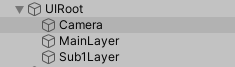
1. UI 框架的使用

框架代码命名空间 namespace UIFrame

1. 制作UI 界面跟节点创建Canvas 改名为 UIRoot

习惯设置Canvas 的 RenderModel:Screen Space -Camera



MainLayer、Sub1Layer 等可以随意创建，后边会设置加载上来的UI预制体放置在哪个Layer下

1. 创建界面预制体，最好以View结尾，以UIMainView 为例
2. 创建界面脚本
3. 界面使用 MVC 模式

UIMainPlane.cs 界面Control

UIMainView.cs 界面 view

UIMainModel.cs 界面 Model

public class UIMainPlane : UIBasePlane

{

private UIMainView \_mainView;

private UIMainModel \_mainModel;

// 界面初始化，界面创建的时候在 UIBasePlane 中自动调用

public override void Init(UIPlaneType type)

{

base.Init(type);

// 初始化 View

View = new UIMainView();

\_mainView = View as UIMainView;

// 初始化 Model

Model = new UIMainModel();

\_mainModel = Model as UIMainModel;

}

// 界面打开时，UIManager 自动调用

public override void Open(IUIDataBase data)

{

// UIBasePlane.Open 中调用了 Model.Open、View.Open

base.Open(data);

}

public void CloseOnClick()

{

CloseSelf();

}

public void ShopOnClick()

{

UIManager.GetInstance().Open(UIPlaneType.Shop, null);

}

}

在 Model.Open 方法中可以做一些数据处理

在 View.Open 方法中可以获取UI组件

1. 配置界面
2. 打开UIPlaneType.cs

添加界面枚举类型

/// <summary>

/// 界面枚举

/// </summary>

public enum UIPlaneType

{

Main, // 主界面

Shop, // 商店界面

Backpacker, // 背包界面

}

1. 打开UIPlaneRegister.cs

在public static void Register(UIConfigController controller) 方法中注册 Main界面

// 注册面板信息

public static void Register(UIConfigController controller)

{

controller.AddPlaneInfo(UIPlaneType.Scene\_Loading\_View, new UISceneLoadingPanel(), "UILoadingView", "MainLayer");

controller.AddPlaneInfo(UIPlaneType.Main\_View, new UIMainPlane(), "UIMainView", "MainLayer");

controller.AddPlaneInfo(UIPlaneType.Level\_Select\_View, new UILevelSelectPlane(), "UILevelSelectView", "MainLayer");

controller.AddPlaneInfo(UIPlaneType.Level\_View, new UILevelPlane(), "UILevelView", "MainLayer");

controller.AddPlaneInfo(UIPlaneType.Role\_Operation\_View, new UIRoleOperationPlane(), "UIRoleOperationView", "MainLayer");

controller.AddPlaneInfo(UIPlaneType.Level\_Win\_View, new UILevelWinPlane(), "UILevelWinView", "MainLayer");

1. }代码调用
2. 打开界面

UIManager.GetInstance().Open(UIPlaneType.Main\_View, null);

/// <summary>

/// 打开界面

/// 原则：同一个界面同时只存在一个

/// 如果要打开的界面已经打开了，例：打开界面 A，目前打开的界面顺

/// 序为 A-B-C-D，则依次从栈中取出 D、C、B 并关闭，然后刷新界面A

/// 为什么？避免出现 A-B-C-D-A-B-C-D-A-B-C-D 此类无限循环的界面

/// </summary>

/// <param name="type">界面枚举值</param>

/// <param name="data">需要传递给界面的数据，需要是 IUIDataBase 类型</param>

public void Open(UIPlaneType type, IUIDataBase data)

打开新界面，如果最后一个打开的界面没有执行HangUp() 挂起方法，会执行最后一个打开界面的 HangUp() 挂起方法

/// <summary>

/// 挂起：当前打开 A 界面，打开B 界面的时候，调用 A 界面的 HangUp 挂起函数

/// HangUp 可以执行一个界面出屏幕的动画，或者 SetActive(false)

