## 概述

**1）概述**

该文档介绍 数据更新与旧存档 之间的关系以及机理，并提供部分优化策略。

|  |
| --- |
| V3.20版本对所有插件的存储结构进行了翻新。  你可以在保存旧存档的同时，加入 新插件 或 添加/修改插件数据 了，不会报错。  （但最好不要使用 旧存档 ，如果插件涉及游戏平衡、战斗数值，最好删掉重来） |

**2）来源**

数据更新问题的万恶之源就是来自于 存档 的数据差异。

要想尽可能解决此问题，你有必要先去了解：[存储的数据问题](#_存储的数据问题) 。

**3）解决方案**

首先，[删除旧存档](#_删除旧存档) 能解决100%的问题。

但由于很多朋友的存档非常宝贵，不能删，这里作者我提供了一些优化策略：

[优化 - 空数据同步更新](#_2）优化_-_空数据同步更新)

[优化 - 关闭插件的存储功能](#_优化_-_关闭插件的存储功能)

## 数据更新问题

### 存储的数据问题

存入存档的数据，存在 同步更新问题。

举一个简单的例子。

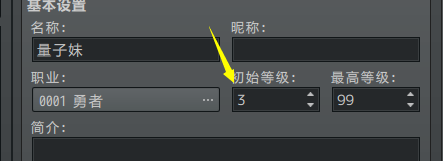
新开一个工程，配置第一个角色为量子妹：



进入游戏，保存工程。



在插件中，将初始等级修改为3，



现在新开游戏，量子妹的等级是3，

但是读取旧存档，量子妹的等级是1。

按照同样的方法，

修改量子妹的 名称、头像、经验、能力值成长 等数据，

修改后进入旧存档也都不能同步。

因为这些数据都存入了存档里，读取存档后，数据会以存档的为准。

由此可见，游戏的部分数据更新后，更新无法保证旧存档的数据完全同步。

存档中不保存真实时间、版本号等标记数据，

使得你无法辨别 ”等级3” 和 ”等级1” 哪个是最新的数据。

如果强制覆盖为 等级3，那么已经玩到等级8的存档数据改如何处理？游戏版本与存档会造成一连串的连锁问题。



所以，此问题要从根本上解决是不可能的。

（存档对于所有游戏开发者来说一直都是大问题，游戏更新造成坏档的情况屡见不鲜）

但是，我们可以根据一些特殊情况进行优化，见后面章节。

### 只读的数据

只读的数据，不存在 同步更新问题。

比如：敌人数据、物品数据、敌群数据等。

这些数据经过编辑器编辑后，在游戏中是直接读取的，不会专门去保存一份，所以这部分数据能直接实现旧存档同步更新。





|  |
| --- |
| 注意，由于数据库本身结构复杂，游戏中没有明确也无法明确标明哪些数据会存入存档，哪些数据不保存。这些数据保存与否只能依靠个人经验去试。 |

