ProjectBase简述

ProjectBase_v1.2.unitypackage
127.64KB

1. 概述

ProjectBase + DOTween

将游戏的功能整理分类，集合了常用的单例，降低样板代码，适合mini、小型游戏开发、GameJam，既可作为扩展，也能作为基础框架。

包含 资源管理、UI 管理、事件、持久化、对象池、音乐、输入、常量管理、Mono等常见功能。

（后续看情况吧，计划把有屏幕自动横竖屏适配、语言本地化的版本也写成文档上传）

- **总体结构**
  - 通过 SingletonBase 、SingletonMono把管理器类暴露为全局访问点（例如 ResManager、UIManager、EventsManager、PPDataManager、JsonManager、ObjectPool、MusicManager、InputManager 等）。
- **核心模块与职责**
  - SingletonBase / SingletonMono（单例基类）
    - 通过 `SingletonBase<T>` 与 `SingletonMono<T>` 快速创建全局单例，降低样板代码。
    - SingletonBase是不带Mono的单例；SingletonMono是有Mono的单例；
  - ResManager（资源同步、异步加载）
    - 提供同步/异步加载接口，简化资源获取并能在多个系统间复用加载逻辑。
    - 提供了加载回调。
  - UIManager + BasePanel（UI管理）（最常用）
    - 基于MainCanvas，将其分层（bot/mid/top/system），按层级创建与缓存面板。
    - UIManager统一加载、卸载UI、检查UI是否卸载、给控件添加事件
    - BasePanel是所有UI基类。能自动获取子控件、给button和toggle绑定事件、有UI显示与关闭调用（可添加进入、退出动画等）（类似状态机）
    - 每个UI制作成预制体，默认需要放在Resources/UI目录下。

- EventsManager
  - 基于泛型 UnityAction<T> 的事件系统，使用 IEventInfo + EventInfo<T> 避免装箱拆箱，按事件名订阅与触发，类型安全。（事件名称可以放在Constants下）
- PPDataManager / JsonManager（数据持久化）
  - PPDataManager 使用反射把对象字段逐一保存为键值（SaveData/LoadData），适合简单持久化场景。
  - JsonManager 封装 JsonUtility 的保存/加载到文件（persistentDataPath），便于存储结构化数据。
  - （推荐直接使用Easy Save3，简单，无需考虑序列化，速度比PlayerPrefs快）
- ObjectPool
  - 实现对象复用、父节点管理与回收，减少频繁 Instantiate/Destroy 带来的 GC 与性能开销。
- MusicManager
  - 统一管理 BGM 与音效（异步加载、音量控制、回收），与 MonoController 联动做播放结束回收，自带AudioSource池。
- MonoController
  - MonoController提供统一的 Update 回调注册点（Add/RemoveUpdateListener），便于跨系统的每帧调度与管理。
- InputManager
  - `InputManager` 抽象输入逻辑，便于替换/扩展。

2. 功能简述

2.1 UI功能

2.1.1 结构 UIManager 和 BasePanel

UIManager 负责全局 UI 的管理、加载与调度；

BasePanel 为继承它的面板提供控件查找、适配、动画与生命周期（类似状态机）的通用实现，供具体面板继承使用。

UIManager的主要职责

- 管理整个 UI 画布与分层（bot/mid/top/system）。
- **显示/回收**面板：加载 UI prefab，将UI（BasePanel ）加载到不同的层级。内部维护了 pnlDictionary 和 pnlList，处理面板的 Show/Hide 流程。

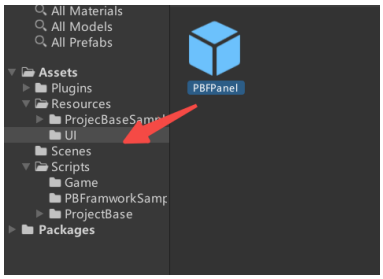
- 显示面板示例：`UIManager.Instance.ShowPanel<PBFPanel>("PBFPanel");` 推荐面板的名称 和 脚本名称 同名。
- 面板队列与返回键：维护面板栈，用于 ESC/返回键行为（派发 OnEscape）。
- UI 动画队列：维护 preparateSeqList 与 playingSeq，顺序播放 UI 动画并提供添加/移除/跳过接口。
- 可手动给控件添加事件，如点击、拖动等（AddCustomEventListener）
- ~~屏幕与适配管理：记录屏幕尺寸、横竖屏判断、响应 OnOrientationChange 并调整 CanvasScaler。~~

