目录

[一、 序 1](#_Toc13913)

[二、 帮助按钮 1](#_Toc24460)

[三、 幸运探宝 2](#_Toc17900)

[四、 东海寻仙系列(活动) 3](#_Toc25042)

[五、 巅峰战（赛程、8强、奖励） 3](#_Toc572)

[六、 观星台（已弃用） 4](#_Toc24858)

[七、 群雄榜 4](#_Toc28383)

[八、 公会红包 5](#_Toc18019)

[九、 新法宝 6](#_Toc32143)

[十、 新魂印 6](#_Toc7761)

[十一、 新编队 6](#_Toc10989)

[十二、 公会祭祀 7](#_Toc13155)

[十三、 公会十绝阵 7](#_Toc24147)

[十四、 点将台 8](#_Toc25340)

[十五、 通用弹窗 8](#_Toc3393)

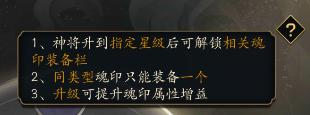
[十六、 其余 8](#_Toc26629)

1. **序**

每个功能的脚本都有注释，交接文档的内容不能完全解释功能！

待优化项是可能会有的坑，或者是代码存在可优化点，并不是真的去优化，只是起到若有因代码优化、质量引起的问题，会有一定帮助。不是必须要优化！不是必须要优化！不是必须要优化！

1. **帮助按钮**



1. **相关脚本、预设**

HelpPopup.lua；HelpPopup.prefab。

1. **使用流程**

* 策划配表：QAConfig。
* 程序配置类型：GlobalDefine.lua 配置HELP\_TYPE，注意配置参数要与QAConfig-Id索引一致。
* 在面板初始化组件（InitComponent）时，获取帮助按钮相关组件，一定要获取它的localPosition属性。
* 在绑定点击事件时创建帮助弹窗时，分别传入帮助类型，localPosition的x、y属性。

1. **待优化项**

* 由于帮助按钮位置会不确定，也就是说按钮位置相对于中轴线，在轴左侧点击，面板会显示右面板，反之亦然。但是目前只适配了x轴，弹窗默认在y轴（水平）下显示，当帮助按钮处于屏幕顶端是没问题的，但是处于屏幕底端弹窗部分内容会显示在屏幕外，所以当前打开弹窗传入的y属性是没作用的。
* 不应在创建面板时传入下x、y属性，应该只传入localPosition字段，在弹窗内部判断x、y属性。

1. **幸运探宝**

幸运探宝原来是个活动，现在是个功能。

1. **相关脚本、预设**

LuckyTurnTablePanel.lua；

LuckyTurnTableManager.lua；

LuckyTurnTablePanel.prefab。

1. **实现**

* 初始化、绑定点击事件、添加监听事件、特效层级重设都是正常的UI启动流程。
* 在OnShow时运行LuckyTurnOnShow()，判断探宝类型的显示，默认显示幸运探宝。
* SwitchTreasureType()是每个探宝具体刷新方法，功能点击事件以及红点检测。
* SetTableTurnEffect()控制转盘滚动特效（该方法比较诡异），主要是用计时器、计时器重设方法（TurnEffectReset()）与携程实现。计时器控制转盘速率、开始运转与暂停运转，携程控制多久后加速，多久后减速旋转。调整转盘速率SetTableTurnEffect()下TurnEffectReset(value)传入时间间隔。调整加减速效果在coroutine.wait(value)传入时间，具体效果需要多次调整数值。
* 剩下的方法都是配合面板刷新的具体方法。
* LuckyTurnTableManager管理器下主要是针对数据的操作，转盘的数据都是保存在这里，每天的5点刷新，免费刷新都会改变这里的数据。还包括一些请求抽取的方法，跑马灯数据，红点检测方法。
* 转盘奖品内容是后端给的数据，不是前段读表。
* 转盘奖品不是itemview

1. **待优化项**

* 面板刷新逻辑比较冗杂，两个探宝的刷新逻辑实际上是一样的，只是操作的数据不一样，先判断类型刷新，会导致目前的代码量比较冗杂，一个具体刷新方法代码里被调用了两次，实际上只是传入了不同的参数。
* 进度奖盒预览，现在做法是预览面板内嵌在转盘面板的。若想做成技能那样点击不同技能，预览面板不关闭，需要使用RewardPreviewPopupPanel.lua，具体功能看公会祭祀的奖盒预览，用的就是该预览弹窗。

