gameMessage.setChoices(choices, defaultType, cancelType);
```

选项的回调函数

```
 $gameMessage.setChoiceCallback(callback);
```

示例：

```
$gameMessage.setChoices([ "我","是","选","项"],0,2);
$gameMessage.setChoiceBackground(1);
$gameMessage.setChoicePositionType(1);
$gameMessage.setChoiceCallback(function(n){
  console.log("你选择的是第"+(n+1)+"项！")
});
```

数值输入处理（Input Number）

参数

- variableId：所要输入进的变量ID，变量ID可以在编辑器里面直接找到。
- maxDigits：输入的位数。你可以发现，在事件编辑器里面，这个是最多8位的，但是用代码的话，可以想写几位就写几位。非常自由。

```
$gameMessage.setNumberInput(6,20);
```

物品选择处理（Select Item）

这个函数可以储存所选物品的物品ID，前提是你必须拥有这个物品。虽然我从来没有用过，但是我想这个功能还是很有用的。

参数

- variableId：要把物品ID存入哪个变量中，变量ID可以在编辑器里面直接找到。
- itemType：物品的类型，一共有四种数值，而且是从1开始的。真搞不懂官方怎么设计的。
    - 1：普通物品
    - 2：重要物品
    - 3：隐藏物品A
    - 4：隐藏物品B

```
$gameMessage.setItemChoice(5,1);
```

显示滚动文字

参数：

- speed：文字滚动的速度，2是默认值

```
$gameMessage.setScroll(2,false);
```

示例：

```
$gameMessage.add("内容1");
$gameMessage.add("内容2");
$gameMessage.add("内容3");
$gameMessage.setScroll(2,false);
```

显示滚动文字（Show Scrolling Text）

参数：

- speed：文字滚动的速度，2是默认值

```
$gameMessage.setScroll(2,false);
```

示例：

```
$gameMessage.add("内容1");
$gameMessage.add("内容2");
$gameMessage.add("内容3");
$gameMessage.setScroll(2,false);
```

游戏进程

开关操作（Control Switches）

```
$gameSwitches.setValue(variableId, value);
```

变量操作（Control Variables）

```
$gameSwitches.setValue(variableId, value);
```

独立开关操作（Control Self Switch）

需要传递四个参数

1. 地图ID，表示第几张地图
2. 变量ID
3. 独立开关名称
4. true还是false

```
$gameSelfSwitches.setValue([4,287,'A'],false)
```

查询值

```
$gameSelfSwitches.value([4,72,'A'])
```

计时器操作（Control Timer）

```
$gameTimer.start(count);//
$gameTimer.stop();
$gameTimer.seconds();//返回秒数
```

流程控制

分支条件（Conditional Branch）

在脚本中建议使用if else来代替

循环（Loop）

在脚本中建议使用for循环替代

跳出循环（Break Loop）

终止事件处理

公共事件

标签（Label）

转到标签（Jump to Label）

注释（Comment）

队伍

增减金币（Change Gold）

实际上源代码中，loseGold直接调用了gainGold(-amount)，就离谱。

```
$gameParty.gainGold(300);//得到300块钱
$gameParty.loseGold(300);//失去300块钱
```

增减物品（Change Items）

实际上源代码还是比较复杂的，我这个把源代码提炼精简了一下，你只管放心用。

参数

- item：需要传递一个物品对象，可以直接传递`$dataItems[1]`这种格式。代表第一个物品，注意这些`$`开头的都是从1开始的。
- amount：得到的数量

示例：

```
$gameParty.gainItem($dataItems[1], 30);//得到30个1号物品
$gameParty.loseItem($dataItems[1], 10);//失去10个1号物品
```

增减武器（Change Weapons）

```
 $gameParty.gainItem($dataWeapons[this._params[0]], value,
this._params[4]);
```

增减护甲（Change Armors）

```
$gameParty.gainItem($dataArmors[this._params[0]], value,
this._params[4]);
```

队伍管理（Change Party Member）

注意！如果你队伍中已经有了1号角色，那么再调用增加1号角色的代码将不会有效果（你应该庆幸没有报bug）。

```
$gameParty.addActor(1);//为队伍增加1号角色
$gameParty.removeActor(1);//1号角色离队
```

获取领队信息*

```
$gameParty.leader();
```

角色

增减HP（Change HP）

```
Game_Interpreter.prototype.changeHp(target, value, allowDeath)

$gameParty.leader().gainHp(-30)
```

示例：

```
Game_Interpreter.prototype.changeHp($gameParty.leader(),-300,true)
```

增减MP（Change MP）

```
$gameParty.leader().gainMp(-30)
```

增减TP（Change TP）

```
$gameParty.leader().gainTp(-30)
```

更改状态（Change State）

```
   actor.addState(this._params[3]);
actor.removeState(this._params[3]);
```

完全恢复（Recover All）

```
actor.recoverAll();
```

增减经验值（Change EXP）

```
 actor.changeExp(exp, show);//show代表是否展示升级信息，exp代表经验值
```

增减等级（Change Level）

```
 actor.changeLevel(level, show);//同上
```

增减能力值（Change Parameter）

增减技能（Change Skill）

更改装备（Change Equipment）

更改名字（Change Name）

```
$gameParty.leader().setName("起的名字");
$gameParty.leader().setName(prompt("自己起一个名字吧") );
```

更改职业（Change Class）

更改昵称（Change Nickname）

```
actor.setNickname(nickname);
```

更改简介（Change Profile）

移动

场所移动（Transfer Player）

第一种就是直接移动，但是画面不会移动。

```
$gamePlayer.setPosition(23,24)
```

第二种就是编辑器里面那个

```
$gamePlayer.reserveTransfer(mapId, x, y, d, fadeType);
//d代表direction，默认写0.fadetype也写0
$gamePlayer.reserveTransfer(3,33,34,0,0)
```

fadeType:

1. 0：黑
2. 1：白

3. 2：无

设置载具位置（Set Vehicle Location）

设置事件位置（Set Event Location）

滚动地图（Scroll Map）

设置移动路线（Set Movement Route）

```
$gameMap._events[event.id].moveStraight(2);//向下移动

$gameMap._events[event.id].moveStraight(4);//向左移动



this.setThrough(true); //开启穿透

this.setThrough(false); //关闭穿透

this.setTransparent(true); //开启透明状态

this.setTransparent(false);//关闭透明状态

this.turnRandom(); //随机转向

this.turnTowardPlayer(); //朝向玩家

this.turnAwayFromPlayer(); //背向玩家



this.moveTowardPlayer();//朝向玩家移动

this.moveAwayFromPlayer();//远离玩家移动

this.setMoveFrequency(3); //移动频率

$gamePlayer.setMoveSpeed(4); //角色移动速度
//4是默认速度，要注意，这个速板是指数增加的，一旦变了一点点，速度就有极大变化。所以每次增加0.01之类的比较好，如果直接加1速度会太快。



this.moveDiagonally(4, 2);//左下移动

```

```
41
42  this.moveDiagonally(6, 2);//右下移动
43
44
45  this.moveDiagonally();//左上移动
46
47
48  this.moveDiagonally(6, 8);//右上移动
49
50
51
52
53
54
55
56
57  switch (command.code) {
58      case gc.ROUTE_END:
59          this.processRouteEnd();
60          break;
61
62      case gc.ROUTE_MOVE_RIGHT:
63          this.moveStraight(6);//向右移动
64          break;
65      case gc.ROUTE_MOVE_UP:
66          this.moveStraight(8);//向上移动
67          break;
68
69      case gc.ROUTE_MOVE_RANDOM:
70          this.moveRandom();//随机移动
71          break;
72
73      case gc.ROUTE_MOVE_FORWARD:
74          this.moveForward();
75          break;
76      case gc.ROUTE_MOVE_BACKWARD:
77          this.moveBackward();
78          break;
79      case gc.ROUTE_JUMP:
80          this.jump(params[0], params[1]);
81          break;
82      case gc.ROUTE_WAIT:
83          this._waitCount = params[0] - 1;
84          break;
85      case gc.ROUTE_TURN_DOWN:
86          this.setDirection(2); //设置朝向下方
87          break;
88      case gc.ROUTE_TURN_LEFT:
89          this.setDirection(4);//设置朝向左方
90          break;
91      case gc.ROUTE_TURN_RIGHT:
92          this.setDirection(6);//设置朝向右方
93          break;
94      case gc.ROUTE_TURN_UP:
95          this.setDirection(8);//设置朝向上方
96          break;
97      case gc.ROUTE_TURN_90D_R:
98          this.turnRight90();
```

```
99          break;
100     case gc.ROUTE_TURN_90D_L:
101         this.turnLeft90();
102         break;
103     case gc.ROUTE_TURN_180D:
104         this.turn180();
105         break;
106     case gc.ROUTE_TURN_90D_R_L:
107         this.turnRightOrLeft90();
108         break;
109 
110 
111     case gc.ROUTE_WALK_ANIME_ON:
112         this.setWalkAnime(true);
113         break;
114     case gc.ROUTE_WALK_ANIME_OFF:
115         this.setWalkAnime(false);
116         break;
117     case gc.ROUTE_STEP_ANIME_ON:
118         this.setStepAnime(true);
119         break;
120     case gc.ROUTE_STEP_ANIME_OFF:
121         this.setStepAnime(false);
122         break;
123     case gc.ROUTE_DIR_FIX_ON:
124         this.setDirectionFix(true);
125         break;
126     case gc.ROUTE_DIR_FIX_OFF:
127         this.setDirectionFix(false);
128         break;
129 
130     case gc.ROUTE_CHANGE_IMAGE:
131         this.setImage(params[0], params[1]);
132         break;
133     case gc.ROUTE_CHANGE_OPACITY:
134         this.setOpacity(params[0]);
135         break;
136     case gc.ROUTE_CHANGE_BLEND_MODE:
137         this.setBlendMode(params[0]);
138         break;
139     case gc.ROUTE_PLAY_SE:
140         AudioManager.playSe(params[0]);
141         break;
142 
143     }
```

载具乘降（Getting On and Off Vehicles）

人物

在这里特指地图上那个人物的动画。要和角色区分开。角色是记录角色数据信息的，而人物则反映了在画面上的移动速度、动画等信息。

更改透明状态

更改队列行进

集合队列成员

显示动画（Show Animation）

```
$gamePlayer.requestAnimation(1);
```

显示气泡图标（Show Balloon Icon）

```
$gamePlayer.requestBalloon(1) //显示惊讶
```

要注意理论上来说这个id可以无限大1

有多大取决于系统中Ballon.png有多高，默认720像素，每一个小方块48像素。默认有10个，其图片还预留了5个扩展位置，开发者可以自己画，如果不够了只需要向下拓展即可。

例如多扩展48*5像素，那么就可以多画5个气泡，最多可以访问到20号气泡。

暂时消除事件

```
$gameMap.eraseEvent(this._eventId);
```

设置人物位置*

注意！这个设置不会使画面一起移动。如果想要画面也一起移动，建议使用移动中的"场所移动"。

```
$gamePlayer.setPosition(3,6)
```

查看队伍朝向*

MV中朝向被定义为4个数字

- 左：4
- 右：6
- 上：8
- 下：2

其实，，这个和小键盘的方向是一一对应的。

```
$gamePlayer.direction() //查看队伍方向
```

图片

显示图片（Show Picture）

```
$gameScreen.showPicture(this._params[0], this._params[1],
this._params[2],
        x, y, this._params[6], this._params[7], this._params[8],
this._params[9]);
```

移动图片（Move Picture）

```
$gameScreen.movePicture(this._params[0], this._params[2], x, y,
this._params[6],
        this._params[7], this._params[8], this._params[9],
this._params[10]);
```

旋转图片（Rotate Picture）

```
$gameScreen.rotatePicture(this._params[0], this._params[1]);
```

更改图片色调（Tint Picture）

```
$gameScreen.tintPicture(this._params[0], this._params[1],
this._params[2]);
```

消除图片（Erase Picture）

```
$gameScreen.erasePicture(this._params[0]);
```

清除所有图片*

```
$gameScreen.clearPictures();
```

计时

等待

要注意，这个脚本需要写在事件中

```
this.wait(duration);
```

画面

淡入淡出

```
$gameScreen.startFadeOut(duration) //淡出
$gameScreen.startFadeIn(duration) //淡入
```

更改画面色调

```
$gameScreen.startTint(tone, duration);
```

这个tone同样是一个颜色。数组各项分别为RGB+灰度。

```
//下面是官方的预设值：
$gameScreen.startTint([0, 0, 0, 0], 60); //正常
$gameScreen.startTint([-68, -68, -68, 0], 60); //黑暗
$gameScreen.startTint([34, -34, -68, 170], 60); //茶色
$gameScreen.startTint([68, -34, -34, 0], 60); //黄昏
$gameScreen.startTint([-68, -68, 0, 68], 60); //夜晚

//下面是我自己总结的
$gameScreen.startTint([50, 50, 50, 0], 60); //正午
$gameScreen.startTint([-120, -100, 0, 68], 60);//深夜
```

闪烁画面

```
$gameScreen.startFlash(color, duration);
```

颜色使用的是RGB+强度，调用时使用数组

```
$gameScreen.startFlash([255, 0, 0, 128], 8);
```

震动屏幕

默认5，5，60

```
$gameScreen.startShake(power, speed, duration);
```

设置天气（Set Weather Effect）