/// </summary>

public virtual void HangUp()

(2)关闭界面

UIManager.GetInstance().Open(UIPlaneType.Main\_View, null);

1. 返回上一个界面(关闭最后一个打开的界面)

UIManager.GetInstance().Back();

关闭一个界面后，剩余打开界面中，获取最后一个打开的界面，如果已经执行了HangUp挂起方法，则会执行最后一个打开界面的Resume()恢复 方法

1. 判断界面是否已打开

UIManager.GetInstance().IsOpen(UIPlaneType.Main);

1. 无限循环列表

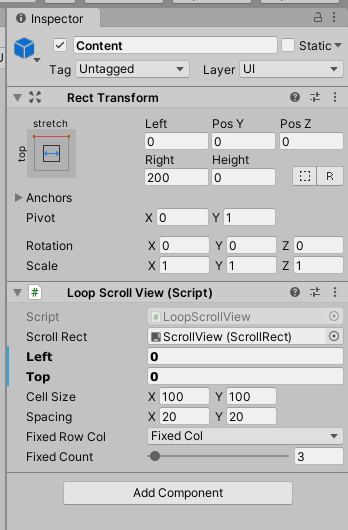
界面上展示多项物品，使用滚动列表Scroll Rect，为了提高性能，添加了无限循环滚动列表

(1)创建UGUI -> ScrollView

1. ScrollView 、Viewport 用法不变

Content 只需要添加脚本 LoopScrollView

Content 的宽高由 LoopScrollView 根据配置、子项多少自动计算



ScrollRect：需要引用ScrollView

Left：最左侧物体创建出来，距离左边界距离

Top：最上侧物体创建出来，距离上方边界距离

Cell Size：每一项的X：宽、Y：高

Spacing：X：每两个之间横向间距、Y：纵向间距

Fixed Row Col：可选项 FixedCol：固定列、FixedRow：固定行

FixedCount：选择FixedCol时限制最大列数，选择FixedRow时限制行数

(3)当 ScrollView 设置纵向滚动时，要设置为 FixedCol：固定列，创建后的下标顺序如下



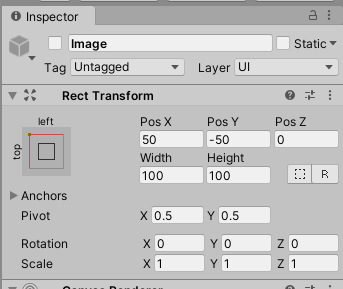
当 ScrollView 设置横向滚动时，要设置为 FixedRol：固定行，创建后的下标顺序如下



(4)在Content下创建一个子项，然后将其隐藏

子项锚点设置左上，Pivot设置 (0.5, 0.5),

Width/Height 根据第(2)条中 Cell Size 设置(运行时也会自动设置为 Cell Size)



(5)代码调用

// 获取脚本

LoopScrollView \_loopScrollView = target.GetComponent<LoopScrollView>();

// 初始化设置子项刷新回调方法

\_loopScrollView.Init(RefreshItem);

// 设置子项个数

\_loopScrollView.ItemCount(130);

// 自动跳转到第多少个子项

\_loopScrollView.GoToIndex(goToIndex);

// 获取第50个子项在当前显示区域的 上、下、左、右、中间(当前显示)

// 当ScrollView设置为横向滑动时，cellShowType ：左、右、中间(当前显示)

// 当ScrollView设置为纵向滑动时，cellShowType ：上、下、中间(当前显示)

CellShowType cellShowType = \_loopScrollView.GetCellShowType(50);

// 滑动时新出现的子项会调用刷新方法

// Transform itemTr 需要刷新的子项 Transform

// int index表示这个是要刷新的第几个子项

private void RefreshItem(Transform itemTr, int index)

{

Text text = itemTr.Find("Text").GetComponent<Text>();

Var data = dataList[index];

text.text = data.name;

}