1. **东海寻仙系列(活动)**
2. **相关脚本、预设**

FindFairyPanel.lua（核心面板）；

FindFairyPopup.lua（核心弹窗）；

FindFairyManager.lua（管理器）；

View文件夹（寻仙各子模块脚本）；

FindFairyPanel.prefab；FindFairyPopup.prefab。

1. **实现**

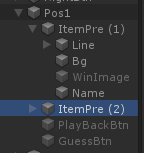
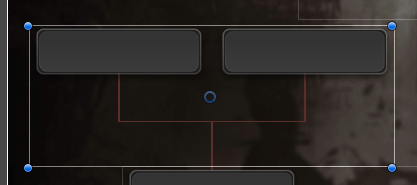
* FindFairyPanel为总入口，控制所有子模块的切换，默认显示MainView。
* 寻仙每期结束时，会切换到预告状态，面板会自动刷新，具体时间配置策划配置即可。
* 抽卡机制已经改为和招募抽卡一样，免费刷新次数的显示可能会有问题，只优化了抽卡逻辑。
* 寻仙管理器ResetItemView()这个方法要注意。Functions里有个同样的方法，当时只是寻仙活动有这个方法的需求，就没通用，后来有别的功能有同样需求，就暂时通过寻仙管理器引用了该方法（实际上不应该这么调用）。该方法目的在于优化滚动条中，上下滑动滚动条动态实例化、重设itemview，使用方法看functions下该功能注释。传递参数较多，限制的数据有限，请慎用该方法。
* 关于寻仙的所有弹窗预设脚本，都在FindFairyPopup里，根据不同功能类型显示不同内容，代码上有模块的区分。

1. **巅峰战（赛程、8强、奖励）**
2. **相关脚本、预设**

ATM\_EliminationView.Lua；ArenaTopMatchPanel.Prefab。

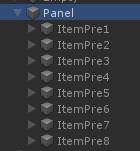
1. **实现**

* 赛程底部页签负责切换32、4强预设，左右按钮刷新每个站的每组数据。
* 每次打开赛程时，会主动请求并取得所有的32、4强数据，切战页签和切组只是从全部数据中抽取你要显示的数据内容。当巅峰战数据更新时，本地的数据会自动更新，表现也会自动刷新。
* 由于赛程表现是不规则的，不能动态实例化预设，所以预设是摆死的。但还是大体将零散的预设分组做刷新逻辑。

如图：

ABCD组每组分为7小组（Pos1~Pos7），每一小组分为ItemPre1、2。在阶段轮换时，服务器在什么阶段给了几组数据，就显示几组Pos。Pos下ItempPe显示当前玩家信息，胜负信息，切换胜负时会自动设置当前ItemPre Hierarchy层级。Pos还包含竞猜、战斗回放功能入口，根据不同阶段不同状态显示。

* 8强也是摆死的位置，因为要根据不同阶段不同位置显示数据，显示流程为8-4（8765），4-2（43），2-1（21）。



* 奖励是普通的滚动条刷新。

1. **观星台（已弃用）**
2. **相关脚本、预设**

SoulPrintAstrologyPanel.lua；SoulPrintAstrologyPanel.prefab。

1. **实现**

* 原观星台用于抽取魂印，协议为抽取魂印、强制召唤。强制召唤用于直接将星阶花钱搞到4阶段，领取4阶段的魂印奖励。
* 设置星级位移效果方法为ShowPosEffect()，主要是根据后端数据判断位置信息、与计时器结合控制。
* 观星台无管理器。
* 功能弃用了，如果有类似推星阶的动态效果，直接Copy预设代码即可。

1. **群雄榜**
2. **相关脚本、预设**

RankingManager.lua；

RankingListPanel.lua（原排行榜）；

RankingListMainPanel.lua（新）；

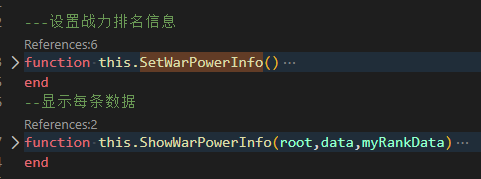
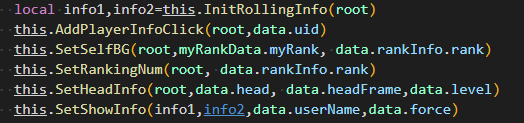
RankingSingleListPanel.lua（新）；

RankingListPanel.prefab（原排行榜）；

RankingListMainPanel.prefab（新）；

RankingSingleListPanel.prefab（新）。

1. **实现（原排行榜）**

* 由于在设计排行榜最初没有考虑到需要设计总排行榜，各排行榜都是相互独立的逻辑，各数据字段略有不同，导致在设计总排行榜时，数据比较复杂，代码量也较冗杂。
* 表现以及使用逻辑在panel，数据操作在管理器下。
* 在打开总排行榜时，根据TabBox页签请求数据，默认打开第一个页签排行榜。数据请求时会只在切换到该页签时请求，只请求一次，保证不会连续请求数据。
* GetRankInfo()方法根据页签Id，请求对应数据并执行对应表现的刷新方法，如果是分页请求，在下一页触发请求数据并拿到后，会触发刷新事件来刷新表现。
* 排行数据会在管理器维护一套数据，封装统一的字段。
* 每种排行大体刷新逻辑为
* 刷新每条数据下，是拆分的各模块表现的通用逻辑。
* 新排行榜是强哥对表现结构进行了修改，但核心的刷新流程与原排行榜大同小异。
* 新排行榜数据拆分操作。