```
$gameScreen.changeWeather(this._params[0], this._params[1],
this._params[2]);
```

音频&视频

播放BGM（Play BGM）

注意，如果之前已经有bgm的话，这个会把之前的给挤掉。

```
var bgm = new Object();
bgm.name = "对峙";//bgm的名称
bgm.volume = 100;//响度
bgm.pitch = 100;//音调
bgm.pan = 0;//偏移
AudioManager.playBgm(bgm);
//AudioManager.playBgm(bgm,pos);//pos可以缺省
```

淡出BGM（Fadeout BGM）

参数为秒数

```
AudioManager.fadeOutBgm(duration);
```

保存BGM（Save BGM）

```
$gameSystem.saveBgm();
```

还原BGM（Resume BGM）

```
$gameSystem.replayBgm();
```

播放BGS（Play BGS）

```
 AudioManager.playBgs(this._params[0]);
```

淡出BGS（Fadeout BGS）

```
AudioManager.fadeOutBgs(this._params[0]);
```

播放ME（Play ME）

```
AudioManager.playMe(this._params[0]);
```

播放SE（Play SE）

```
var se  = new Object();
se.name = "Bell3";//se的名称
se.volume = 100;//响度
se.pitch = 100;//音调
se.pan = 0;//偏移
AudioManager.playSe(se);//播放音效
```

经过我的感觉，一些音名对照表如下。

C	D	E	F	G	A	B
40	45	50	52	58	65	75

停止SE（Stop SE）

播放影像（Play Movie）

```
Graphics.playVideo('movies/PV.webm');
```

播放系统音效*

这个所谓的系统，并不是说windows系统，而是说MV数据库里面那个系统。这个音效的顺序和MV系统里面的音保持一致。

```
SoundManager.playCursor()//光标
SoundManager.playOk()//确定
SoundManager.playCancel()//取消
SoundManager.playBuzzer()//无效
SoundManager.playEquip()//装备
SoundManager.playSave()//存档
SoundManager.playLoad() //读档
SoundManager.playBattleStart() //战斗开始
SoundManager.playEscape() //逃跑
SoundManager.playEnemyAttack() //敌人攻击
SoundManager.playEnemyDamage() //敌人受伤
SoundManager.playEnemyCollapse() //敌人消失
SoundManager.playBossCollapse1() //Boss消失1
SoundManager.playBossCollapse2() //Boss消失2
SoundManager.playActorDamage() //角色受伤
SoundManager.playActorCollapse() //角色无法战斗
SoundManager.playRecovery() //恢复
SoundManager.playMiss() //未命中
SoundManager.playEvasion() //回避
SoundManager.playMagicEvasion() //魔法回避
SoundManager.playReflection() //魔法反射
SoundManager.playShop() //商店
SoundManager.playUseItem() //使用物品
SoundManager.playUseSkill() //使用技能
```

场景控制

战斗处理（Battle Processing）

商店处理（Shop Processing）

名字输入处理（Name Input Processing）

打开菜单画面（Open Menu Screen）

```
SceneManager.push(Scene_Menu);
```

打开存档画面（Open Save Screen）

```
SceneManager.push(Scene_Save);
```

游戏结束（Game Over）

```
SceneManager.goto(Scene_Gameover);
```

返回标题画面（Return to Title Screen）

```
SceneManager.goto(Scene_Title);
```

系统设置

更改战斗BGM（Change Battle BGM）

```
 $gameSystem.setBattleBgm(this._params[0]);
```

更改胜利ME（Change Victory ME）

```
 $gameSystem.setVictoryMe(this._params[0]);
```

更改战败ME（Change Defeat ME）

```
$gameSystem.setDefeatMe(this._params[0]);
```

更改载具BGM（Change Vehicle BGM）

```
var vehicle = $gameMap.vehicle(this._params[0]);
if (vehicle) {
    vehicle.setBgm(this._params[1]);
}
```

启用/禁用存档（Change Save Access）

```
$gameSystem.disableSave();
$gameSystem.enableSave();
```

启用/禁用菜单（Change Menu Access）

```
$gameSystem.disableMenu();
$gameSystem.enableMenu();
```

启用/禁用遇敌

启用/禁用整队

更改窗口颜色（Change Window Color）

```
$gameSystem.setWindowTone(this._params[0]);
```

更改角色图像

更改载具图像

游戏时间*

```
$gameSystem.playtimeText() //游戏时间，用时分秒表示
$gameSystem.playtime() //游戏时间，用秒表示
```

显示帧数*

```
Graphics.showFps();
```

关闭游戏*

这个是直接暴力关闭窗口，不会有任何弹窗！！！慎用!

```
window.close()
```

地图

启用/禁用显示地图名称（Change Map Name Display）

```
$gameMap.enableNameDisplay();
$gameMap.disableNameDisplay();
```

更改地图图块（Change Tileset）

更改战斗背景（Change Battle Back）

更改远景（Change Parallax）

```
$gameMap.changeParallax(this._params[0], this._params[1],
        this._params[2], this._params[3], this._params[4]);
```

获取指定位置的信息（Get Location Info）

- 变量

  ```

  ```

-

地图尺寸*

```
$gameMap.height()
$gameMap.width()
```

战斗

增减敌人HP（Change Enemy HP）

```
enemy.gainHp(value)
```

增减敌人MP（Change Enemy MP）

```
enemy.gainMp(value)
```

增减敌人TP（Change Enemy TP）

```
enemy.gainTp(value)
```

更改敌人状态（Change Enemy State）

敌人完全恢复（Enemy Recover All）

敌人出现（Enemy Appear）

敌人变身（Enemy Transform）

显示战斗动画（Show Battle Animation）

强制战斗行动（Force Action）

中断战斗（Abort Battle）

获取敌人名字*

```
$gameTroop.enemyNames()
```

获取敌人对象*

```
$gameTroop._enemies[0]
```

战斗状态判断*

是否处于战斗状态

```
$gameTroop._inBattle
```

高级

脚本（Script）

请使用eval()代替

插件指令（Plugin Command）

```

function 调用插件指令(插件指令){
    var args =插件指令.split(" ");
    var command = args.shift();
    Game_Interpreter.prototype.pluginCommand(command, args);
}
//使用时，直接把需要的插件指令复制到形式参数里面就可以了
//比如说
调用插件指令（"我的插件 参数"）

```

动画*

播放动画

直接可以用事件来播放动画，动画会作用在其身上。

里面的参数只需要传递动画的编号即可

```
$gameMap._events[11].requestAnimation(1)
```

事件*

获取事件注释

```
//这个方法可以直接获得注释的内容
$dataMap.events[eventID].note

//如果注释是以对象形式写的，那么可以用meta属性进行访问
//<name:莲华><age:13>
$dataMap.events[eventID].meta.name
$dataMap.events[eventID].meta.age
```

获取指定位置的事件编号

如果指定的位置没有事件，那么返回0。

```
$gameMap.eventIdXy(x,y);
```

运行事件

可以直接用脚本来开启事件

```
$gameMap._events[11].start()
```

访问事件对象

在地图上，直接在对象的事件指令里面创建一个脚本，里面写上下面这个，就可以打印出该事件对象。

```
console.log($gameMap._events[this._eventId])
```

如果不是写在事件编辑器的话，可以指定事件ID来进行访问。

```
console.log($gameMap._events[11])
```

```

```

也可以直接根据坐标来访问

```
$gameMap.eventsXy(31, 33)
```

事件对象的移动速度

标准速度是4，也就是人物的速度。

```
$gameMap._events[11]._moveSpeed
```

鼠标*

鼠标移动

这个函数也比较特殊，并不是只要你移动了鼠标就能被检测到，而是说你必须一直按着并且移动才会true。这个其实是为触摸板设计的。

```
TouchInput.isMoved() //是否按下鼠标左键同时移动了鼠标
```

鼠标左键是否正在被按下

```
TouchInput.isPressed()  //鼠标左键是否正在被按下
```

是否切换为鼠标左键

```
TouchInput.isTriggered() //鼠标左键刚刚是否按下
```

鼠标左键是否连续按下

```
TouchInput.isRepeated() //鼠标左键是否连续点击，或者一直处于按下状态
TouchInput.isLongPressed() //鼠标左键是否一直按下
```

要注意的就是这个`isRepeated()`和`isLongPressed()`，这两个比较像，但是还是有很大区别的。如果你鼠标一直狂点，那么`isRepeated()`会返回true，但是`isLongPressed()`不会。而长按鼠标左键，两个都会返回true。

鼠标左键释放

```
TouchInput.isReleased() //鼠标左键是否释放
```

鼠标右键点击

```
TouchInput.isCancelled() //鼠标右键是否点击
```

鼠标位置

```
TouchInput.x
TouchInput.y
```

键盘*

增加WASD控制方向

```
Input.keyMapper = {
    9: 'tab',
    13: 'ok',
    16: 'shift',
    17: 'control',
    18: 'control',
    27: 'escape',
    32: 'ok',
    33: 'pageup',
    34: 'pagedown',
    37: 'left',
    38: 'up',
    39: 'right',
    40: 'down',
    45: 'escape',
    81: 'pageup',
    87: 'pagedown',
    88: 'escape',
    90: 'ok',
    96: 'escape',
    98: 'down',
    100: 'left',
    102: 'right',
    104: 'up',
    120: 'debug',
    //WASD的按键
    65: 'left',     // 左箭头键
    87: 'up',       // 上箭头键
    68: 'right',    // 右箭头键
    83: 'down'     // 下箭头键
};
```

检测键盘事件

下面这些函数要传递一个参数，也就是要检测的按键的名字，刚刚那个映射表里面都写了，直接复制就行。（注意把引号也带上）

```
Input.isPressed('ok')  //按键是否正在被按下
Input.isTriggered() //按键刚刚是否按下
Input.isRepeated() //按键是否连续点击，或者一直处于按下状态
Input.isLongPressed() //按键是否一直按下
```

这个和上面的鼠标一模一样，不多做解释。

存档*

载入存档

-1：载入配置文件

0：载入第一个文件

```
StorageManager.load(savefileId);//返回一个json文件
```

载入全局配置

```
DataManager.loadGlobalInfo();//返回一个数组，从1开始。
```

武器与装备*

查看武器信息

```
$dataWeapons[$gameParty.leader()._equips[0]._itemId].name
```

获取装备信息*

```
$gameParty.leader()._equips//获取领队的装备数据
```

这个会返回一个数组，里面有5个元素，分别对应游戏里面的装备信息

- 0：武器
- 1：盾牌
- 2：头部
- 3：身体
- 4：装饰品

但是这个的信息非常简陋，里面只存放了武器和装备的编号而已，并没有真正地存放物品信息。所以如果想获取真正的数据，还得去访问$dataWeapons。

```
$dataWeapons[$gameParty.leader()._equips[0]._itemId]
```

这样子，先获取玩家的装备编号，再进行查询。然后就好了。

这个$dataWeapons的数据如下：

属性	说明
animationId	动画编号
description	说明
etypeId	
iconIndex	图标编号
id	武器的编号
maxItem	最大持有量
meta	注释的JSON
name	武器名字
note	备注
params	武器数据的数组，从0到7分别是[0:最大HP，1:最大MP，2:攻击力，3:防御力，4:魔法攻击，5:魔法防御，6:敏捷，7:幸运]
price	价格
traits	特性，比如追加能力什么的
wtypeId	武器类型的编号

另外，装备里面的备注是可以直接用对象的形式访问的。

```
$dataWeapons[$gameParty.leader()._equips[0]._itemId].meta.你写的标签属性
```

nw.js脚本API速查

窗口

> `Window` 是DOM中 顶层 `window`对象的包装类 . 可扩展操作以及接收窗口事件 .
>
> 每个 `Window`都是EventEmitter类实例 , 使用 `Window.on(...)`可响应窗口事件 .

获取当前窗口

有两种等价写法

```
var win = require('nw.gui').Window.get();
var win2 = nw.Window.get();
```

但是实际上`win===win2`，因为窗口是一个非常重要的对象，所以这个是一个单例，不管是谁，获取到的对象都是一个。

打开新窗口

一共三个参数，其中，第二个是配置信息，详情参考配置中的窗口子字段。

```
nw.Window.open('test.html', {}, function (new_win) {
    //新窗口创建时的回调函数
    new_win.on('focus', function () {
        console.log('点击了新窗口！');
    });
});
```

窗口的位置

属性

窗口相对于显示器的位置

```
nw.Window.get().x
nw.Window.get().y
```

通过坐标移动

```
nw.Window.get().moveTo(x, y)//移动窗口到指定位置,将窗口的左上角移动到指定的坐标
nw.Window.get().moveBy(x, y)//移动窗口到相对于当前窗口左上角的横纵偏移,比如窗口上移10px
```

通过默认值移动

`position` String -指定的窗口位置,可选值: `null`(不固定), `center`(屏幕居中), `mouse`(鼠标所在位置)

```
nw.Window.get().setPosition('mouse')//移动窗口到鼠标所在位置
```

窗口的大小

属性

```
nw.Window.get().width
nw.Window.get().height
```

设置大小

```
nw.Window.get().resizeTo(width, height)//重设窗口尺寸为 width 和 height
nw.Window.get().resizeBy(width, height)//调整窗口尺寸为相对于当前窗口尺寸的width 和 height,比如窗口加宽10px
```

设置最大/最小值

