一、内容摘要

在 BVRP 中 MVVM 是 Watch 框架的衍生产物，它应用范围是 UI 的业务逻辑处理，应用范围不包括三维物体，这一点一定要与 Watch 区分开。具体而言，Watch 目标是三维物体的数据层次化，MVVM 目标是 UI 的数据层次化，它是 Watch 框架在 UI（暂时只支持 UGUI）上的扩展应用。

二、 MVVM 支持的基本组件列表

序号	UGUI 组件	对应 View 组件	绑定值类型
1	Button	ButtonViewModel	int
2	Dropdown	DropdownViewModel	int
3	InputField	InputFieldViewModel	string
4	Scrollbar	ScrollbarViewModel	float
5	Slider	SliderViewModel	float
6	Text	TextViewModel	string
7	Toggle	ToggleViewModel	bool
8	ToggleGroup	ToggleGroupViewModel	
9	Image	ImageViewModel	

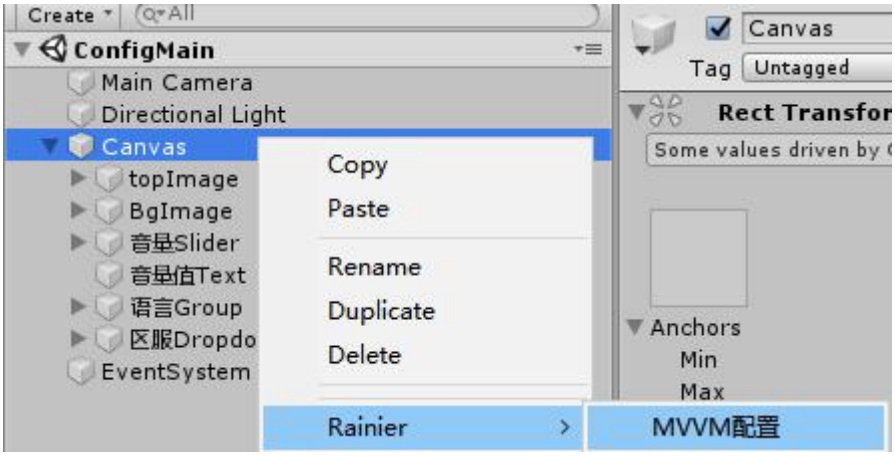
MVVM 对 UGUI 中交互组件包括 Button、Dropdown、InputField、Scrollbar、Slider、Text、Toggle、ToggleGroup、Image 添加对应的 ButtonViewModel、DropdownViewModel、InputFieldViewModel、ScrollbarViewModel、SliderViewModel、TextViewModel、ToggleViewModel、ToggleGroupViewModel、ImageViewModel 组件，其余 UI 组件暂不支持。

三、MVVM 使用

MVVM 初始化

BVRP3.0 中 MVVM 不需要再去编写 Entity 和 DataModelBehaviour 类以及进行实体绑定工作，只需添加 MVVM 事件侦听器并且给 UI 组件挂载相应的 ViewModel，然后在 LogicBehaviour 中编写业务逻辑即可。

BVRP3.0 框架中提供了 MVVM 一键初始化编辑器工具，鼠标选择 UI 根节点右键—>Rainier-->MvvM 配置—>初始化 MVVM。初始化后，UI 根节点上添加了事件侦听器（MvvmContext），同时 UI 根节点下 MVVM 支持的基本 UI 组件都会添加上相应的 ViewModel 组件，程序初始化时，会把跟节点下的所有 viewmodel 收录到 MvvmContext 中，初始化 MVVM 过程如下图所示：



点击 MVVM 配置出现如下弹窗，再点击初始化 MVVM 按钮，完成初始化操作。

MVVM 事件侦听器（MvvmContext）

MVVM 事件侦听器提供了以下几个重要方法使用:

```
// 搜索特定类型的 ViewModel
public T[] FindViewModels<T>() where T : ViewModel

// 通过类型搜索指定类型的 ViewModel
public ViewModel[] FindViewModels(Type type)

// 通过指定类型和物体名称查找 ViewModel
public ViewModel[] FindViewModels(Type type,string name)

// 搜索特定类型的 ViewModel
public T[] FindViewModels<T>(string name)

// 搜索特定类型的 UI 组件
public T[] FindUIBehaviours<T>() where T : UIBehaviour

// 搜索特定类型的 UI 组件
public UIBehaviour[] FindUIBehaviours(Type type)

// 搜索特定类型的 UI 组件
public T[] FindUIBehaviours<T>(string name)

// 搜索特定类型的 UI 组件
public UIBehaviour[] FindUIBehaviours(Type type,string name)

// 搜索特定类型的 ViewModel
public T FindViewModel<T>() where T : ViewModel

// 搜索特定类型的 ViewModel
public ViewModel FindViewModel(Type type)

// 搜索特定类型的 ViewModel
public T FindViewModel<T>(string name)
```

```
// 搜索特定类型的 ViewModel
public ViewModel FindViewModel(Type type,string name)

// 搜索特定类型的 UI 组件
public T FindUIBehaviour<T>()

// 搜索特定类型的 UI 组件
public UIBehaviour FindUIBehaviour(Type type)

// 搜索特定类型的 UI 组件
public T FindUIBehaviour<T>(string name) where T : UIBehaviour

// 搜索特定类型的 UI 组件
public UIBehaviour FindUIBehaviour(Type type,string name)

// 绑定动态 UI
public void BindingUI(Transform trans)
```

LogicBehaviour 编写

MVVM 初始化完成后，在 LogicBehaviour 中编写业务逻辑（使用模板创建更方便）。

```
public class NewMvvmLogicBehaviour : CommonLogic
{
     /// <summary>
    /// MVVM 上下文环境
    /// </summary>
    MvvmContext context;

    /// <summary>
    /// 获取 MVVM 上下文环境
    /// </summary>
    protected override void Awake()
    {
        base.Awake();
        context = GetComponent<MvvmContext>();
    }

    /// <summary>
    /// 处理业务逻辑
    /// </summary>
    /// <param name="evt"></param>
    public override void ProcessLogic(IEvent evt)
    {
        //忽略所有属性初始化事件、Diable 事件、有 Destroy 事件
        if ((evt is PropertyInitEvent) || evt.EventName.Equals("OnDisable") ||
evt.EventName.Equals("OnDestroy"))   return;

        //忽略除 ViewModel 外其他 Entity 发出的事件
        if (!(evt.EventSource is ViewModel))    return;

          //获取事件源
        ViewModel vm = evt.EventSource as ViewModel;

    }
}
```

四、注意事项

如果使用 context.FindViewModel<T>()方法的时候报 Null,请检查 context 初始化是否完成，一般需要等待一帧的时间。