1. **待优化项**

严格意义讲，所有的排行表现是一样的，都可以用一套刷新方法去操作面板刷新，而不是像现在这样每个排行榜有个单独的刷新方法（虽然每个刷新方法下的每条刷新方法使用的是通用逻辑）。

1. **公会红包**
2. **相关脚本、预设**

RedPacketPanel.lua（主面板）；

RedPacketPopup.lua；

RedPacket\_SendView.lua（发红包）；

RedPacket\_GetView.lua（抢红包）；

RedPacket\_RankView.lua（排行）；

GuildRedPacketManager.lua；

RedPacketPanel.prefab。

1. **实现**

* RedPacketPanel为总功能入口，控制着三个子页签逻辑的切换，默认打开发红包页签。
* 发红包核心功能为红包左右点击、滑动切换红包的效果。在InitComponent时搞到dragview，对红包预设初始化，并创建出来，添加触摸事件，设置滑动属性。
* 红包刷新逻辑是SetInfo()，该方法在打开面板时、点击左右切换红包时、拖动结束时调用。
* 抢红包、排行榜、红包弹窗都是滚动条操作。
* 红包管理器鸡儿都莫得，只有个红点检测方法（检测方法我没看懂啥意思，虽然是我写的...）。

1. **新法宝**
2. **相关脚本、预设**

RoleTalismanPanelV2.lua（主面板）；

TalismanInfoPanel.lua（进阶）；

TalismanUpStarSuccessPanel.lua（进阶成败）；

TalismanInfoPopup.lua（法宝详情弹窗）；

TalismanInfoPopup.prefab。

**2. 实现**

* 法宝主面板、进阶面板主要为自动布局设计，大量自动布局的使用，在修改预设时需要注意结构。
* 法宝天赋是一个大的Text，内容是字符串拼接操作，配合富文本标签使用来实现的效果，若是这部分需要大量改动，一个Text就不适合了，可能需要多预设的做法了。

1. **新魂印**

**1. 相关脚本、预设**

SoulPrintPanel.lua（主面板）；

SoulPrintPopupV2.lua（详情、替换弹窗）；

SoulPrintHandBook.lua（图鉴）；

SoulPrintPopUp.lua（Tip）；

SoulPrintPanel.prefab；SoulPrintPopupV2.prefab；SoulPrintHandBook.prefab； SoulPrintPopUp.prefab。

**2. 实现**

* 魂印数据是强哥设计的，主要是没有了静态ID，我还是比较懵逼的，魂印数据相关的接口由强哥编写，数据接口分布在SoulPrintManager和HeroManager下。
* 魂印主面板打开时，主要是设置环形布局、魂印列表。
* 环形布局先根据special表配置获取配置属性，配置属性和玩家等级决定魂印格子开启数量、最后一格是否锁定。确定布局格子后，再对格子进行数据赋值。值得注意的是SetLoopPre方法，它决定了设置格子的锁定、魂印等信息，以及点击事件；GetCurPosByIndex设置魂印格子的位置信息，根据配置开启的数量，动态计算位置信息。
* 魂印Tip主要是动态布局的操作、包含装备卸下等操作，具体功能脚本内有注释。
* 魂印详情和替换UE是一样的，所以做成了同脚本、预设。
* 魂印数据比较操蛋，由于是本地维护一套数据，一些功能操作都是跟服务器请求，数据是自己在本地置，比较容易出现与服务器数据不一致的问题。

**3. 待优化项**

魂印操作逻辑比较复杂，层层嵌套，比如从别的英雄上穿到当前英雄强制替换魂印的操作...操作逻辑都在对应的面板View下，这样代码耦合较高（当时着急开发也没想这么多），如果碰到问题，捋操作逻辑就能找到问题。