```
nw.Window.get().setMaximumSize(width, height)//设置窗口的最大宽高
nw.Window.get().setMinimumSize(width, height)//设置窗口的最小宽高
```

是否允许玩家调整大小

```
nw.Window.get().setResizable(true)
```

聚焦窗口

就是让窗口保持最前面,

```
nw.Window.get().focus()
```

永久置顶

```
nw.Window.get().setAlwaysOnTop(false) //是否允许窗口总是置顶(在其余窗口上面)
```

控制台窗口（仅SDK模式下可用）

打开/关闭控制台

```
require('nw.gui').Window.get().showDevTools();//打开
require('nw.gui').Window.get().closeDevTools(); //关闭
```

是否打开了控制台

```
nw.Window.get().isDevToolsOpen()
```

窗口最大化/最小化

```
nw.Window.get().enterFullscreen();//全屏
nw.Window.get().leaveFullscreen() //退出全屏

nw.Window.get().minimize();//最小化

nw.Window.get().maximize()//最大化
用途:Linux和Window中最大化窗口 ， Mac中放大窗口

win.minimize()
用途:Windows/Mac中最小化窗口 ， Linux中图标化窗口
```

隐藏窗口

```
nw.Window.get().hide()
```

窗口事件和监听

最小化/最大化时

```
// 监听最小化事件
nw.Window.get().on('minimize', function() {
  console.log('最小化了! ');
});

// 移除最小化监听事件
nw.Window.get().removeAllListeners('minimize');

```

```
// 监听最大化事件
nw.Window.get().on('maximize', function() {
  console.log('最大化了！');
});

// 移除最大化监听事件
nw.Window.get().removeAllListeners('maximize');
```

窗口关闭时

```
nw.Window.get().on('close', function () {
    this.hide(); // 关闭窗口
    this.close(true); // 真正关闭窗口进程，注意一旦里面写false，就会无限循环
});

nw.Window.get().removeAllListeners('close');
```

窗口获得/失去焦点时

```
//获取焦点事件
nw.Window.get().on('focus', function () {

});

nw.Window.get().removeAllListeners('focus');
```

```
//失去焦点事件
nw.Window.get().on('blur', function () {

});

nw.Window.get().removeAllListeners('blur');
```

全屏时

```
//失去焦点事件
nw.Window.get().on('enter-fullscreen', function () {

});

nw.Window.get().removeAllListeners('enter-fullscreen');
```

调试工具被关闭时

```
//阻止游戏关闭
nw.Window.get().on('devtools-closed', function () {
    this.hide(); // 关闭窗口
    this.close(true); // 真正关闭窗口进程，注意一旦里面写false，就会无限循环
});


nw.Window.get().removeAllListeners('devtools-closed');
```

注册快捷键

```

var option = {
    key : "Ctrl+Shift+A",
    active : function() {
        console.log("发现快捷键: " + this.key + " 被激活! ");
    },
    failed : function(msg) {
        //报错信息
        console.log(msg);
    }
};

var shortcut = new nw.Shortcut(option);
nw.App.registerGlobalHotKey(shortcut);

```

下面是一些支持的快捷键

- 字母表的: `A-Z`
- 数字: `0-9`
- 功能键: `F1-F24`
- `Comma`
- `Period`
- `Tab`
- `Home` / `End` / `PageUp` / `PageDown` / `Insert` / `Delete`
- `Up` / `Down` / `Left` / `Right`
- `MediaNextTrack` / `MediaPlayPause` / `MediaPrevTrack` / `MediaStop`
- `Comma` or `,`
- `Period` or `.`
- `Tab` or `\t`
- `Backquote` or ```
- `Enter` or `\n`

- `Minus` or `-`
- `Equal` or `=`
- `Backslash` or `\`
- `Semicolon` or `;`
- `Quote` or `'`
- `BracketLeft` or `[`
- `BracketRight` or `[`
- `Escape`

打开默认应用

```
// 默认浏览器中打开指定URL
nw.Shell.openExternal('https://github.com/nwjs/nw.js');

// 默认编辑器中打开指定文件。
nw.Shell.openItem('test.txt');

// 文件管理器中显示指定文件所在目录。
nw.Shell.showItemInFolder('test.txt');
```

访问剪切板

```
// 获取系统中剪贴板对象
var clipboard = nw.Clipboard.get();

// 剪贴板中读取信息
var text = clipboard.get('text');
console.log(text);

// 向剪贴板写入
clipboard.set('I love NW.js :)', 'text');

// 清空剪贴板
clipboard.clear();
```

全局对象

```
global.a =1;
```

package.json配置速查

示例

```
{
    /**指定程序的起始页面。*/
    "main": "index.html",

    /**字符串必须是小写字母或者数字，可以包含.或者_或者-不允许带空格。name必须全局唯一。*/
    "name": "demo",

    /**程序描述*/
    "description": "demo app of node-webkit",

    /**程序版本号*/
    "version": "0.1.0",

    /**关键字*/
    "keywords": ["demo","node-webkit"],

    /**bool值，如果设置为false，将禁用webkit的node支持。*/
    "nodejs": true,
    /**
    * 指定一个node.js文件，当程序启动时，该文件会被运行，启动时间要早于node-webkit加载html的时间。
    * 它在node上下文中运行，可以用它来实现类似后台线程的功能。
    * （不需要可注释不用）
    */
    "node-main": "js/node.js",


    /**
    * bool值。默认情况下，如果将node-webkit程序打包发布，那么只能启动一个该应用的实例。
    * 如果你希望允许同时启动多个实例，将该值设置为false。
    */
    "single-instance": true,

    /**窗口属性设置 */
    "window": {
        /**字符串，设置默认title。*/
        "title": "demo",
        /**窗口的icon。*/
        "icon": "link.png",
        /**bool值。是否显示导航栏。*/
        "toolbar": false,
        /**bool值。是否允许调整窗口大小。*/
        "resizable": true,
        /**是否全屏*/
        "fullscreen": false,
        /**是否在win任务栏显示图标*/
        "show_in_taskbar": true,
        /**bool值。如果设置为false，程序将无边框显示。*/
        "frame": true,
        /**字符串。窗口打开时的位置，可以设置为“null”、“center”或者“mouse”。*/
        "position": "center",
        /**主窗口的的宽度。*/
```

```
        "width": 800,
        /**主窗口的的高度。*/
        "height": 670,
        /**窗口的最小宽度。*/
        "min_width": 400,
        /**窗口的最小高度。*/
        "min_height": 335,
        /**窗口显示的最大宽度，可不设。*/
        "max_width": 800,
        /**窗口显示的最大高度，可不设。*/
        "max_height": 670,
        /**bool值，如果设置为false，启动时窗口不可见。*/
        "show": true,
        /**是否在任务栏显示图标。*/
        "show_in_taskbar":true,
        /**
         * bool值。是否使用kiosk模式。如果使用kiosk模式，
         * 应用程序将全屏显示，并且阻止用户离开应用。
         * */
        "kiosk": false
    },

    /**webkit设置*/
    "webkit": {
        /**bool值，是否加载插件，如flash，默认值为false。*/
        "plugin": true,
        /**bool值，是否加载Java applets，默认为false。*/
        "java": false,
        /**bool值，是否启用页面缓存，默认为false。*/
        "page-cache": false
    }
}
```

RPGMaker原生配置

```
{
    "name": "",
    "main": "index.html",
    "js-flags": "--expose-gc",
    "window": {
        "title": "",
        "toolbar": false,
        "width": 816,
        "height": 624,
        "icon": "icon/icon.png"
    }
}

```

PIXIJS速查

常用代码模板速查

插件模板

```
/*:
 * @plugindesc 对插件的描述
 *
 * @author 作者
 *
 * @param 插件的第一个参数
 * @desc 描述
 * @default 默认值
 *
 * @param 插件的第一个参数
 * @desc 描述
 * @default 默认值
 *
 *
 *
 * @help
 * 下面这些是对脚本的说明，格式没有强制要求
 * 但是至少应该把使用方法和插件指令列出来。
 *
 */

//这个一般都是你的名字，但是没有强制要求，
//只要是一个对象就可以
var Yan = Yan || {};
//下面这个就是来注册插件，格式是固定的
//注意名字要和文件名一样
Yan.Parameters = PluginManager.parameters('你插件的名字');

//下面这个就是插件的参数，
Yan.Parameters.参数1 = String(Yan.Parameters['参数'] || '默认值');
Yan.Parameters.参数2 = String(Yan.Parameters['参数2'] || '默认
值');

//下面这个格式也是固定的
//注意，要是不需要插件指令的话，可以不写
var _Game_Interpreter_pluginCommand =
Game_Interpreter.prototype.pluginCommand;
Game_Interpreter.prototype.pluginCommand = function (command,
args) {
  _Game_Interpreter_pluginCommand.call(this, command, args);
  //设计你的插件指令，command就是你插件指令的名字，当然这个可以不止一个，
可以用else来创建多个指令
  if (command === '插件指令名字') {
    //参数0就是你的第一个参数
    //比如说插件指令为command 第一个参数为one
    //那么指令就是这样调用的  command one
    switch (args[0]) {
      case '某':
```

```
        //代码
        break;
      case '某某某':
        //代码
        break;
    }
  }
};
```

队伍移动

让玩家可以根据按键进行移动。

不建议直接使用，但是这个很适合重写。

```
Game_Player.prototype.moveByInput = function(){

}
```

窗体重写常用模板

自定义菜单指令

```
(() => {

    //1.自己的自定义窗体
    function Window_Test() {
        this.initialize.apply(this, arguments);
    }

    //2.继承一个窗体类，根据不同需求可以自行改动
    Window_Test.prototype =
Object.create(Window_Command.prototype);

    //3.把构造函数指向自己
```

```
    Window_Test.prototype.constructor = Window_Test;

    //4.写初始化函数
    Window_Test.prototype.initialize = function (x, y) {
        Window_Command.prototype.initialize.call(this, x, y)//xy代表位置
        this.refresh();//刷新窗口
        this.activate();//激活
    }

    //窗口类的核心方法，自定义代码就在这里写
    Window_Test.prototype.makeCommandList = function () {
        this.addCommand("自定义的指令", '指令名', true);//指令名是代码内部调用的，可以通过setHandler来使用
        //自定义代码，都在这里写。
    }


    //右侧的说明区域
    function 自定义HELP(){
        this.initialize.apply(this, arguments);
    }
    自定义HELP.prototype = Object.create(Window_Base.prototype);
    自定义HELP.prototype.constructor = 自定义HELP;

    自定义HELP.prototype.initialize = function () {
        var x = 250;
        var y = 0;
        var width = 500;
        var height = this.fittingHeight(2);//行宽
        Window_Base.prototype.initialize.call(this, x, y, width, height);
        this._text = '';
    }

    //窗口类的核心方法，自定义代码就在这里写
    自定义HELP.prototype.setInfo = function(text){
        this.contents.clear();
        this._text =text;
        this.drawTextEx(this._text, this.textPadding(), 0);
    }



    //这个就是点击指令后的场景
    function Scene_Test() {
        this.initialize.apply(this, arguments)
    }

    Scene_Test.prototype = Object.create(Scene_MenuBase.prototype);//自行查看窗体所在的场景
    Scene_Test.prototype.constructor = Scene_Test;
    Scene_Test.prototype.initialize = function () {
        Scene_MenuBase.prototype.initialize.call(this)
    }

    Scene_Test.prototype.create = function () {
```

```
        Scene_MenuBase.prototype.create.call(this)
        this.createTestWindow();
        this.createHelpWindow();
        this.bindCommands();
    }

    Scene_Test.prototype.start = function () {
        Scene_MenuBase.prototype.start.call(this)
        this.windowTest.refresh();
    }

    Scene_Test.prototype.createHelpWindow =function(){
        this._helpWindow = new 自定义HELP();
        this._helpWindow.setInfo("hello world");
        this.addWindow(this._helpWindow);
    }

    Scene_Test.prototype.createTestWindow = function () {

        var x = 0;
        var y = 0;
        // console.log(xx,yy)
        this.windowTest = new Window_Test(x, y)
        this.addWindow(this.windowTest);//根据需求，可以添加很多个窗口

    }

    Scene_Test.prototype.bindCommands = function () {
        this.windowTest.setHandler('ok', this.windowOK.bind(this));
        this.windowTest.setHandler('cancel', this.windowCancel.bind(this));
    }

    //当点击时触发
    Scene_Test.prototype.windowOK = function () {
        console.log("ok")
        this.windowTest.activate();

        // 监听按键，改变说明区域的文字
        this._helpWindow.setInfo("我是自定义的指令哟");

    }

    //当返回时除法
    Scene_Test.prototype.windowCancel = function () {
        console.log("cancel")
        SceneManager.pop();//返回上一级
    }




    Window_MenuCommand.prototype.addOriginalCommands = function () {//这个方法是专门给我们提供的，可以用来重写。
        //在菜单指令中添加指令，但是要注意，如果只写了这个，是没有效果的，
```

```
        //必须在createCommandWindow中，给test注册响应函数才能点进去
        this.addCommand('指令名', 'test', true);
    }


    Scene_Menu.prototype.commandTest = function () {
        SceneManager.push(Scene_Test);
    }