### 插件的数据问题

插件数据 与 上述的规则是一样的。

若插件数据存入了存档中，读取旧存档，会存在 同步更新问题。

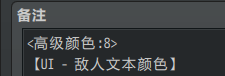
若插件数据不存，直接读取，则不存在 同步更新问题。

（插件中也无法明确告诉你哪些数据存入了存档，哪些数据是只读的。）

#### 1）正常存储的更新问题

举个例子，敌人文本颜色插件：

初始的注释是：高级颜色8。

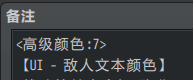


现在保存，



将改成 高级颜色改成7，（此操作，相当于在游戏中做出了新版本的改变）

新开游戏，可以看到 文本颜色变了。



但是，读取旧存档，发现颜色仍然不变。

（由于旧存档中存储的是 颜色8，所以读档后，回到了旧存档状态。）



这说明 敌人文本颜色插件 存在数据更新差异问题。

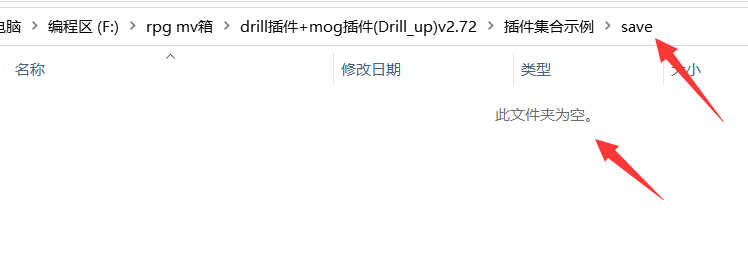
|  |
| --- |
| 此问题只在v3.20旧版本之前存在，后面都尽可能改成了只读形式，  如果你没有使用插件指令强制修改颜色，则不会出现颜色不变的问题。 |

#### 2）全局存储的更新问题

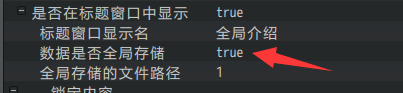
如果 插件数据 是**全局存储**的，则此问题会更加明显。

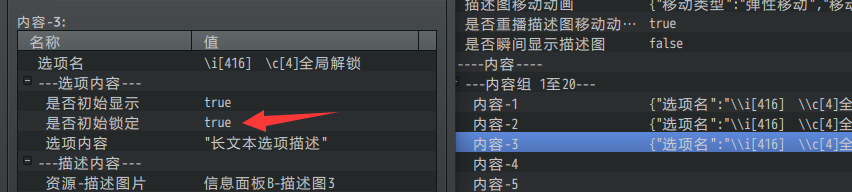
举一个简单的例子。

插件示例中，save文件夹中没有任何文件。



全自定义信息面板B中，开启了全局存储，并且，第三条数据处于锁定状态。





开始游戏，进入”初始点”，去面板管理层找到信息面板B，可以看到内容的确是锁的。

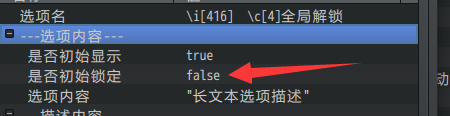


这时候看看save文件夹，已经全局存储了数据。



现在在插件中，将第三条信息解锁。

（此操作，相当于在游戏中做出了新版本的改变）



开始游戏，进入信息面板B，可以看到仍然是锁的。

（由于全局存储是实时的，开游戏后，相当于回到了旧存档状态。）



由此可见，数据更新与旧存档 之间的关系很难贴合。

必须通过一些优化策略尽可能避免这种 无法同步更新问题。

|  |
| --- |
| **旧版本的示例按上述操作会出现 不同步的问题。**  **新版本(2.80)优化了此问题，用上述操作反而不会出现 修改后 仍为锁的问题了。**  优化后，因为 面板B 的数据没有被修改过，所以默认存空值。  存档中为空值时，以 更新的数据 为准。可见：[优化 - 空数据同步更新](#_优化_-_空数据同步更新)  但是，如果你用插件指令 修改了面板B的显示/锁定 情况，此问题就会复现。  （插件指令执行后会 存非空值，非空之后 优化策略 就失效了） |