1. **新编队**

**1. 相关脚本、预设**

FormationPanelV2.lua；FormationPanelV2.prefab；

FormationExamplePopup.lua；FormationExamplePopup.prefab。

1. **实现**

* 编队刷新时先初始化可上阵位置，设置上阵英雄数据，同时计算战力。
* 刷新英雄列表时排序数据、设置筛选。
* 新编队通过dragview实现拖拽，每次拖拽或从英雄列表选择英雄上阵时会先计算空余上阵位置。拖拽时，会把选中的卡牌预设位移到Root层下，保证始终在其他图片上，按下抬起结束拖动具体操作看方法实现即可。
* 结束拖动时会检测拖拽物体下方的ui名称（未用标签是因为对比标签比较耗；要单独创建标签，工程里对标签的操作都是用在大模块上），切记不要随便更改已有子物体名称，切记新增的子物体RaycastTarget不必要时一定要关闭！！！否则会影响拖拽判断逻辑！！！。
* 元素共鸣逻辑修改，主要是针对配表配置进行判断。
* 编队推荐阵容，纯配表操作，里面的文字描述信息是字符串拼接，若有小的表现修改，可使用富文本。

1. **公会祭祀**
2. **相关脚本、预设**

GuildFetePopup.lua；GuildFetePopup.prefab。

1. **实现**

刷新逻辑都是普通的UI刷新，这里要说下点击进度条时的奖励预览（RewardPreviewPopupPanel.lua）。它与英雄技能实现方式一样，通过添加Canvas控制层级转换，实现点击其他预览时，预览面板不关闭。这个预览面板经过2次修改，可直接当插件使用。

1. **公会十绝阵**
2. **相关脚本、预设**

DeathPosManager.lua；DeathPosPanel.lua（主面板）；DeathPosInfoPanel.lua；DeathPosRankPopup.lua；

DeathPosRewardPopup.lua。

1. **实现**

* 十绝阵主面板进入后，各阵是根据代码里配置的位置动态实例化的，公会相关信息是打开页面、服务器推送更新数据。
* 十绝阵各阵详情编队是直接摆死的，需要进入后赋值，下方是用TabBox控制切换排行。每个阵都有个单独排行，加起来排行总数较多，后端处理麻烦，前端只需要请求不同阵的排行类型即可。
* 排行面板普通的滚动条赋值。
* 十绝阵奖励数据是登录后获取，之后数据变换需要自己维护，在Indication推送时会改变数据。
* 宝箱面板的刷新逻辑要区分是普通的刷新还是Indication推送。普通刷新获取的是登陆数据，再之后获取的数据都是Indication改变的数据。Indication推送数据只推送一条，一个optional，它会改变普通刷新的数据。
* 管理器主要维护的是宝箱数据，以及红点判断。

1. **点将台**
2. **相关脚本、预设**

RecruitPanel.lua；SingleRecruitPanel.lua；TenRecruitPanel.lua；

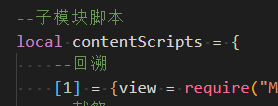
1. **实现**

* 抽卡请求的逻辑都是通用的，就是表现恶心点，当时脑子瓦特了，写的有点恶心。
* 在抽卡脚本里有个抽卡配置，区分不同的抽卡，目的是为了方便以后有新的抽卡类型的添加。
* 每个抽卡类型下会有若干功能的字段，需要什么功能直接赋值这些字段，若没有该功能，不写即可。比如Normal里有免费刷新需要检测privilegeId，而Friend的里没有该功能就不必写了。
* 每个抽卡类型下会区分不同的按钮，每个按钮的表现也会稍许不同，也需要看具体的配置内容。
* 刷新面板时遍历该配置，做一些判断，具体功能需要看代码。
* 恶心的是SingleRecruitPanel，抽卡时的动画表现，以及再来一次都在这里。动画表现是计时器驱动的，一些时间配置是在TenOpenPanel()方法里配置的。若抽卡表现有BUG，去该脚本查原因即可。
* 抽卡管理器的RecruitRequest()，通用的抽卡方法，寻仙、元素、秘盒、抽卡。

1. **通用弹窗**
2. **相关脚本、预设**

GeneralPopup.lua；View/各弹窗的子模块；GeneralPopup.prefab。

1. **实现**

* 通用弹窗是把类似界面较为复杂的弹窗集合。
* 通用弹窗是把不同种弹窗整合到一个预设下，拆分不同脚本的子模块，控制每一个弹窗的显示。
* GeneralPopup.lua是弹窗的主入口，下面引用了所有弹窗脚本，需要添加新的弹窗，需要在这里配置。
* 每个弹窗的开关都是自动控制的，只需要在打开弹窗面板时，传入对应的弹窗类型即可。弹窗类型也需要对应配置。

1. **其余**

百宝商会，有预设、脚本，但是没拼逻辑，UI效果图、资源也都做进预设。

回溯献祭当时和强哥做的，使用的通用弹窗，强哥写的献祭奖励的计算。

森罗幻境排行使用的是之前的，没按案子里修改。

暂时想起来的就是这些...

路漫漫其修远兮 吾将上下而求索

