



腾讯GAD
游戏开发者平台

做有梦想的游戏人!

贪吃蛇大作战

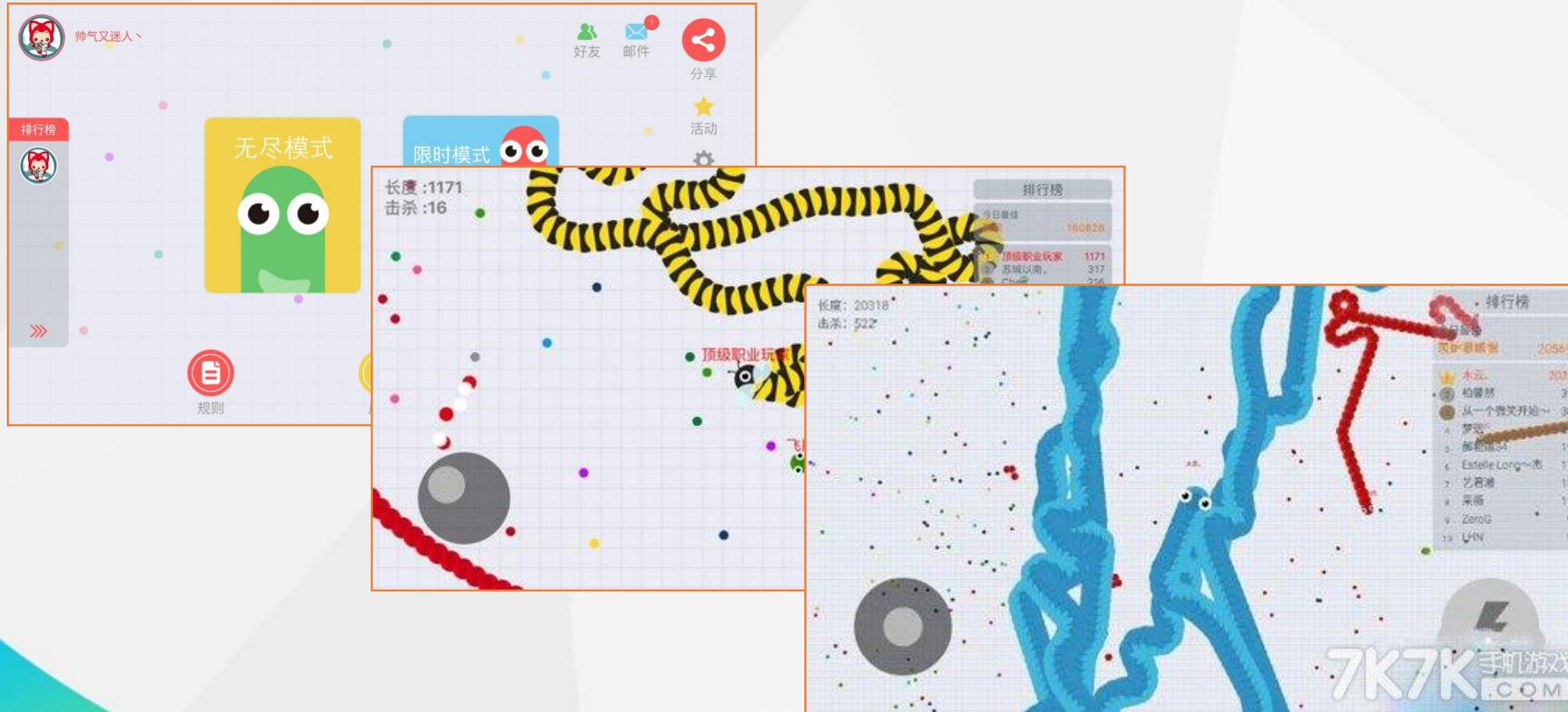
Tencent , SlicolTang

0、课程概述

- 0.1、游戏简介
- 0.2、课程目标

0.1、游戏简介

- 来自一款AppStore上架的同名游戏。



0.2、课程目标

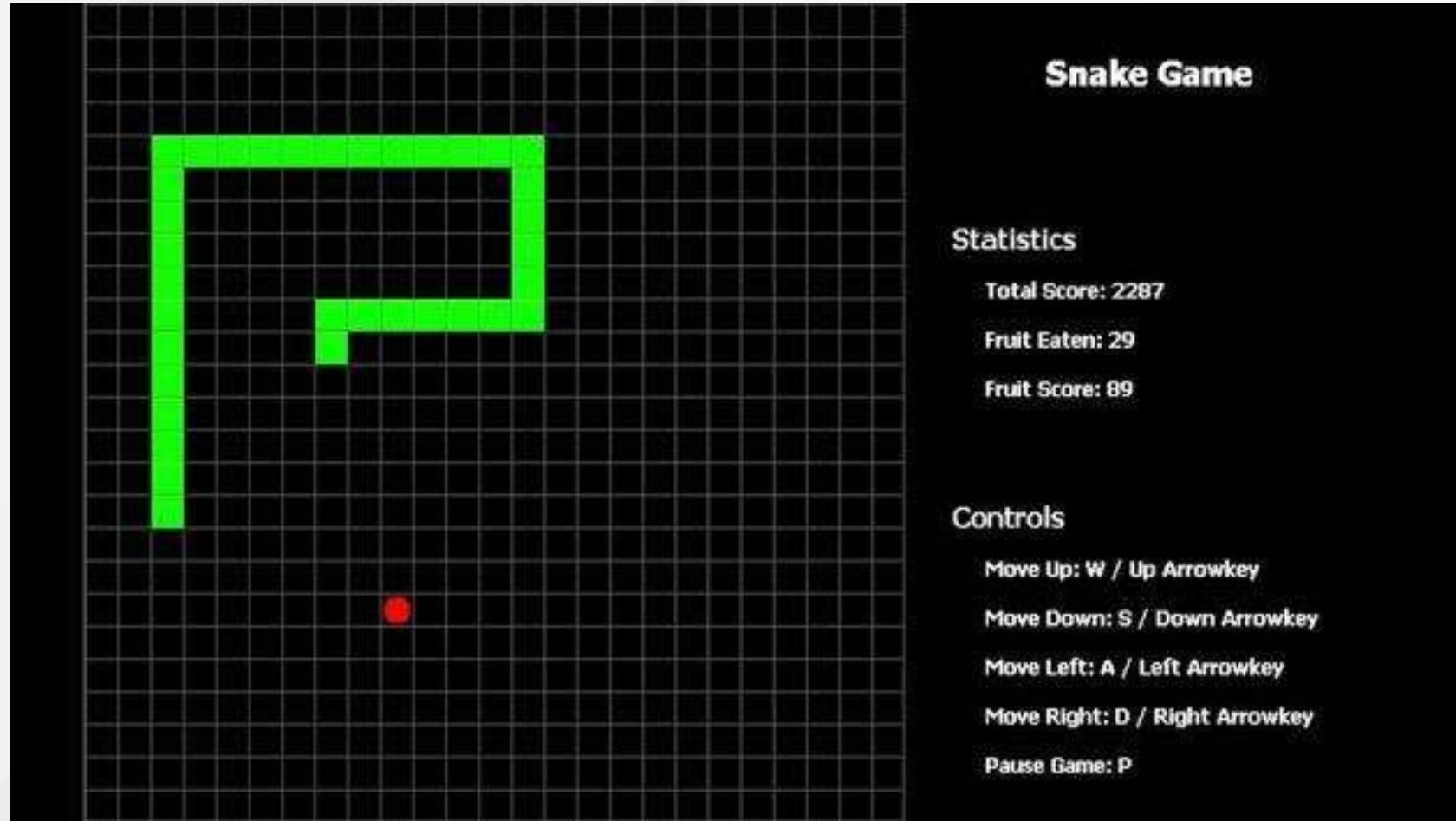
- 课程大纲：

第一部分	第二部分	第三部分
1、规划游戏功能	6、UI框架及MVC模式	12、游戏打包
2、功能模块划分	7、主城及相关模块	