    Scene_Menu.prototype.createCommandWindow = function () {

        this._commandWindow = new Window_MenuCommand(0, 0);
        this._commandWindow.setHandler('item',
this.commandItem.bind(this));
        this._commandWindow.setHandler('skill',
this.commandPersonal.bind(this));
        this._commandWindow.setHandler('equip',
this.commandPersonal.bind(this));
        this._commandWindow.setHandler('status',
this.commandPersonal.bind(this));
        this._commandWindow.setHandler('formation',
this.commandFormation.bind(this));
        this._commandWindow.setHandler('options',
this.commandOptions.bind(this));
        this._commandWindow.setHandler('save',
this.commandSave.bind(this));
        this._commandWindow.setHandler('gameEnd',
this.commandGameEnd.bind(this));
        this._commandWindow.setHandler('cancel',
this.popScene.bind(this));
        this._commandWindow.setHandler('test',
this.commandTest.bind(this));
        this.addWindow(this._commandWindow);
    }

})()
```

状态界面

```
(function () {

    //1.创建一个函数，名字就是你自定义窗体的名字

    function 窗体名() {
        this.initialize.apply(this, arguments);
    }
    //2.继承Window_Base类
    窗体名.prototype = Object.create(Window_Base.prototype);
    窗体名.prototype.constructor = 窗体名;

    //3.初始化窗体大小
    var statusWindow = {
        x: 240,
        y: 0,
```

```
        width: 576,
        height: 624
    };
    窗体名.prototype.initialize = function () {
        Window_Base.prototype.initialize.call(this, statusWindow.x, statusWindow.y,
            statusWindow.width, statusWindow.height);
    };

    //4.写刷新函数，系统会自动调用这个函数
    窗体名.prototype.refresh = function () {
        this.contents.clear();
        this.drawText("hello world", 144, 0, 200);
    };



    //5.将窗口布局到游戏里面
    Scene_Menu.prototype.createStatusWindow = function () {
        this._statusWindow = new 窗体名();
        this.addWindow(this._statusWindow);
    };

})();
```

地图界面

```
(function () {
  //1.创建一个函数，名字就是你自定义窗体的名字
  function 窗体名() {
    this.initialize.apply(this, arguments);
  }
  //2.继承Window_Base类
  窗体名.prototype = Object.create(Window_Base.prototype);
  窗体名.prototype.constructor = 窗体名;

  //3.初始化窗体大小
  窗体名.prototype.initialize = function () {
    Window_Base.prototype.initialize.call(this, 0, 0, 100, 100);//分别代表位置和长宽
    this.drawText('莲华', 5, 5, 100, 'left');
  };

  //4.写创建窗体函数
  Scene_Base.prototype.创建窗体 = function () {
    this.addWindow(new 窗体名());
  };


  //5.创建窗体
  var _Scene_Map_createDisplayObjects = Scene_Map.prototype.createDisplayObjects;
  Scene_Map.prototype.createDisplayObjects = function () {
    _Scene_Map_createDisplayObjects.call(this);
```

```
    this.创建窗体();
  };

})();
```

自定义精灵类

```
//我们对系统提供的精灵类进行复制，创建一个"自定义精灵类"
function 自定义精灵类() {
    this.initialize.apply(this, arguments);
}
自定义精灵类.prototype = Object.create(Sprite.prototype);
自定义精灵类.prototype.constructor = 自定义精灵类;
自定义精灵类.prototype.initialize = function () {
    Sprite.prototype.initialize.call(this);
    //以上为固定写法，不需要弄清楚
    //以下是设置这个精灵类的共同属性，所有这个对象构造的精灵都具有的属性，类比C++中的类
    /*设置框架，即精灵所处矩形范围，精灵的活动区域被限制在此框架内，四个参数分别为左上角的X,Y的坐标，宽度（横），长度（竖）
    需要注意的是X和Y一般是负数，如果大于0图片就会往屏幕外移动*/
    this.setFrame(0, 0, 1000, 1500);

    //运行创造子精灵的函数
    this.createAll();

    //绘制位图（bitMap）的文字函数
    this.drawText();

    //move作用和setFrame是一样的，两个数值相当于把框架沿x轴y轴移动多少单位
    this.move(20, 10);
};

自定义精灵类.prototype.createAll = function () {
    //使用图片来建立精灵
    this._clock = new Sprite(ImageManager.loadBitmap('img/pictures/', "SF_Actor3_8", true));
    //使用系统提供的位图对象建立精灵,这里使用的是一个空位图，具体的文字会用drawText函数上描绘
    this._cont = new Sprite(new Bitmap(380, 418));        //定义位图的宽和高
    //anchor，范围0-1，设置精设置绘制精灵的起始坐标，0.5可以令其居中显示
    this._clock.anchor.x = this._clock.anchor.y = 0.5;
    /*这里的setFrame是精灵中的精灵活动范围，低级精灵不能超过其容器活动范围，不然无法显示
    _clock精灵属于mySprite类，其活动范围不能超过他的类，如果不setFrame，系统将会给一个默认值*/
    this._clock.setFrame(0, 0, 100, 100);
    this._clock.move(32, 42);
```

```
    this.addChild(this._clock);
    this.addChild(this._cont);
};

//这里定义了一个函数，是用于对前面构建_cont精灵所用位图进行描绘的
自定义精灵类.prototype.drawText = function () {
    this._cont.bitmap.fontSize = 30;    //设置字型大小
    this._cont.bitmap.textColor ="rgb(255,255,255)"||'red';  //设置字体颜色，可用RGB表示法或者英文字符
    //'' +：根据JavaScript语法规则，可以将后面的数据强制转化为字符串，这里$spriteTest.Parameters.text原本就是字符串，这样做法是为了防止出错
    var text = "测试文本 ";  //设置文本
    this._cont.bitmap.drawText(text,0, 88, 136, 44);
}


SUBupdate = Scene_Map.prototype.update;
Scene_Map.prototype.update = function () {
    SUBupdate.call(this);
    this.myMapSprite = new 自定义精灵类();
    //前面已经说过，在构建对象的函数里面已经设置了对象初始值，这里就不需要再设置了
    this.addChild(this.myMapSprite);
    //用addChild函数将精灵加入场景中
};
```

文件和文件格式速查

audio

文件分类

存放音频素材，下分4个子类

- bgm（background music）：背景音乐
- bgs（background sound）：背景音效:
- me（music effect）：音乐效果
- se（sound effect）：声音效果

音频分类

- m4a文件：用于手机端，加密后变成rpgmvm文件。
- ogg文件：用于PC端，加密后形成rpgmvo文件。

data

文件分类

- Actor.json——角色数据
- Animations.json——动画模块
- Armor.json——装备数据
- Classes.json——职业数据
- CommonEvents.json——公共事件数据
- Enemies.json——敌人数据
- Items.json——道具数据
- MapXXX.json——各地图的详细信息（包括事件
- MapInfos.json——各地图的大致信息
- Skills.json——技能数据
- States.json——状态数据
- System.json——系统、类型、用语
- Tileset.json——图块组模块
- Troop.json——敌群数据
- Weapons.json——武器数据

fonts

游戏的字体文件

1. 去根目录下fonts文件夹中
2. 打开gamefont.css文件，这个里面就存着字体的信息。

```
@font-face {
    font-family: GameFont;
    /*url里面就是字体文件的路径*/
    src: url("mplus-1m-regular.ttf");
}

.IIV::-webkit-media-controls-play-button,
video::-webkit-media-controls-start-playback-button {
    opacity: 0;
    pointer-events: none;
    width: 5px;
}
```

3. 把自己下载的ttf文件放入fonts文件夹中，把url里面的名字换成你下载的就行了

顺便一说，fonts文件都放在了C:\Windows\Fonts中，有需要可以自取。

icon

游戏的图标文件

img

游戏中所有的图片素材

js

游戏源代码，包括官方的代码和插件代码

movies

游戏中的视频文件

素材格式和尺寸

图片

素材类型	大小
立绘	高度为600px
CG	816*624px
脸图	144*144px
一个图块大小	48*48px
标题图片	816*624px

常用屏幕分辨率

- PC：16:9

  1920*1080

  1600*900

文件目录

- animations：动画特效
- battlebacks1：战斗背景图，一般都是地面或者战斗近景图
- battlebacks2：战斗的远景图，远景图可以省略，近景图必须指定。
- characters：角色的行走图，或者其他物品的动作图，大小为48*48

  在文件名前面加上半角符号“$”，那么该文件就只能容纳一个角色元。
- enemies：敌人的战斗图，也就是战斗时看到的立绘
- faces：各种角色的头像，144*144
- parallaxes：远景图，尺寸没有限制
- pictures：这些图片可以通过事件指令中的显示图片指令显示在游戏中，图片尺寸大小不限。
- titles1：标题的背景图
- titles2：标题背景图的边框，图片的大小为 816x624 ，Titles1 包含标题画面的背景，titles2 则主要是边框，使用它们可以组成标题画面。

插件脚本案例编写速查

自定义按键映射

在游戏开发中经常出现需要自己添加一个新按钮或者快捷键的需求。

实现自定义映射其实很简单，我们首先需要知道JS里面如何给一个对象增加属性。其中就有一个**对象键值对的赋值**。不懂的可以看我的JS基础。

在脚本中，左边写所映射按键的键码，右边写你给这个键起的keyName。

```
Input.keyMapper[65] = "A";
```

这样就把A，这个按键绑定到MV里面了。

具体的键码可以参考附录1。

```
Input.isPressed('A')  //按键是否正在被按下
Input.isTriggered() //按键刚刚是否按下
Input.isRepeated() //按键是否连续点击，或者一直处于按下状态
Input.isLongPressed() //按键是否一直按下
```

但是这就有一个问题，在脚本中如果直接使用if语句，会造成一些问题

禁用鼠标

```
TouchInput._onMouseDown = function(event) {
    // 什么都不要写
};
```

修改姓名输入界面的字符

```
Window_NameInput.LATIN1 =
        [ '赵','钱','孙','李','王',  '马','朱','雷','冯','叔',
          '张','闫','何','庞','郭',  '魏','田','和','依','姬',
          '白','丽','展','羽','彭',  'k','l','穆','奴','欧',
          '文','礼','月','日','芳',  '鑫','梦','世','叶','晨',
          '华','如','幸','雅','知',  '昕','微','杰','悦','翊',
          '荣','美','梅','兰','竹',  '菊','珍','镇','|','兴',
          '零','一','二','三','四',  '杨','#','$','百','己',
          '五','六','七','八','九',  '真','鉴','定','为','建',
          '雷','电','尔','这','边',  '可','以','点','换页','确认'
];
Window_NameInput.LATIN2 =
        [ '熊','大','二','砍','树',  '光','头','强','狗','猫',
          '克','苏','鲁','小','师',  '丛','雨','时','乃','宁',
          '斯','坦','纳','亚','总',  '锉','刀','枣','子','姐',
          '洛','必','达','桐','人',  '宫','保','鸡','丁','香',
          '深','见','夏','彦','爷',  '曼','珠','沙','华','ū',
          '少','中','哥','狡','滑',  '莲','花','我','老','婆',
          '奈','亚','拉','托','提',  '普','信','男','女','子',
          '帽','子','社','绊','爱',  '皮','卡','丘','冒','险',
          '萤','雪','映','月','蛇',  '人','乔','的','换页','确认'
];
Window_NameInput.RUSSIA =
        [ 'А','Б','В','Г','Д',  'а','б','в','г','д',
          'Е','Ё','Ж','З','И',  'е','ё','ж','з','и',
          'Й','К','Л','М','Н',  'й','к','л','м','н',
          'О','П','Р','С','Т',  'о','п','р','с','т',
          'У','Ф','Х','Ц','Ч',  'у','ф','х','ц','ч',
          'Ш','Щ','Ъ','Ы','Ь',  'ш','щ','ъ','ы','ь',
          'Э','Ю','Я','^','_',  'э','ю','я','%','&',
          '0','1','2','3','4',  '(',')','*','+','-',
          '5','6','7','8','9',  ':',';',' ','','OK' ];
Window_NameInput.JAPAN1 =
```

