MA80 开发笔记

郑华

2018年8月1日

第一章 架构

1.1 Canvas 图层架构

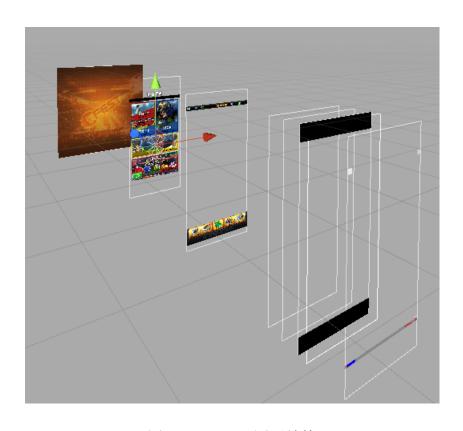
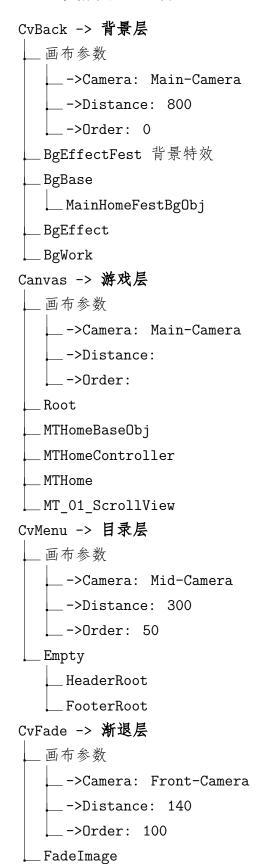


图 1.1: Canvas 图层结构

在本项目中的所有画布的渲染模式均为 Screen-Camera 模式。

1.1.1 图层结构组件



CvOverlap -> 交叠层
画布参数
>Camera: Front-Camera
>Distance: 120
>0rder: 150
Empty
CvOverlap2 -> 交叠层 2
画布参数
>Camera: Front-Camera
>Distance: 90
>0rder: 180
CvFore -> 黑框层
画布参数
>Camera: Front-Camera
>Distance: 100
>Order: 900
InputPanel
top bottom Bg
left rightBg
NewSpinner
DownloadingView
CvOverlapFull -> 网页层
画布参数
>Camera: Front-Camera
->Distance: 90
>0rder: 180
WebViewTarget
WebViewObject
ImageBgTapArea
ImageBg
CvDebug -> 调试层
■ 画布参数
>Camera: Front-Camera
>Distance: 50
>0rder: 999
DebugViewButton
BackKeyButton
DebugDispButton
DebugPerfMeter