BasePanel主要指责

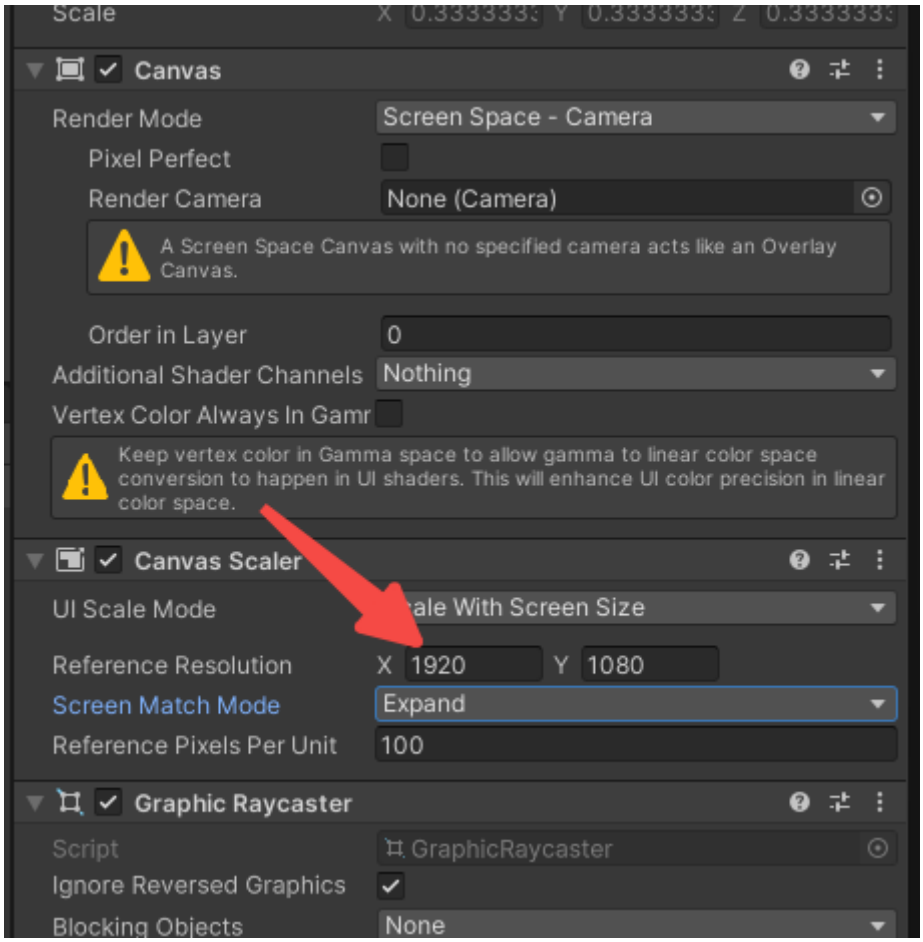
- 面板基类，封装常用 UI 行为和生命周期钩子（Awake/Start/OnDestroy/ShowMe/HideMe）。
- 自动收集并缓存子控件（Button/Image/Text/Toggle/...），并统一绑定点击/切换回调（子类重写OnBtnClick/OnToggleValueChange方法即可）。
- 通用接口：提供 GetControler、FindInParents、OnEscape 等供子类使用。
- 动画封装：统一管理面板内的 DOTween Sequence（nowSeq/isTweening），在动画期间控制交互（cG.interactable），并把动画加入 UIManager 的动画队列。
- 子类UI重写OnBtnClick方法，通过按钮名称执行对应的事件，简化操作
- ~~适配支持：管理 AdaptiveElement、ScreenAdaptTitle/Panel，提供 DoAdaptive 与横竖屏回调处理。~~

2.1.2 使用方法：

- 引擎中的操作
  - 创建一个UI界面：给根canvas，挂载一个脚本（暂时取名PBFPanel），让脚本继承BasePanel
  - 给MainCanvas打上MainCanvas的tag，MainEventSystem打上MainEventSystem的tag；或者直接把MainCanvas和MainEventSystem放在Resource/UI文件目录下