```
        [ 'あ','い','う','え','お',  'が','ぎ','ぐ','げ','ご',
          'か','き','く','け','こ',  'ざ','じ','ず','ぜ','ぞ',
          'さ','し','す','せ','そ',  'だ','ぢ','づ','で','ど',
          'た','ち','つ','て','と',  'ば','び','ぶ','べ','ぼ',
          'な','に','ぬ','ね','の',  'ぱ','ぴ','ぷ','ぺ','ぽ',
          'は','ひ','ふ','へ','ほ',  'ぁ','ぃ','ぅ','ぇ','ぉ',
          'ま','み','む','め','も',  'っ','ゃ','ゅ','ょ','ゎ',
          'や','ゆ','よ','わ','ん',  'ー','～','・','＝','☆',
          'ら','り','る','れ','ろ',  'ゔ','を','　','カナ','決定'
];
Window_NameInput.JAPAN2 =
        [ 'ア','イ','ウ','エ','オ',  'ガ','ギ','グ','ゲ','ゴ',
          'カ','キ','ク','ケ','コ',  'ザ','ジ','ズ','ゼ','ゾ',
          'サ','シ','ス','セ','ソ',  'ダ','ヂ','ヅ','デ','ド',
          'タ','チ','ツ','テ','ト',  'バ','ビ','ブ','ベ','ボ',
          'ナ','ニ','ヌ','ネ','ノ',  'パ','ピ','プ','ペ','ポ',
          'ハ','ヒ','フ','ヘ','ホ',  'ァ','ィ','ゥ','ェ','ォ',
          'マ','ミ','ム','メ','モ',  'ッ','ャ','ュ','ョ','ヮ',
          'ヤ','ユ','ヨ','ワ','ン',  'ー','～','・','＝','☆',
          'ラ','リ','ル','レ','ロ',  'ヴ','ヲ','　','英数','決定'
];
Window_NameInput.JAPAN3 =
        [ 'A','B','C','D','E',  'a','b','c','d','e',
          'F','G','H','I','J',  'f','g','h','i','j',
          'K','L','M','N','O',  'k','l','m','n','o',
          'P','Q','R','S','T',  'p','q','r','s','t',
          'U','V','W','X','Y',  'u','v','w','x','y',
          'Z','[',']','^','_',  'z','{','}','|','～',
          '0','1','2','3','4',  '!','#','$','%','&',
          '5','6','7','8','9',  '（','）','*','＋','－',
          '/','=','@','<','>',  ': ','; ','　','かな','決定'
];
```

自定义姓名输入

```
$gameParty.leader()._name =  prompt("请输入姓名！")
```

修改存档最大数量

直接修改数字。

```
DataManager.maxSavefiles = function() {
    return 你想要的数字;
};
```

自定义开始界面按钮

```
Window_TitleCommand.prototype.makeCommandList = function () {
    this.addCommand(TextManager.newGame, 'newGame');
    this.addCommand(TextManager.continue_, 'continue',
this.isContinueEnabled());
```

```
    this.addCommand(TextManager.options,   'options');

    this.addCommand("测试",   'test');
};


//重写窗体
Scene_Title.prototype.createCommandWindow = function () {
    this._commandWindow = new Window_TitleCommand();
    this._commandWindow.setHandler('newGame',
this.commandNewGame.bind(this));
    this._commandWindow.setHandler('continue',
this.commandContinue.bind(this));
    this._commandWindow.setHandler('options',
 this.commandOptions.bind(this));
    this._commandWindow.setHandler('test',
 this.test.bind(this));
    this.addWindow(this._commandWindow);
};

Scene_Title.prototype.test  = function(){
    alert("ok")
    SceneManager.push(Scene_Title);
}
```

自定义存档内容

```
DataManager.makeSavefileInfo = function() {
    var info = {};
    info.globalId   = this._globalId;
    info.title      = $dataSystem.gameTitle;
    info.characters = $gameParty.charactersForSavefile();
    info.faces      = $gameParty.facesForSavefile();
    info.playtime   = $gameSystem.playtimeText();
    info.timestamp  = Date.now();
    return info;
};
```

自定义右键菜单按钮

```
//构造函数体
function Window_PlayerCount() {
    this.initialize.apply(this, arguments);
}
//继承自Window_Base
Window_PlayerCount.prototype =
Object.create(Window_Base.prototype);
//设定构造函数
Window_PlayerCount.prototype.constructor = Window_PlayerCount;
//初始化
Window_PlayerCount.prototype.initialize = function (x, y) {
    var width = 240;
```

```
    var height = this.fittingHeight(1);
    Window_Base.prototype.initialize.call(this, x, y, width, height);
    this.drawText('团伙中有' + $gameParty.members().length + '人', 0, 0, 200);
};
//重写Scene_Menu，加入我们自定窗口
Scene_Menu.prototype.create = function () {
    Scene_MenuBase.prototype.create.call(this);
    this.createCommandWindow();
    this.createGoldWindow();
    this.createStatusWindow();
    //加入我们自己的窗口
    var win = new Window_PlayerCount(0, this._commandWindow.height);
    this.addWindow(win);
};
```

自动存档系统

RPGMV如果想要存档，在以往只能通过 **场景控制**中的**打开存档界面**进行保存。这就会产生一个很难解决的需求问题。

如何在RPGMV播放剧情的时候存档？比如在制作GalGame的时候，如何做到随时存档？

```
class StoryControl {
    //自动保存
    static autoSave() {//默认第一个存档
        $gameSystem.onBeforeSave();
        DataManager.saveGame(1)
    }
    //自动加载
    static autoLoad() {

        DataManager.loadGame(1)
    }


    //加载故事
    static loadStoryInfo() {//获取故事控制对象的属性

        //游戏每次加载的时候，首先加载存档，如果没有存档，那么就先保存再加载。

        var tempGlobalInfo = DataManager.loadGlobalInfo() || [];
        try {
            $gloablData = tempGlobalInfo[1].storyInfo;
        } catch (error) {
            //要是第一次游戏，那么递归一下，再次加载存档
            this.autoSave();
```

```
            $gloablData = this.loadStoryInfo();
        }
        return $gloablData;
    }


}



//不需要的话，注释掉这个！！！
nw.Window.get().on('close', function () {
    StoryControl.autoSave();
    this.hide(); // 关闭窗口
    this.close(true); // 关闭窗口进程
});

// //自动开始
Scene_Title.prototype.start = function () {



    StoryControl.autoLoad();

    this.fadeOutAll();
    SceneManager.clearStack();
    $gamePlayer.reserveTransfer($gameMap.mapId(), $gamePlayer.x, $gamePlayer.y);//加载自动事件
    $gamePlayer.requestMapReload();
    SceneManager.goto(Scene_Map);

}
```

当然，这里只是演示了关闭游戏时自动保存的功能。为各位做一个抛砖引玉而已。

弹幕系统

在一些创意性游戏中，弹幕可以增加氛围和趣味性。

```
/*:
* @plugindesc  弹幕系统
* @author  闫辰祥
* @help  可以发送弹幕
* 使用时可以使用默认值，比如 new Danmaku(['测试','文字'])
* 也可以详细进行配置，比如  new Danmaku(['测试','文字'], {color:"red"}),
* 也可以同时对多个弹幕进行配置，比如  new Danmaku(['测试','文字',"123"],{color:"red"},{fontSize:"100px"})
* new Danmaku(['测试','文字'],{color:"red",fontSize:"100px"})
*/
```

```
class DannmakuConfig {
    constructor(config) {
        config = config || {};
        this.time = config.time || 1;//每隔几秒发送一个弹幕
        this.speed = config.speed || 5;//弹幕速度
        this.color = config.color || 'black';//弹幕颜色
        this.fontSize = config.fontSize || '30px';//弹幕大小
    }
}

class Danmaku {
    constructor(texts, ...config) {
        let tempThis = this
        let configs = [new DannmakuConfig()]//
        for (let i = 0; i < config.length; i++) {
            configs[i]= new DannmakuConfig(config[i])
        }

        for (let i = 0; i < texts.length; i++) {
            let useconfig = {}
            if(i>=configs.length)
            {
                useconfig =  configs[configs.length-1]

            }else{
                useconfig =  configs[i]
            }

            setTimeout(function () {

               tempThis.send(texts[i],useconfig)
            }, i*useconfig.time* 1000)
        }
    }

    send(text, config) {
        let 弹幕 = document.createElement("div");
        let node = document.createTextNode(text);
        弹幕.appendChild(node);

        弹幕.style.position = "fixed";
        弹幕.style.zIndex = "20"
        弹幕.style.whiteSpace = 'nowrap '
        let height = window.innerHeight;
        let width = window.innerWidth;
        弹幕.style.top = Math.random() * height * 0.8 + "px";
        弹幕.style.left = width - 300 + "px"

        //可修改变量
        弹幕.style.color = config.color;
        弹幕.style.fontSize = config.fontSize;


        document.body.appendChild(弹幕);

```

```
        let timeControler = setInterval(() => {
            弹幕.style.left = parseInt(弹幕.style.left) -
config.speed + "px";
            if (parseInt(弹幕.style.left) < -50) {
                clearInterval(timeControler);
                弹幕.parentNode.removeChild(弹幕)
            }
        }, 10);
    }
}
```

成就系统

成就系统理论上来说并不是游戏内的一部分，不过随着游戏理论的发展，成就已经成了游戏内不可或缺的一部分。今天我就来给各位科普一下在RPGMV中，成就系统的编写原理。

在css文件夹下创建achievement.css文件

```
.achievment-node{
    background-color: rgba(0, 0, 0, 0.4);

    z-index: 10;
    position: absolute;
    left: 50%;
    top: 0%;
    transform: translate(-50%, 0);
    animation: showAchievement 2s;
    color :white;

    padding:5%;
}
.title-node{
    font-size : x-large;
}

.info-node{
    font-size: large;
}

@keyframes showAchievement {
    0% {
        top: -100px;
    }

    50% {
        top: 50px;
    }

    70% {
```

```
        top: 50px;
    }

    100% {
        top: 0px;
    }
}
```

```
/*:
 * @plugindesc  成就系统
 * @author  闫辰祥
 * @help  直接使用 例如 $gameAchievement.add("梦之始","开始游戏")
 */


(function () {
    var stylesheet = document.createElement("link")
    stylesheet.rel = "stylesheet"
    stylesheet.href = "./css/achievement.css"
    stylesheet.type = "text/css"
    document.head.appendChild(stylesheet)
})()




//游戏成就系统
var $gameAchievement = {}
$gameAchievement._achievements = [];

$gameAchievement.add = function (title, info) {

    //如果已经获得了这个成就，那么就不添加这个了
    for (let i = 0; i < $gameAchievement._achievements.length; i++) {
        if ($gameAchievement._achievements[i].title == title)
            return false;

    }
    //添加成就信息
    var achievmentRecord = { title, info }
    $gameAchievement._achievements.push(achievmentRecord);


    var achievmentNode = document.createElement("div");
    achievmentNode.classList += "achievment-node"

    var titleNode = document.createElement("div");
    titleNode.appendChild(document.createTextNode("获得成就：" + title))
    titleNode.classList += "title-node"


    var infoNode = document.createElement("div");
    infoNode.appendChild(document.createTextNode(info))
```

```
    infoNode.classList += "info-node"

    achievmentNode.appendChild(titleNode);
    achievmentNode.appendChild(infoNode);
    document.body.appendChild(achievmentNode);



    setTimeout(() => {

        achievmentNode.parentNode.removeChild(achievmentNode);

    }, 10000)

    return true;
}


```

类废都物语的鼠标点击触发事件

```
/*:
 * @plugindesc 本插件用于给一个事件添加可以鼠标左键点击的效果
 *
 * @author 闫辰祥
 *
 *
 *
 *
 * @help
 * 插件指令：
 * 事件点击 开始
 * 事件点击 关闭
 *
 * 安装好之后，用鼠标点击事件，然后事件就会被触发
 *
 * 技术群：529245311
 */


var 闫 = 闫 || {};

闫.Parameters = PluginManager.parameters('闫_鼠标触发事件');


var _Game_Interpreter_pluginCommand =
Game_Interpreter.prototype.pluginCommand;
```

```
Game_Interpreter.prototype.pluginCommand = function (command, args) {
  _Game_Interpreter_pluginCommand.call(this, command, args);

  if (command === '事件点击') {
    switch (args[0]) {
      case '开始':
        可以开始点了 = true
        break;
      case '关闭':
        可以开始点了 = false
        break;
    }
  }
};

var 可以开始点了 = false
var temp_TouchInput_onMouseDown = TouchInput._onMouseDown

TouchInput._onMouseDown = function (event) {
    temp_TouchInput_onMouseDown.call(this,event)
    if (event.button === 0 &&可以开始点了) {
        var x = Graphics.pageToCanvasX(event.pageX);
        var y = Graphics.pageToCanvasY(event.pageY);

        x = $gameMap.canvasToMapX(TouchInput.x);
        y = $gameMap.canvasToMapY(TouchInput.y);

        var event=  $gameMap.eventsXy(x,y)[0];

        if(!event)
            return

        event.start();
    }
};
```

类仙剑1的对某事件使用物体

```
/*:
 * @plugindesc  可以对某一个事件来使用道具
 * @author  闫辰祥
 * @help   首先对需要使用的事件写上注释，格式为 <name:xxx>
 * case里面的是使用道具的名称  meta.name就是对应的事件名称
 */

function itemUse(itemName) {

    var eventID = getEventID();
    if (!eventID) {
        $gameMessage.add("似乎没有什么用");
        $gameAchievement.add("虚空对线", "对着空地使用道具")
        return false;
```

```
    }

    var metaData = $dataMap.events[eventID].meta
    switch (itemName) {
        case "钥匙":
            if (metaData.name == "门") {
                if (metaData.switch) {
                    $gameSwitches.setValue(metaData.switch,
true);
                }
                if (metaData.info) {
                    $gameMessage.add(metaData.info);
                }
                return true;
            }

            break;
        case "除草镰":
            if (metaData.name == "草") {
                $gameSelfSwitches.setValue([metaData.mapID,
eventID, 'A'], true)
                let value = $gameVariables.value(1) + 1
                $gameVariables.setValue(1, value);
                $gameMessage.add("目前一共除草了" +
$gameVariables.value(1) + "个")

                if ($gameVariables.value(1) == 15)
                    $gameAchievement.add("除草狂魔", "除了15个
艹")
                if ($gameVariables.value(1) == 25)
                    $gameAchievement.add("手不累吗？", "除了所有
的草")
                return true;
            }
            else if (metaData.name == "草字头") {
                if (metaData.word == "荽") {