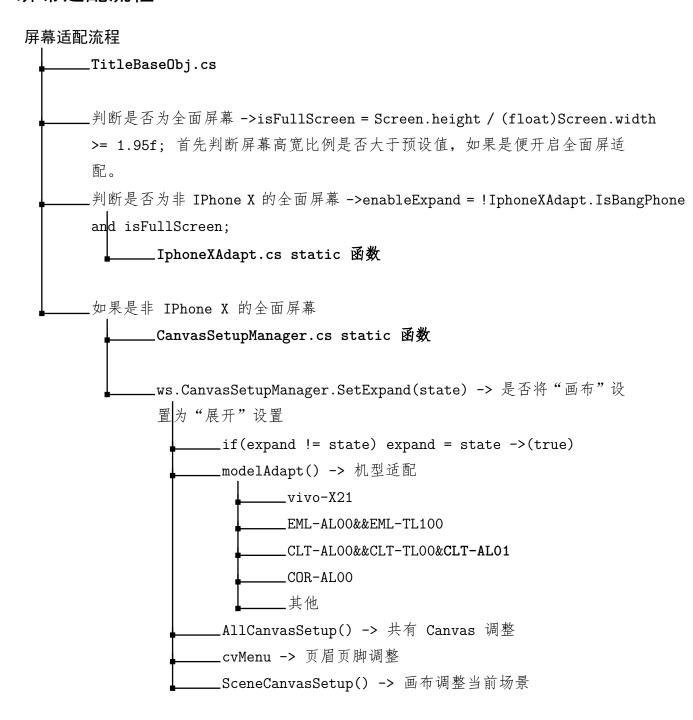
1.1.2 图层渲染顺序说明

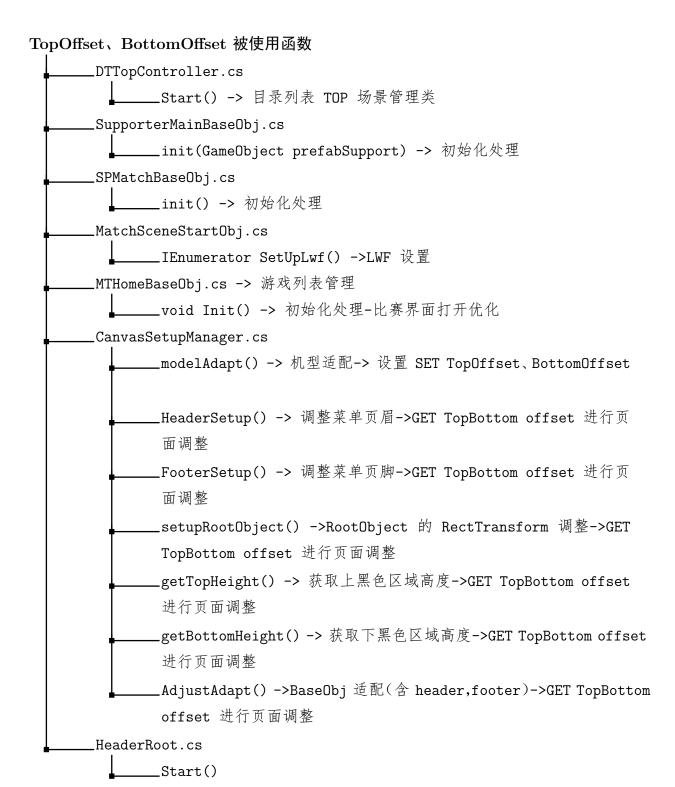
这个只针对在一个同一个相机下渲染的情况。在本项目中只针对Front-Camera。渲染顺序如下所示:

- 1. CvFade 渐退层: 100
- 2. CvOverlap 覆盖层: 150
- 3. CvOverlap2 覆盖层 2: 180
- 4. CvOverlapFull 网页层: 180
- 5. CvBug 调试层: 999

第二章 流程

2.1 屏幕适配流程





modelAdapt() 具体流程细节

HeaderSetup(canvas, obj1) 具体流程细节

if (GetExpand ()) -> 是否将"画布"设置为"expand"设置

target = targetSetup(target, obj2); -> 在目标对象 target
上获取 obj2 对象,没有创建并将 target 设置为其父节点,并返回

objectSetParent (target, obj); -> 将 target 设置为 obj1 的父节点
addFooterImage () -> 添加页脚背景
obj.GetComponent<RectTransform> ().offsetMin = new Vector2 (0.0f, 0.0f - BottomOffset); -> 重新设置组件的矩形大小与位置

getTopHeight()

rate = Screen.width / 720.0f

height = Screen.height / rate

if (height > 1280)

height = GetExpand() ? (height - 1280)/2*rate-BottomOffset*rate

: 0

else
height = 0

AdjustAdapt(GameObject go)

if(go != null)

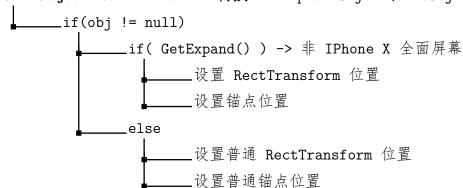
设置 Header RectTransform 位置 ->go.transform.Find("Header")

.GetComponent<RectTransform>().offsetMax

设置 Footer RectTransform 位置 ->go.transform.Find("Footer")

.GetComponent<RectTransform>().offsetMax

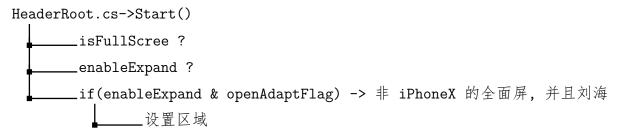
RootObject RectTransform 转换 setupRootObject(GameObject obj)