## 手动优化策略

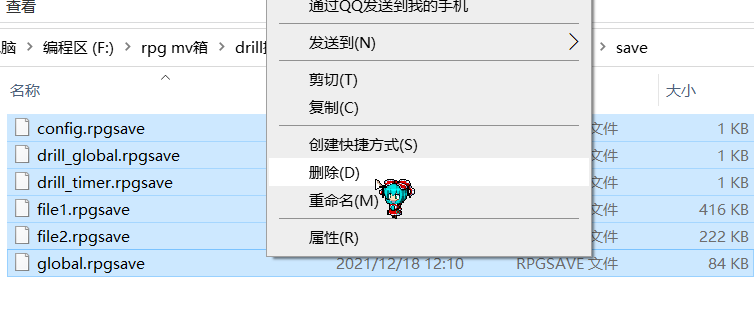
### 删除旧存档

**该方法解决100%的同步更新问题。**

删除save文件夹下的所有文件。

因为没有了存档，数据直接按最新的初始化，

不需要面对存档数据与新数据不一致的问题。



### 强制覆盖数据

**该方法非常局限，只能解决部分问题。**

比如，角色数据库中，添加了一个新角色，并且设置入队。



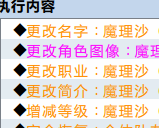
可以发现，入队后，第二个勇者的名称和脸图都是空的。

这是因为存档保存了原有的完全空白的数据，脸图、名称全都没有。



解决办法就是直接建立一个事件，强制给指定的角色重新赋值 脸图、名字 等信息。

这样就能够保存到存档并同步了。

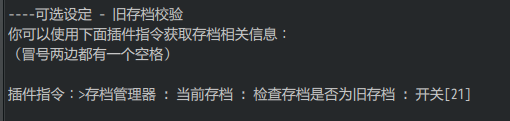


如果你使用了插件，通过调用插件相关指令强制赋值，也能解决部分 旧存档 数据不一致问题。

### 识别旧存档

存档管理器能够给你方法识别旧存档的方法：

◆Drill\_GlobalStorageManager 管理器 - 存档管理器



通过插件指令，你可以针对指定的旧存档，

给予更新提醒、或者直接执行覆盖某数据的操作。

另外，这里提一下旧存档的定义：

**旧存档：**指你对游戏工程进行 保存/部署 操作后，以前该游戏任何时间存储过的存档，都会被列为旧存档。只有用新版游戏再次存档后，旧存档标记才会消失。



也就是说，你只需要先开游戏保存一个存档，然后保存工程，重新读取那个存档，那个存档就被识别为“旧存档”了。

你可以写一个简单的用法来验证一下。



## 插件优化策略

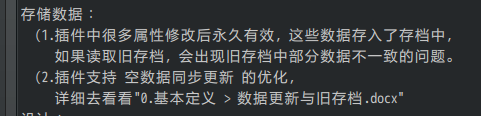
### 优化 - 空数据同步更新

|  |
| --- |
| **在3.20版本之后，作者我对所有插件都进行了 存储结构 翻新，**  **所有插件都具备此优化功能。** |

这里作者我这里分析了一些情况，对特定的存储做了优化：

如果插件数据一开始为空，并且存入了旧存档，后来数据添加上了，那么该数据能够在旧存档中正常显示。

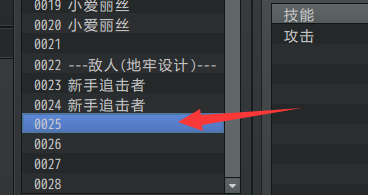
通常插件中作者我会对此优化做特定的标记，留意即可：



**注意，此优化是为了解决 旧存档带来的另一个问题 空数据 也被存储到旧存档问题。**

举个例子：

一开始，25没有添加敌人。



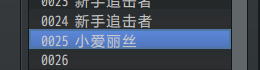
现在保存，



新版本中，添加了 小爱丽丝。

新开游戏没问题。

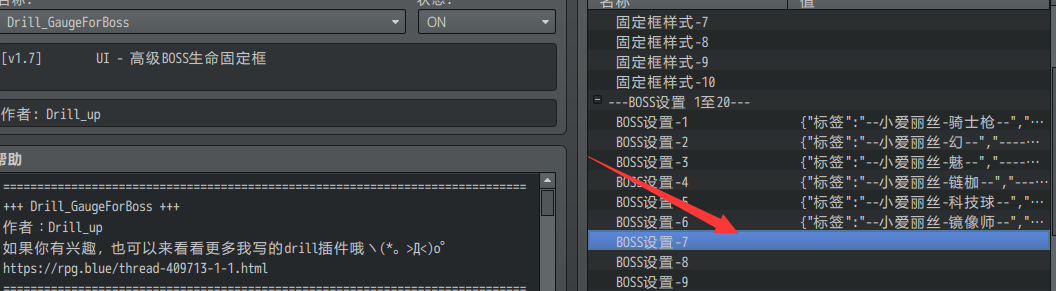
但是读取旧存档后，小爱丽丝的颜色是黑色。（因为数据为空）



优化后，新版本的 敌人文本插件 不会再出现此问题。

再举一个例子：