 $gameSelfSwitches.setValue([$gameMap._mapId, eventID, 'A'],
true)
                    return true;
                }
            }
            else if (metaData.name == "方妍") {
                $gameSelfSwitches.setValue([$gameMap._mapId,
eventID, 'A'], true)
                return true;
            }
            else if (metaData.name == "草席") {
                $gameMessage.add("（还是不要动人家东西吧，）")
                return true;
            }
            else if (metaData.name == "芳") {
                $gameSelfSwitches.setValue([$gameMap._mapId,
eventID, 'A'], true)
                return true;
            }
            else if (metaData.name == "小兰") {
                $gameMessage.add("，，，，，，")
```

```
                $gameMessage.add("你在干什么？")
                $gameMessage.add("（糟糕，她好像生气了）")
                $gameAchievement.add("见人就砍", "试图用除草镰去砍小兰")
                return true;
            }

            else if(metaData.name == "花"){
                $gameSelfSwitches.setValue([$gameMap._mapId, eventID, 'A'], true)
                $gameAchievement.add("辣手摧花","除掉了花")
                return true;
            }

            else if(metaData.name == "莲华"){
                console.log(metaData.name)
                if($gameTroop._inBattle){
                    $gameSwitches.setValue(30, true);
                }
                return true;
            }

            break;
        case "苗":
            if (metaData.name == "犬") {
                $gameSelfSwitches.setValue([$gameMap._mapId, eventID, 'A'], true)
                $gameAchievement.add("我虽然不是人，但你是真的狗", "把犬变成猫")
                return true;
            }
            break;
        case "伐木斧":
            if (metaData.name == "栅") {
                $gameSelfSwitches.setValue([$gameMap._mapId, eventID, 'A'], true)
                $gameMessage.add("砍掉了栅的木，掉落了册")
                return true;
            }

            else if (metaData.name == "栏") {
                $gameSelfSwitches.setValue([$gameMap._mapId, eventID, 'A'], true)
                $gameMessage.add("砍掉了栅的木，掉落了兰")
                return true;
            }
            else if (metaData.name == "樯") {
                $gameSelfSwitches.setValue([$gameMap._mapId, eventID, 'A'], true)
                $gameMessage.add("木墙被轻易地砍坏了")
                return true;
            }
            else if (metaData.name == "树") {
                $gameMessage.add("乱砍别人家树，人家会生气的。")
                $gameAchievement.add("光头强", "不分青红皂白砍别人的树")
                return true;
            }
```

```
            else if(metaData.name == "渠"){
                $gameSelfSwitches.setValue([$gameMap._mapId, eventID, 'A'], true)
                return true;
            }
            else if(metaData.name == "桌子"){
                $gameMessage.add("（如果把人家桌子砍坏了，估计会被打死吧--__--）")
                return true;
            }
            else if(metaData.name == "椅子"){
                $gameMessage.add("（为什么最近总想着去砍东西呢）")
                return true;
            }
            else if(metaData.name == "木门"){
                $gameMessage.add("门被劈开了")
                $gameSelfSwitches.setValue([$gameMap._mapId, eventID, 'A'], true)
                return true;
            }
            break;

        default:
            return false;
    }
    $gameAchievement.add("判断失败", "道具没有生效")
    $gameMessage.add("似乎没有什么用");
    return false;
}

function getEventID() {
    var vector = [0, 0];
    switch ($gamePlayer.direction()) {
        case 2:
            vector = [0, 1]
            break
        case 4:
            vector = [-1, 0]
            break
        case 6:
            vector = [1, 0]
            break
        case 8:
            vector = [0, -1]
            break
        default:
    }
    var eventPosition = {}
    eventPosition.x = $gamePlayer.x + vector[0]
    eventPosition.y = $gamePlayer.y + vector[1]
    return $gameMap.eventIdXy(eventPosition.x, eventPosition.y)
}


```

文件读写系统

```
/*:
 * @plugindesc 用于文件的读写
 *
 * @author 闫辰祥
 *
 * @param 文件
 * @desc 文件名
 * @default test
 *
 * @param 文件夹
 * @desc 保存文件的文件夹名
 * @default ./
 *
 *
 * @param 扩展名
 * @desc 文件扩展名，默认txt
 * @default txt
 *
 *
 * @param 内容
 * @desc 需要写入的内容
 * @default hello world
 *
 *
 * @help
 * MV是不支持fs模块的，但是还是需要文件读写怎么办呢？
 * 这时候就可以用这个插件了。
 *
 *
 * 其中文件夹的路径是从游戏项目根目录开始的
 * 你不需要写引号，MV会自己给你加上
 * 比如你想访问 "./data/test.txt"
 * 那么你的参数应该这样写
 *
 * - 文件：test
 * - 文件夹 ./data/
 * - 扩展名 txt
 *
 * 然后游戏的data文件夹下就会出现test.txt
 *
 * 插件指令：
 * FileAccess write //写入文件
 * FileAccess isEqualTo "某某某字符串"  //是否和文件的内容一致
 *
 * ！！！！！！！注意！！！！！！！！！！
 * 本代码是基于node环境的，
 * 也就是说如果用网页打开的话，
 * 本代码将会报错
 *
 * 很容易理解，因为浏览器要是可以随随便便读写文件的话，
 * 那么你的网络环境也太危险了。
 *
 * 所以说，调试这个插件的时候，请直接在工程里面打开游戏，
```

```
 * 不要点index.html打开游戏。
 *
 *
 *
 */
//作者：闫辰祥
var Yan = Yan || {};
Yan.Parameters = PluginManager.parameters('Yan_FileAccess');
Yan.Parameters.folder = String(Yan.Parameters['文件夹'] ||
'./');
Yan.Parameters.file = String(Yan.Parameters['文件'] || 'test');
Yan.Parameters.extension = String(Yan.Parameters['扩展名'] ||
'txt');
Yan.Parameters.content = String(Yan.Parameters['内容'] ||
'hello world');


var _Game_Interpreter_pluginCommand =
Game_Interpreter.prototype.pluginCommand;

Game_Interpreter.prototype.pluginCommand = function (command,
args) {
  _Game_Interpreter_pluginCommand.call(this, command, args);
  if (command === 'FileAccess') {
    switch (args[0]) {
      case 'write':
        write();
        break;
      case 'isEqualTo':
        isEqualTo(args[1]);
        break;
    }
  }
};


function isEqualTo(string){
  const fs = require('fs');

  let folder = Yan.Parameters.folder;
  let file = Yan.Parameters.file;
  let extension = Yan.Parameters.extension;
  let path = folder + file + "." + extension;

  fs.readFile(path,'utf-8', function (err, data) {
    if (err)
      alert("读取失败！");

    alert(data.toString()=="true");
  });
}

function write() {
  const fs = require('fs');

  let folder = Yan.Parameters.folder;
  let file = Yan.Parameters.file;
  let extension = Yan.Parameters.extension;
```

```
  let content = Yan.Parameters.content;

  content = `<!DOCTYPE html>
  <html lang="ch">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
  </head>
  <body>
    hello world
  </body>
  </html>
  `

  let path = folder + file + "." + extension;
  //writeFile会覆盖原有的内容,而且若找不到文件,会自己创建一个文件
  fs.writeFile(path, content, (err) => {
    if (err)
      alert("写入失败！");
  });
}
```

昼夜系统

```
/*:
 * @plugindesc 本游戏专用的数据脚本
 * @author 闫辰祥
 */

function Data(year, month, day) {
  this.year = year;
  this.month = month;
  this.day = day;
}

function Game_Infor_Manager(chapter, title, date) {
  this.chapter = chapter; //章节数
  this.title = title;  //标题
  this.date = date;  //日期
  //this.gamePlayedNums = 1;//周目数
}


//注意这里明显没写完，因为我懒得写
//而且我估计应该没有人能打完这么多天
Game_Infor_Manager.prototype.datePlus = function () {
  this.date.day++;
  if (this.date.day == 31) {
    this.month++;
    this.date.day = 1;
  }
```

```
}

Game_Infor_Manager.prototype.dateMinus = function () {
  this.date.day--;
  if (this.date.day == 0) {
    this.month++;
    this.date.day = 31;
  }

}

Game_Infor_Manager.prototype.dateChange = function (year, month, day) {
  this.date.day = day;
  this.date.month = month;
  this.date.year = year;
}




// 时间循环
Game_Infor_Manager.prototype.time = 0;
Game_Infor_Manager.prototype.bgmPlay = true;
Game_Infor_Manager.prototype.timeToSleep = true;

Game_Infor_Manager.prototype.timeControler = function () {



  if (this.bgmPlay == false) return;



  var bgm = new Object();
  bgm.volume = 100;//响度
  bgm.pitch = 100;//音调
  bgm.pan = 0;//偏移


  //$gameVariables._data[4]代表一天的时间
  //0代表清晨，1代表正午，2代表下午，3黄昏，4代表夜晚，5代表深夜。

  if ($gameVariables._data[6])
    this.time = $gameVariables._data[6];
  this.time += 0.4;
  $gameVariables._data[6] = this.time;



  if (this.time < 1000) {
    $gameScreen.startTint([0, 0, 0, 0], 1); //清晨
    bgm.name = "晨曦";//bgm的名称
    AudioManager.playBgm(bgm);
    $gameVariables._data[4] = 0;
    this.timeToSleep = true;
```

```
  }

  if (this.time >= 1000 && this.time < 2500) {
    $gameScreen.startTint([60, 60, 60, 0], 800); //正午
    AudioManager.fadeOutBgm(40);
    $gameVariables._data[4] = 1;

  }



  if (this.time >= 2500 && this.time < 3500) {
    $gameScreen.startTint([0, 0, 0, 0], 600); //下午
    bgm.name = "黄昏";//bgm的名称
    AudioManager.playBgm(bgm);
    $gameVariables._data[4] = 2;
  }

  if (this.time >= 3500 && this.time < 4500) {
    $gameScreen.startTint([68, -34, -34, 0], 1000); //黄昏
    $gameVariables._data[4] = 3;
    AudioManager.fadeOutBgm(30);
  }


  if (this.time >= 4500) {
    $gameScreen.startTint([-68, -68, 0, 68], 1000);  //夜晚
    bgm.name = "深夜";//bgm的名称
    AudioManager.playBgm(bgm);
    $gameVariables._data[4] = 4;
  }




  if (this.time >= 6000) {
    $gameScreen.startTint([-120, -100, 0, 68], 200); //深夜

    if (this.timeToSleep) {
      if ($gameMap._mapId == 6) {
        //如果已经在房子里面的话，不进行跳转
      } else {
        $gameMessage.add("太晚了，回去睡觉吧。");
        $gamePlayer.reserveTransfer(6, 8, 14, 0, 0);

      }
      this.timeToSleep = false;

    }
    $gameVariables._data[4] = 5;
  }

}


Game_Infor_Manager.prototype.setTime = function (num) {
  this.time = num;
  $gameVariables._data[6]=num;
```

```
}

Game_Infor_Manager.prototype.bgmClose = function () {
  this.bgmPlay = false;
}

Game_Infor_Manager.prototype.bgmOpen = function () {
  this.bgmPlay = true;
}

Game_Infor_Manager.prototype.clearItems = function () {
  for (let i = 0; i < 21; i++)
    $gameParty._items[i] = 0;
}




var $gameInfor = new Game_Infor_Manager(1, "美好的美一天", new Data(2021, 3, 15));

```

更换首页背景图

```
let leftBtn= document.createElement("div");
leftBtn.id = "leftBtn"
leftBtn.innerText ="左翻页"
leftBtn.style.fontSize = "30px"
leftBtn.style.color = "pink"
leftBtn.style.position = "fixed"
leftBtn.style.top = "500px"
leftBtn.style.left = "0px"
leftBtn.style.zIndex = 9999;
document.body.appendChild(leftBtn);



leftBtn.onclick = function(){
  $dataSystem.title1Name="CrossedSwords"
  SceneManager.goto(Scene_Title);
}



let rightBtn= document.createElement("div");
rightBtn.id = "rightBtn"
rightBtn.innerText ="右翻页"
rightBtn.style.fontSize = "30px"
rightBtn.style.color = "pink"
rightBtn.style.position = "fixed"
rightBtn.style.top = "500px"
rightBtn.style.right = "0px"
rightBtn.style.zIndex = 9999;
document.body.appendChild(rightBtn);

```

```
rightBtn.onclick = function(){
  $dataSystem.title1Name="Castle"
  SceneManager.goto(Scene_Title);
}
```

即时战斗系统

```

var Yan_util = Yan_util || {}

Yan_util.getTouchEvent = function () {
    var vector = [0, 0];
    switch ($gamePlayer.direction()) {
        case 2:
            vector = [0, 1]
            break
        case 4:
            vector = [-1, 0]
            break
        case 6:
            vector = [1, 0]
            break
        case 8:
            vector = [0, -1]
            break
        default:
    }
    var eventPosition = {}
    eventPosition.x = $gamePlayer.x + vector[0]
    eventPosition.y = $gamePlayer.y + vector[1]
    return $gameMap.eventsXy(eventPosition.x, eventPosition.y)
[0]
}