去掉黑色填补区域,并使用背景填充 ->

适配刘海屏一般**上方区域不需要黑色填补区域**,所以我们**关掉在刘海屏中上方黑色填补区域**。然后把背景图进行同黑色填补区域长度的拉升。

(这里有个坑: 黑色填补区域跟HeadRoot **不在同一个**Canvas,所以长度比例不一样,要进行转换,转换系数为: 720f / Screen.width)

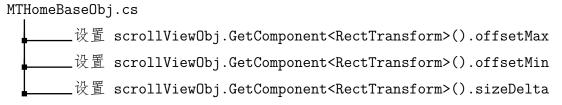


其他场景预制拉伸 -比赛界面 ->

其他场景预制拉伸,并未测试所有场景

比赛场景界面的 ScrollView 拉升。

(这边也有个坑:之前的向上拉伸都是加上 TopOffset,而现在却变成了 (TopOffset+BottomOffset)/2,原因是: 为了适配上下不同的程度的拉伸,我们进行了根节点的偏移,为了保证游戏中心点的准确。Root 偏移之后上下拉升量便相等了。所以预制的拉伸为 (TopOffset+BottomOffset)/2)



2.2 Web 适配流程

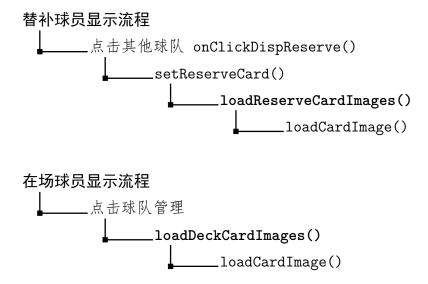
2.3 添加荣耀球员描述

流程

相关变量

- msCard.isGlory
- _panel.setActive(true);

2.4 重复球员显示标记



相关变量

- FormationDataManager._dicAllDeckCardView[_Deck_Key] 代表某球队 (_Deck_Key) 的 所有成员
- FormationDataManager._dicAllDeckCardView[_Deck_Key][0~17] 代表某球队的某个球员
- _deckNow_AllUser.Add((LocalCardData.LocalCardData.localCardData[UserCardId]).CardId):
 将 18 名在场球员对应的角色卡片 ID 存入哈希表中

在loadReserveCardImages() 中,

```
FormationMainCardInfo[] objCards = FormationDataManager._objReserveCardTop.
    gameObject.GetComponentsInChildren<FormationMainCardInfo>();
```

用于表示剩余球员的部分。

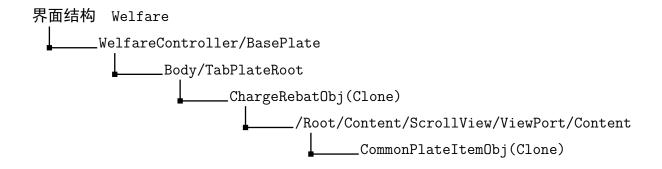
在loadDeckCardImages() 中,

```
var temp = FormationDataManager._dicAllDeckUserCardIds[tempKey];
cnt = 0;
foreach (var temp2 in temp)
{
    // 某个甲板(tempKey)场上的18名球员(temp2)
}
```

具体的球员表示如上, 而卡片的具体表示如相关变量中的第3条 (LocalData.localData..)。

2.5 福利活动未领取显示底部

- 1- 在任务界面中,将未领取的福利显示未完成,将领取的放到界面底部。
- 2- 存在部分**可以多次领取**的奖励**在领取一次后不能再次领取**的 bug

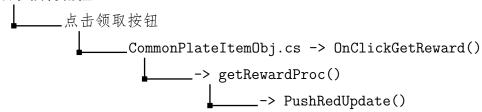


福利横条框集合 Content ->Scrow_View -> Vertical Layout Group 福利横条框 CommonPlateItemObj(Clone) -> CommonPlateItemObj.prefab

福利条框核心组件

- desc1 描述 1
- desc2 描述 2
- other 奖励
- Button 领取按钮,修改部分
- desc3 描述 3 (1/3)

领取福利流程



界面更新流程



title

第三章 相关技巧

- 3.1 黑框存在的合理性架构思考
- 3.2 SVN 冲突解决