  - 注意游戏的屏幕尺寸，默认是1920*1080



- 给MainCanvas创建子Canvas，用于分层显示UI，子Canvas命名：bot，mid，top，system

相关代码

```
//UIManager的构造函数

GameObject canvasObj = GameObject.FindWithTag("MainCanvas");
GameObject eventSystemObj = GameObject.FindWithTag("MainEventSystem");

//通过Tag查找，找不到则自动加载Canvas和EventSystem
if (canvasObj == null)
    canvasObj = ResManager.Instance.Load<GameObject>("UI/Canvas");
if (eventSystemObj == null)
    eventSystemObj = ResManager.Instance.Load<GameObject>
("UI/EventSystem");

GameObject.DontDestroyOnLoad(canvasObj);
GameObject.DontDestroyOnLoad(eventSystemObj);
canvasRect = canvasObj.transform as RectTransform;
canvasScaler = canvasObj.GetComponent<CanvasScaler>();

bot = canvasRect.Find("bot");
mid = canvasRect.Find("mid");
top = canvasRect.Find("top");
system = canvasRect.Find("system");
```

- 场景中MainCanvas和bot，mid，top，system和其他根Canvas的AnchorPreset设置成stretch

- 制作的UI放到Resource下，需要时候再通过UIManager加载
- UI的名字和路径可以放到Constants的脚本中

- button、toggle事件

  因为统一绑定回调，所以只需根据名称来选择需要执行的方法

  举例：

```
代码块
protected override void OnBtnClick(string btnName)
{
    switch (btnName)
    {
        //设置按钮
        case "CloseBtn":
            UIManager.Instance.HidePanel("DailyPlayView");
            break;
        case "PlayBtn":
            DailyPlayManager.Instance.SetPlayButtonState();
            break;
    }
}
```

- 界面切换
  - 显示UI（参数：UI名称，层级，回调）：`public void ShowPanel<T>(string panelName, UIM_layer layer = UIM_layer.Mid, UnityAction<T> callback = null)`

    - 关闭UI（参数：UI名称）：`public void HidePanel(string panelName)`
- 其他：获取UI、移除等待动画...略

2.2 单例基类

- SingletonMono继承mono，能挂载
- SingletonBase，不能被挂载

2.3 事件系统EventsManager

- 添加事件 `EventsManager.Instance.AddEventsListener<T>(string name, UnityAction<T> action)`
- 触发事件 `EventsManager.Instance.EventTrigger<T>(string name,T info)`
- 移除事件 `EventsManager.InstanceRemoveListener<T>(string name,UnityAction<T> action)`
- 事件的name可以放到Constants.Event里

2.4 常量Constants类

- 参考脚本

2.5 数据本地存储

- 主要使用PPDataManager，能存基础类型、IDictionary和List等
- 不能直接存自定义数据类型，如果有这方面需求，可以使用Easy Save 3，能直接保存，比PlayerPrefs更加简单，无需考虑序列化。

2.6 输入系管理器InputManager

- 目前检测了鼠标左右键的3种状态，和Esc键的输入
- InputUpdate添加到了MonoController的AddUpdateListener中，每帧检测

2.7 MonoController

- 每帧触发UnityAction，常用于检查等（AddUpdateListener和RemoveUpdateListener）

2.8 MusicManager全局音效、音乐管理器（TODO优化）

- 对象池管理AudioSource，播放音效时候自动获取AudioSource
- 支持异步加载音效与加载成功回调
- 支持播放/停止/暂停背景音乐、播放一次性音效、调整/保存音量和静音、跨场景保持（DontDestroyOnLoad）。

- 包含切换曲目、淡入淡出、音量渐变、按名称或 AudioClip 播放音乐、对象池池化等辅助逻辑。

2.9 ObjectPool和PoolData

- 将GameObject包装成PoolData，PoolData里有列表保存同类GameObject。ObjectPool的池是GameObject名称-PoolData的字典，通用对象池，可以容纳多种不同类别的GameObject。
- 通过回调获取物品。

示例

```
// 异步获取并使用
void SpawnAsync(Transform parent)
{
    string path = "Prefabs/MyBullet"; // 资源路径，最后一段 "MyBullet" 作为池 key
    ObjectPool.Instance.GetObj(path, (go) =>
    {
        // go 已获取（来自池或新加载）
        go.transform.SetParent(parent);
        go.transform.localPosition = Vector3.zero;
        go.SetActive(true);
        // 初始化逻辑...
    }, parent, true);
}

// 回收（用完后）
void ReturnToPool(GameObject go)
{
    // 注意：使用 obj.name 作为 key，确保它等于资源名 "MyBullet"
    ObjectPool.Instance.TStoreObj(go.name, go);
}
```

优化方向

1. 目前实现风格偏轻量和“实用型”，若团队需要更强的可测试性/依赖注入，可以在单例与模块间引入更严格的接口/注入层。