Yan_util.addMethod = function (object, name, fn) {
    var old = object[name]; object[name] = function () {
        if (fn.length == arguments.length)
            return fn.apply(this, arguments)
        else if (typeof old == 'function')
            return old.apply(this, arguments);
    };
}

Yan_util.random = function (m, n) {
    return Math.ceil(Math.random() * (n - m + 1) + m - 1);
}
```

```
48
49  /**
50   * 即时战斗系统，共分为以下几个部分：
51   * 全局初始化
52  */
53
54
55
56  var RealtimeBattleSystem = RealtimeBattleSystem || {}
57
58
59
60  // 初始化
61  RealtimeBattleSystem.init = function () {
62
63
64      nw.Window.get().maximize()//最大化
65
66
67      this.attack = true;
68
69      this._callbackQueue = [];//回调函数队列，用于事件触发
70      RealtimeBattleSystem.enemyInit()//更新敌人
71
72      RealtimeBattleSystem.bindUpdate()
73
74      RealtimeBattleSystem.UIInit()
75  }
76
77
78
79
80  RealtimeBattleSystem.getWeapon = function () {
81      let weapon =
    $dataWeapons[$gameParty.leader()._equips[0]._itemId]
82      return weapon ? weapon.name : '空手'
83  }
84
85
86  RealtimeBattleSystem.getHat = function () {
87      let armor =
    $dataArmors[$gameParty.leader()._equips[2]._itemId]
88      return armor ? armor.name : '没穿'
89  }
90
91
92
93  RealtimeBattleSystem.getArmor = function () {
94      let armor =
    $dataArmors[$gameParty.leader()._equips[3]._itemId]
95      return armor ? armor.name : '空手'
96  }
97
98
99  RealtimeBattleSystem.creatUI = function () {
100
101     // 左侧的状态栏
102     var stateBar = document.createElement('div')
```

```
    stateBar.classList += "state-bar"

    var hpBar = document.createElement('div')
    hpBar.id = "hp-bar"


    var mpBar = document.createElement('div')
    mpBar.id = "mp-bar"


    var defBar = document.createElement('div')
    defBar.id = "def-bar"


    var atkBar = document.createElement('div')
    atkBar.id = "atk-bar"




    stateBar.appendChild(hpBar);
    stateBar.appendChild(mpBar);
    stateBar.appendChild(defBar);
    stateBar.appendChild(atkBar);
    document.body.appendChild(stateBar)



    //右侧的装备栏
    var equipBar = document.createElement('div')
    equipBar.classList += "equip-bar"


    var weaponBar = document.createElement('div')
    weaponBar.innerHTML = `手持:
${RealtimeBattleSystem.getWeapon()} `


    var headBar = document.createElement('div')
    headBar.innerHTML = `头戴: ${RealtimeBattleSystem.getHat()}
`


    var clothesBar = document.createElement('div')
    clothesBar.innerHTML = `身披:
${RealtimeBattleSystem.getArmor()} `

    equipBar.appendChild(weaponBar);
    equipBar.appendChild(headBar);
    equipBar.appendChild(clothesBar);
    document.body.appendChild(equipBar)
}


RealtimeBattleSystem.updateUI = function () {

    var hpBar = document.querySelector("#hp-bar")
```

```
    hpBar.innerHTML = `生命值：${$gameParty.leader()._hp} 之于 ${$gameParty.leader().mhp}`

    var mpBar = document.querySelector("#mp-bar")
    mpBar.innerHTML = `法术值：${$gameParty.leader()._mp} 之于 ${$gameParty.leader().mmp}`

    var defBar = document.querySelector("#def-bar")
    defBar.innerHTML = `防御力：${$gameParty.leader().def}`


    var atkBar = document.querySelector("#atk-bar")
    atkBar.innerHTML = `攻击力：${$gameParty.leader().atk}`
}



RealtimeBattleSystem.普通攻击 = function () {
    //普通攻击
    if (Input.isTriggered('ok')) {

        var targetEvent = Yan_util.getTouchEvent()
        if (targetEvent == void 0)
            return
        targetEvent.hp -= $gameParty.leader().atk
        targetEvent.requestAnimation(2)

    }  //按键是否正在被按下
}


RealtimeBattleSystem.远程攻击 = function () {

}

RealtimeBattleSystem.蓄力攻击 = function () {

    //蓄力攻击
    if (Input.isLongPressed('ok')) {
        if (this._attack && $gameParty.leader()._mp >= 10) {
            this._attack = false;
            $gamePlayer.requestAnimation(1);
            $gameParty.leader().gainMp(-10)
        }
    }
    if (Input.isPressed('ok') === false) {
        this._attack = true;
    }
}



RealtimeBattleSystem.checkAttack = function () {
    this.普通攻击()
    this.蓄力攻击()
}


```

```
213
214 //判断玩家相对于事件到底在哪个方向
215 RealtimeBattleSystem.eventPositionToPlayer = function
    (gameEvent) {
216     if ($gamePlayer._realX > gameEvent._realX &&
    $gamePlayer._realY > gameEvent._realY) {
217         return "右下"
218     }
219     else if ($gamePlayer._realX > gameEvent._realX &&
    $gamePlayer._realY < gameEvent._realY) {
220         return "右上"
221     } else if ($gamePlayer._realX < gameEvent._realX &&
    $gamePlayer._realY > gameEvent._realY) {
222         return "左下"
223     } else if ($gamePlayer._realX < gameEvent._realX &&
    $gamePlayer._realY < gameEvent._realY) {
224         return "左上"
225     }
226 }
227
228
229 RealtimeBattleSystem.loadCSS = function () {
230     var stylesheet = document.createElement("link")
231     stylesheet.rel = "stylesheet"
232     stylesheet.href = "./css/realtimeBattleSystem.css"
233     stylesheet.type = "text/css"
234     document.head.appendChild(stylesheet)
235 }
236
237 RealtimeBattleSystem.UIInit = function () {
238     RealtimeBattleSystem.loadCSS()//加载UI样式
239     RealtimeBattleSystem.creatUI()//创建UI
240 }
241
242 RealtimeBattleSystem.cheackEnemyAttack = function (enemyEvent)
    {
243
244     var vector = [0, 0];
245     switch (enemyEvent._direction) {
246         case 2:
247             vector = [0, 1]
248             break
249         case 4:
250             vector = [-1, 0]
251             break
252         case 6:
253             vector = [1, 0]
254             break
255         case 8:
256             vector = [0, -1]
257             break
258         default:
259     }
260
261     if (enemyEvent._x + vector[0] == $gamePlayer._x &&
    enemyEvent._y + vector[1] == $gamePlayer._y) {
262
263         $gameParty.leader().gainHp(-enemyEvent.attack)
```

```
        $gamePlayer.requestAnimation(2)

        return true;
    }
    return false;


}


RealtimeBattleSystem.enemyInit = function () {
    RealtimeBattleSystem._enemys = []


    for (let i = 1; i < $dataMap.events.length; i++) {


        let dataEvent = $dataMap.events[i]
        let gameEvent = $gameMap._events[dataEvent.id]


        if (!dataEvent) {
            break;
        }
        if(!dataEvent.meta.type){
            break
        }


        RealtimeBattleSystem._enemys.push(gameEvent)


        if (dataEvent.meta.type == "车") {
            gameEvent.hp = 300;
            gameEvent.attack = 3;

            gameEvent.setMoveSpeed(4.3);
            setInterval(function () {
                gameEvent.moveTowardPlayer();//朝向玩家移动
                gameEvent.turnTowardPlayer();

 RealtimeBattleSystem.cheackEnemyAttack(gameEvent)

            }, 700)
        }

        if (dataEvent.meta.type == "马") {

            gameEvent.hp = 350;
            gameEvent.attack = 5;



            setInterval(function () {
                gameEvent.moveTowardPlayer();//先直线走一步

                setTimeout(function () {

```

```
                    switch (RealtimeBattleSystem.eventPositionToPlayer(gameEvent)) {
                        case "右下":
                            gameEvent.moveDiagonally(6, 2); //右下移动
                            break;
                        case "左下":
                            gameEvent.moveDiagonally(4, 2);//移动
                            break;
                        case "右上":
                            gameEvent.moveDiagonally(6, 8);//移动
                            break;
                        case "左上":
                            gameEvent.moveDiagonally(4, 8);//左上移动
                            break
                    }

                RealtimeBattleSystem.cheackEnemyAttack(gameEvent)
                }, 500)

            }, 1000)
        }

        if (dataEvent.meta.type == "象") {

            gameEvent.hp = 250;
            gameEvent.attack = 10;

            setInterval(function () {


                switch (RealtimeBattleSystem.eventPositionToPlayer(gameEvent)) {
                    case "右下":
                        gameEvent.moveDiagonally(6, 2); //右下移动
                        break;
                    case "左下":
                        gameEvent.moveDiagonally(4, 2);//移动
                        break;
                    case "右上":
                        gameEvent.moveDiagonally(6, 8);//移动
                        break;
                    case "左上":
                        gameEvent.moveDiagonally(4, 8);//左上移动
                        break
                }

                setTimeout(function () {

                    switch (RealtimeBattleSystem.eventPositionToPlayer(gameEvent)) {
                        case "右下":
```

```
                            gameEvent.moveDiagonally(6, 2); //右下移动
                            break;
                        case "左下":
                            gameEvent.moveDiagonally(4, 2);//移动
                            break;
                        case "右上":
                            gameEvent.moveDiagonally(6, 8);//移动
                            break;
                        case "左上":
                            gameEvent.moveDiagonally(4, 8);//左上移动
                            break
                    }

 RealtimeBattleSystem.cheackEnemyAttack(gameEvent)
                }, 500)
            }, 2000)
        }

        if (dataEvent.meta.type == "士") {

            gameEvent.hp = 200;
            gameEvent.attack = 30;
            setInterval(function () {
                if ($gamePlayer.x == 4 && $gamePlayer.y == 2)
                    gameEvent.moveTowardPlayer();//朝向玩家移动

 RealtimeBattleSystem.cheackEnemyAttack(gameEvent)
            }, 1000)
        }
        if (dataEvent.meta.type == "将") {
            gameEvent.hp = 50;
            gameEvent.attack = 5;
            setInterval(function () {

 RealtimeBattleSystem.cheackEnemyAttack(gameEvent)
            }, 1000)
        }
        if (dataEvent.meta.type == "砲") {
            gameEvent.hp = 200;
            gameEvent.attack = 15;
            setInterval(function () {
                gameEvent.moveTowardPlayer();//朝向玩家移动

 RealtimeBattleSystem.cheackEnemyAttack(gameEvent)
            }, 3000)
        }

        if (dataEvent.meta.type == "卒") {
            gameEvent.hp = 300;
            gameEvent.attack = 20;
            setInterval(function () {
                gameEvent.moveTowardPlayer();

 RealtimeBattleSystem.cheackEnemyAttack(gameEvent)
```

```
            }, 1000 * Yan_util.random(1, 5))
        }

    }

}



//增加事件

RealtimeBattleSystem.addEventCallback = function (func) {
    this._callbackQueue.push(func);
}


RealtimeBattleSystem.updateEventCallback = function () {
    if (this._callbackQueue.length == 0)
        return
    (this._callbackQueue.shift())();

}


RealtimeBattleSystem.update = function () {
    RealtimeBattleSystem.checkAttack()

    RealtimeBattleSystem.updateEventCallback()

    RealtimeBattleSystem.updateUI()
    RealtimeBattleSystem.updateEnemy()

}


RealtimeBattleSystem.checkWin = function () {
    return   RealtimeBattleSystem.winFlag
}


RealtimeBattleSystem.updateEnemy = function () {


    RealtimeBattleSystem.winFlag = true


    for (let i = 0; i < RealtimeBattleSystem._enemys.length;
i++) {
        if (RealtimeBattleSystem._enemys[i].hp <= 0) {

 $gameMap.eraseEvent(RealtimeBattleSystem._enemys[i]._eventId)
;
        } else {

            RealtimeBattleSystem.winFlag = false
        }
    }

```

```
}


RealtimeBattleSystem.bindUpdate = function () {
    var tempUpdate = Scene_Map.prototype.update

    Scene_Map.prototype.update = function () {
        tempUpdate.call(this)

        RealtimeBattleSystem.update()
    }
}




//动态事件

function Dynamic_Event() {
    var newId = $dataMap.events.length

    $dataMap.events[newId] = {
        "id": newId,
        "name": "子弹",
        "note": "",
        "pages": [{
            "conditions": {
                "actorId": 1,
                "actorValid": false,
                "itemId": 1,
                "itemValid": false,
                "selfSwitchCh": "A",
                "selfSwitchValid": false,
                "switch1Id": 1,
                "switch1Valid": false,
                "switch2Id": 1,
                "switch2Valid": false,
                "variableId": 1,
                "variableValid": false,
                "variableValue": 0
            },
            "directionFix": true,
            "image": {
                "tileId": 0,
                "characterName": "文字板",
                "direction": 4,
                "pattern": 2,
                "characterIndex": 4
            },
            "list": [{
                "code": 0,
                "indent": 0,
                "parameters": []
            }],
            "moveFrequency": 3,
            "moveRoute": {
```

