## 代码：

实现了新的层级管理，UI Lua 脚本做了如下扩展：

CPanelBase.lua

… …

def.field("number").\_Layer = 2 -- this is not an ID but an enum, check “numDef.UI\_Sorting\_Layer”

def.field("number").\_PanelOrderType = 1 -- check EnumDef.Panel\_Order\_Type

def.field("number").\_FixedOrder = 1

def.field("number").\_RealOrderInLayer = 0

def.field("number").\_CloseMask = 0 -- 层间互斥关系 8 bit mask(对自己层无效), check

def.field("number").\_HideMask = 0 -- Hide other layers, per bit, 1 hide, 0 not

… …

这些内容不要再写在页面的逻辑代码中，统一设置在GameRes\Configs\UISettings.lua中；

页面逻辑初始化代码：

def.static('=>', CPanelXX).Instance = function()

if not instance then

instance = CPanelXX()

instance.\_PrefabPath = 你的PATH

instance.\_PanelCloseType =你的 EnumDef.PanelCloseType

instance.\_DestroyOnHide =你的Bool

instance.\_IsFullScreen = 你的Bool

instance:SetupSortingParam()

end

return instance

end

OnCreate中只做非页面逻辑和非界面表现的逻辑

//override OnData 时要调用 CPanelBase.OnData(self)

override OnHide 时要调用 CPanelBase.OnHide(self)

## 基本：

目前UI通过层（layer）和次序（order）共同排列，分别对应Canvas中的两项(after proper conventions);

\_Layer的值定义在EnumDef.UI\_Sorting\_Layer, 此定义原则上与Unity Sorting Layer的次序一致;

Layer中次序分固定(Fixed)：1-40,（81-100）和浮动(Floating)：41-80 两种;

(represented by) \_PanelOrderType的值定义在EnumDef.Panel\_Order\_Type;

Fixed 会使UI出现在固定次序, \_FixedOrder的值会被使用;

Floating会使后续出现的次序自动提高，真实值会记录在\_RealOrderInLayer;

此处的SortingOrder = Order\*10，每个Order内其实还有10个次序，+6以上都给了教学；所以每个UI自己还可以使用+6以下的次序来做特效等表现；

## 处于实验阶段中的内容：

~~1．在新UI打开时，做层间互斥：~~

~~\_CloseMask：8 bit mask，1表示关闭相应层的所有UI，自己的层是忽略的~~

2．在UI打开和关闭时，做

隐藏：

\_HideMask：8 bit mask，1表示隐藏相应的层，多个UI会叠加；

说明：8bit mask：8位2进制数，每位表示一层，（例：隐藏2，3层：110 = 6）

## 程序和策划注意

## 使用：

新加UI时要在GameRes\Configs\UISettings.lua中填写相应的设置，例如

["Panel\_DialogueNew.prefab"] = { Layer = Sorting\_Layer.ImportantTip, OrderType = Order\_Type.Fixed, FixedOrder = 10, HideMask = 30 },

否则新UI无法正常显示，策划会给好相关参数；