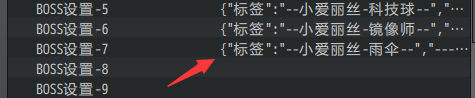
高级boss生命框中，7号位置还没设置 具体敌人BOSS的生命框属性。



保存，

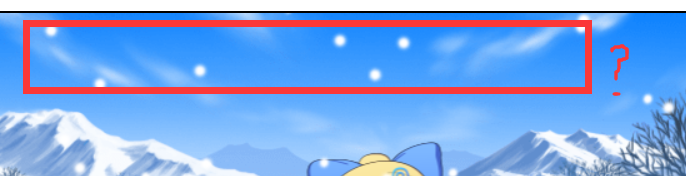


新版本中，添加了 新的boss。



新开游戏，boss能够正常显示生命框。

但是读取旧存档后，生命框根本不显示。（因为数据为空）



优化后，新版本的 高级BOSS生命框 不会再出现此问题。

|  |
| --- |
| 如果部分drill插件没有此 空数据的优化，而你的游戏对于此优化的需求比较迫切，  可以去示例里面找到群号加群，然后和作者我提哦。 |

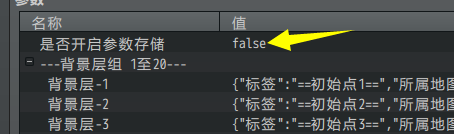
### 优化 - 关闭插件的存储功能

|  |
| --- |
| 此功能开关只有少部分插件具备，大部分插件必须存储数据才能工作。 |

部分插件提供了控制开关，

你可以直接关闭存储功能，这样就能完全避免旧存档同步问题了。

但是对应的，一些插件功能会因为无法存储而失效。



比如 多层地图背景 插件。

地图界面的所有配置数据（包括位置、速度、透明度等），都会在存档中存储。

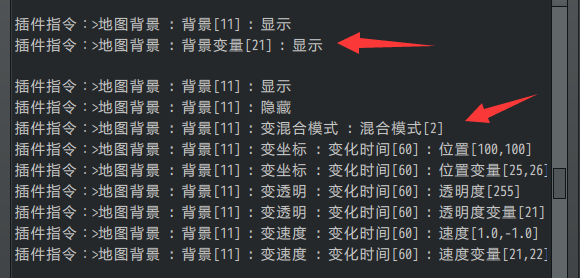
因为这些数据通过插件指令修改后，永久有效。



但是插件关闭存储后，会使得 插件指令 的改变不能永久有效。

并且，显示隐藏关系也无法保存，只能是临时效果，

你不能制作同一个地图具有多种不同背景的永久切换功能了。



## 常见问题（FAQ）

### 在编辑器中删除事件后读取旧存档出错

|  |  |
| --- | --- |
| **问题名称** | 在编辑器中删除事件后读取旧存档出错 |
| **问题图示** |  |
| **问题描述** | 如上图，有钥匙的时候保存存档。  后续再删掉钥匙事件，然后读取旧存档，会出现pages错误。 |
| **发生条件** | 1.通常设计游戏都是加事件，删事件很少，所以这个问题不容易被发现。  2.只要再加上事件，就不报错了。  3.开新游戏不会出错，只有在旧存档中存在这个问题。  4.没有加任何插件时，这样做不会出错，加了drill大部分插件后，就可能会出现这个错误。 |
| **原理解析** | 1.作者我在写插件的时候，发现游戏原脚本里面，有个重刷事件数据的函数：  Scene\_Load.prototype.reloadMapIfUpdated  这个函数会重制存储在事件中的数据，让碎片、遮罩、动画序列、临时插件指令的数据全部清空，所以作者我把它禁用了。  2.禁用之后，就会出现这个bug。  3.这个bug有修复脚本，由于执行禁用的插件很多，不能每个插件都写修复脚本。于是作者我把修复脚本放在 存档管理器插件 中了。 |
| **解决方案** | 添加插件：Drill\_GlobalStorageManager 管理器-存档管理器 就能解决。 |