```
                "list": [{ "code": 0, "parameters": [] }],
                "repeat": true,
                "skippable": false,
                "wait": false
            },
            "moveSpeed": 3,
            "moveType": 0,
            "priorityType": 1,
            "stepAnime": false,
            "through": false,
            "trigger": 0,
            "walkAnime": false
        }],
        "x": 6, "y": 6
    }

    $gameMap._events[newId] = new Game_Event($gameMap._mapId, newId);

    $gamePlayer.reserveTransfer(1, 8, 8, 0, 2)

    setInterval(function () {
        $gameMap._events[newId].moveRandom()
    }, 1000)
}
```

技巧速查

更换nw底层版本以及手动打包游戏

1. 将所需的文件压缩为zip文件，注意rar不行！
2. 把zip文件的后缀名改为nw
3. 把这个文件放到nw.js环境同级目录下
4. 在nw.js的目录下打开命令行，输入下面的命令

```
copy /b nw.exe+app.nw app.exe
```

注意，nw.exe就是咱nw环境的入口，名字可能会变。后面那个就是咱刚才打包的文件。名字也是随意的。

最后那个app.exe就是咱打包之后的文件。

使用RPGMAker调用Unity游戏

1. Compression Format设置为disabled

> Javascript调用c#

> C#调用JavaScript

1. Assets/Plugins/WebGl，用来放置浏览器要执行的JavaScript代码,插件扩展名为**.jslib**,将代码文件保存在此路径,当你发布时,这段代码(具体来说)将在build JS中扩展.
2. 代码

```
mergeInto(LibraryManager.library, {

  Hello: function () {
    window.alert("Hello, world!");
  },

  HelloString: function (str) {
    window.alert(Pointer_stringify(str));
  },

  PrintFloatArray: function (array, size) {
    for(var i = 0; i < size; i++)
    console.log(HEAPF32[(array >> 2) + i]);
  },

  AddNumbers: function (x, y) {
    return x + y;
  },

  StringReturnValueFunction: function () {
    var returnStr = "bla";
    var bufferSize = lengthBytesUTF8(returnStr) + 1;
    var buffer = _malloc(bufferSize);
    stringToUTF8(returnStr, buffer, bufferSize);
    return buffer;
  },

  BindWebGLTexture: function (texture) {
    GLctx.bindTexture(GLctx.TEXTURE_2D,
GL.textures[texture]);
  },

});

```

3. 使用C#调用

```
using System.Runtime.InteropServices;

[DllImport("__Internal")]
private static extern void Hello();

[DllImport("__Internal")]
private static extern void HelloString(string str);
```

```
8
9   [DllImport("__Internal")]
10  private static extern void PrintFloatArray(float[]
    array, int size);
11
12  [DllImport("__Internal")]
13  private static extern int AddNumbers(int x, int y);
14
15  [DllImport("__Internal")]
16  private static extern string
    StringReturnValueFunction();
17
18  [DllImport("__Internal")]
19  private static extern void BindWebGLTexture(int
    texture);
20
21  void Start()
22  {
23      Hello();
24
25      HelloString("This is a string.");
26
27      float[] myArray = new float[10];
28      PrintFloatArray(myArray, myArray.Length);
29
30      int result = AddNumbers(5, 7);
31      Debug.Log(result);
32
33      Debug.Log(StringReturnValueFunction());
34
35      var texture = new Texture2D(0, 0,
    TextureFormat.ARGB32, false);
36
     BindWebGLTexture((int)texture.GetNativeTexturePtr());
37  }
38
```

附录

键码映射表

- 字母与数字

按键	键码	按键	键码	按键	键码	按键	键码
A	65	J	74	S	83	1	49
B	66	K	75	T	84	2	50
C	67	L	76	U	85	3	51
D	68	M	77	V	86	4	52
E	69	N	78	W	87	5	53
F	70	O	79	X	88	6	54
G	71	P	80	Y	89	7	55
H	72	Q	81	Z	90	8	56
I	73	R	82	0	48	9	57

- 小键盘和功能键

按键	键码	按键	键码	按键	键码	按键	键码
0	96	8	104	F1	112	F7	118
1	97	9	105	F2	113	F8	119
2	98	*	106	F3	114	F9	120
3	99	+	107	F4	115	F10	121
4	100	Enter	108	F5	116	F11	122
5	101	-	109	F6	117	F12	123
6	102	.	110				
7	103	/	111				

- 控制键

按键	键码	按键	键码	按键	键码	按键	键码
BackSpace	8	Esc	27	Right Arrow	39	-_	189
Tab	9	Spacebar	32	Dw Arrow	40	.>	190
Clear	12	Page Up	33	Insert	45	/?	191
Enter	13	Page Down	34	Delete	46	`~	192
Shift	16	End	35	Num Lock	144	[{	219
Control	17	Home	36	;:	186	|	220
Alt	18	Left Arrow	37	=+	187	]}	221
Cape Lock	20	Up Arrow	38	,<	188	'"	222

- 多媒体键

按键	键码
音量加	175
音量减	174
停止	179
静音	173
浏览器	172
邮件	180
搜索	170
收藏	171

转义字符速查

控制字符	功能
\V[n]	替换为第n个变量的值。
\N[n]	替换为第 n个角色的名字。
\P[n]	替换为第n个队伍成员。
\G	替换为货币单位。
\C[n]	后方的文字显示为第n号颜色。
\I[n]	绘制第n个图标。
\{	将字体大小增加一级。
\}	将字体大小减小一级。
\\	替换为反斜杠字符。
\$	打开金钱窗口。
\.	等待1/4秒。
\	
\\	反斜杠
\!	等待按键输入。
\>	立刻显示一行内剩余的文字。
\<	取消立刻显示文字的效果。
\^	显示文本后不等待输入。

RPGMV的键盘映射

定义在rpg_core中。Input.keyMapper

MV里面键盘的按键名字和真正的键盘名字是**不一样**的，一个名字会对应多个实际按钮，所以要特别注意。

keyName	实际按钮
'tab'	tab
'ok'	enter、空格、Z
'shift'	shift
'control'	ctrl、alt
'escape'	esc、小键盘0、insert、X
'pageup'	pageup、Q
'pagedown'	pagedown、W
'left'	方向键左、小键盘4
'up'	方向键上、小键盘8
'right'	方向键右、小键盘6
'down'	方向键下、小键盘2
'debug'	F9

类继承表

场景的继承关系

- Scene_Base：所有场景的基类
    - Scene_Boot：用于初始化整个游戏
    - Scene_Title：标题界面的场景
    - Scene_Map：地图界面的场景
    - Scene_MenuBase：所有菜单类型的基类，也就是游戏中点右键那个菜单
        - Scene_Menu：主菜单的场景

          调用窗口：`Window_MenuCommand`、`Window_Gold`、`Window_MenuStatus`
        - Scene_ItemBase：技能界面和物品界面的基类
            - Scene_Item：物品界面

              调用窗口：`Window_Help`
            - Scene_Skill：技能界面
        - Scene_Equip：装备界面

        - Scene_Status：状态界面
        - Scene_Options：选项界面
        - Scene_File：储存和读取界面的基类
            - Scene_Save：储存界面
            - Scene_Load：读取界面
        - Scene_GameEnd：游戏结束界面（点击结束游戏时那个界面）
        - Scene_Shop：商店界面
        - Scene_Name：姓名输入界面
        - Scene_Debug：Debug界面
    - Scene_Battle：战斗场景
    - Scene_Gameover：游戏结束场景

窗体的继承关系

- Window_Base：所有窗体的基类
    - Window_Selectable：可以用鼠标点击和滑轮滑动的的窗体
        - Window_Command：选择指令的父类窗口
            - Window_HorzCommand：The command window for the horizontal selection format.
                - Window_ItemCategory： The window for selecting a category of items on the item and shop screens.
                - Window_EquipCommand： The window for selecting a command on the equipment screen.
                - Window_ShopCommand：展示选择买卖的商店窗口
            - Window_MenuCommand：主菜单可选指令的窗口，也就是鼠标右键之后，左边那一栏
            - Window_SkillType： The window for selecting a skill type on the skill screen.
            - Window_Options：The window for changing various settings on the options screen.
            - Window_ChoiceList：对应着事件中`显示选项`的窗口。
            - Window_PartyCommand： The window for selecting whether to fight or escape on the battle screen.
            - Window_ActorCommand：The window for selecting an actor's action on the battle screen.
            - Window_TitleCommand： The window for selecting New Game/Continue on the title screen.

        - Window_GameEnd：The window for selecting "Go to Title" on the game end screen.
- Window_MenuStatus：展示角色成员状态的窗口
    - Window_MenuActor：The window for selecting a target actor on the item and skill screens.
- Window_ItemList：The window for selecting an item on the item screen.
    - Window_EquipItem：The window for selecting an equipment item on the equipment screen.
    - Window_ShopSell：he window for selecting an item to sell on the shop screen.
    - Window_EventItem：对应着事件中`物品选择处理`的窗口
    - Window_BattleItem：The window for selecting an item to use on the battle screen.
- Window_SkillList：The window for selecting a skill on the skill screen.
    - Window_BattleSkill：The window for selecting a skill to use on the battle screen.
- Window_EquipSlot：The window for selecting an equipment slot on the equipment screen.
- Window_Status：The window for displaying full status on the status screen.
- Window_SavefileList：The window for selecting a save file on the save and load screens.
- Window_ShopBuy：The window for selecting an item to buy on the shop screen.
- Window_ShopNumber： The window for inputting quantity of items to buy or sell on the shop

  // screen.
- Window_NameInput：选择玩家姓名的窗口，里面全都是各种字母还有50音。
- Window_NumberInput：对应着事件指令`数值输入处理`的窗口。
- Window_BattleLog： The window for displaying battle progress. No frame is displayed, but it is

  // handled as a window for convenience.
- Window_BattleStatus： The window for displaying the status of party members on the battle screen.
- Window_BattleEnemy：The window for selecting a target enemy on the battle screen.
- Window_DebugRange：The window for selecting a block of switches/variables on the debug screen.

        - Window_DebugEdit：The window for displaying switches and variables on the debug screen.
    - Window_Help：描述选择物品的窗口
    - Window_Gold：展示队伍金币数量的窗口
    - Window_SkillStatus：The window for displaying the skill user's status on the skill screen.
    - Window_EquipStatus：The window for displaying parameter changes on the equipment screen.
    - Window_ShopStatus：The window for displaying number of items in possession and the actor's

      // equipment on the shop screen.
    - Window_NameEdit：编辑角色姓名的窗口，特指上面那个。
    - Window_Message：展示文字信息的窗口
    - Window_ScrollText：The window for displaying scrolling text. No frame is displayed, but it

      // is handled as a window for convenience.
    - Window_MapName：展示地图名字的窗口

精灵的继承关系

- Sprite_Base：具有显示动画功能的精灵类
    - Sprite_Character：显示角色的精灵
    - Sprite_Battler：Sprite_Actor 和 Sprite_Enemy 的父类
        - Sprite_Actor：显示主角的精灵
        - Sprite_Enemy：显示敌人
    - Sprite_StateOverlay：The sprite for displaying an overlay image for a state.
    - Sprite_Weapon：显示武器攻击图像的精灵
    - Sprite_Balloon：显示气泡图标的精灵
- Sprite_Button：显示按钮的精灵
- Sprite_Animation：播放动画的精灵
- Sprite_Damage：显示弹出式伤害的精灵
- Sprite_StateIcon：展示状态图标的精灵
- Sprite_Picture：显示图片的精灵
- Sprite_Timer：显示时间的精灵
- Sprite_Destination：显示输入区域的精灵
- Spriteset_Base：Spriteset_Map 和 Spriteset_Battle 的父类
    - Spriteset_Map：在地图屏幕上的一组精灵
    - Spriteset_Battle：战斗屏幕的一组精灵

游戏全局对象类继承关系

- Game_Temp：用于并不会保存到存档里面的临时数据
- Game_System：系统数据
- Game_Message：就是用来显示文本或者选择的类
- Game_Switches：开关的类
- Game_Variables：变量的类
- Game_SelfSwitches：独立开关的类
- Game_Screen：用于游戏屏幕特效的类，；例如色调更改或者闪屏等
- Game_Picture：显示图片的类
- Game_Item：
- Game_Action：
- Game_ActionResult：
- Game_BattlerBase：
    - Game_Battler
        - Game_Actor
        - Game_Enemy
- Game_Actors：游戏主角们数组的封装类
- Game_Unit： Game_Party 和Game_Troop的父类
    - Game_Party ： 游戏队伍的类，里面包含着金币和道具等数据
    - Game_Troop：
- Game_Map：
- Game_CommonEvent：
- Game_CharacterBase：
    - Game_Character： Game_Player, Game_Follower, GameVehicle 和Game_Event的父类
        - Game_Player
        - Game_Follower
        - GameVehicle
        - Game_Event
- Game_Interpreter：

颜色对照图

作者信息

姓名：闫辰祥

QQ：1796655849

QQ交流群：529245311

B站教程：https://www.bilibili.com/video/BV1Vb4y1q717?spm_id_from=333.999.0.0

\n<卒>

	工作分配	备注
smell	气泡动画	
王	文字版	
Earn	动画特效	
王	女主头像	兰，莲
单	背景图	

- Be struck:play er or enemy be hurt
- brandish a sword: player attack
- Be burnt: someone be burnt
-