3、系统架构设计	8、核心玩法	
4、开始创建项目	9、帧同步	
5、让架构先跑起来	10、局域网对战	
	11、换肤和特效	

1、规划游戏功能

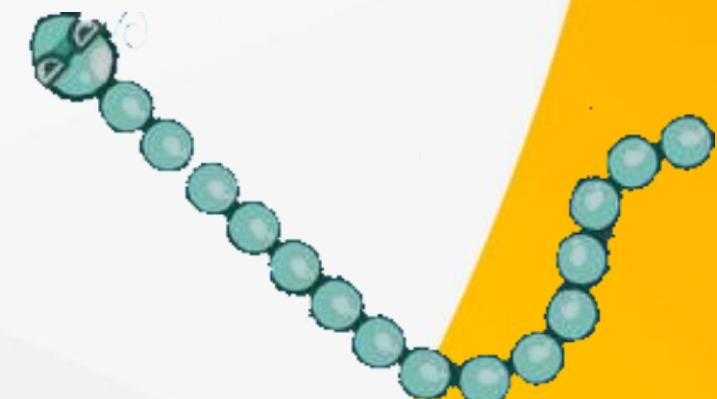
- 1.1 经典贪吃蛇玩法
- 1.2 蛇可有不同种类
- 1.3 食物有不同种类
- 1.4 可多人联机对战

1.1、经典贪吃蛇玩法



1.2、蛇有不同的种类

- 初始长度
- 食物转化率
- 视野范围
- 死亡转化率
- 攻击技能

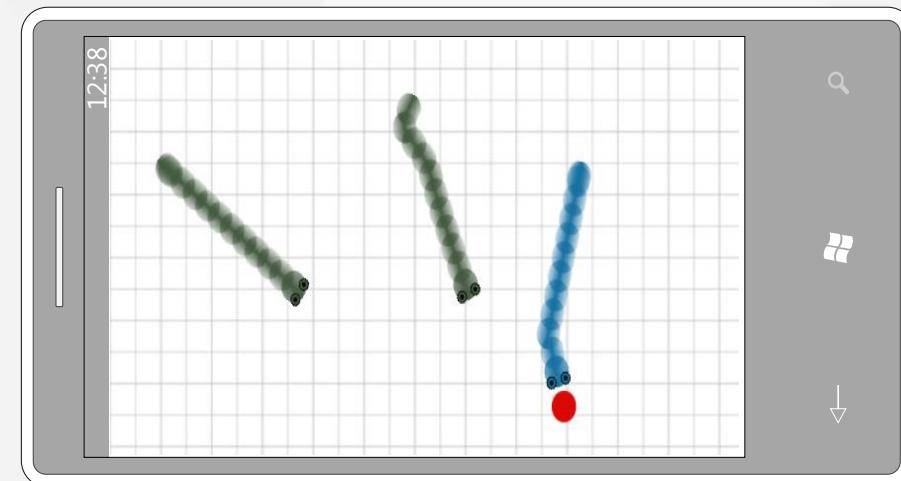
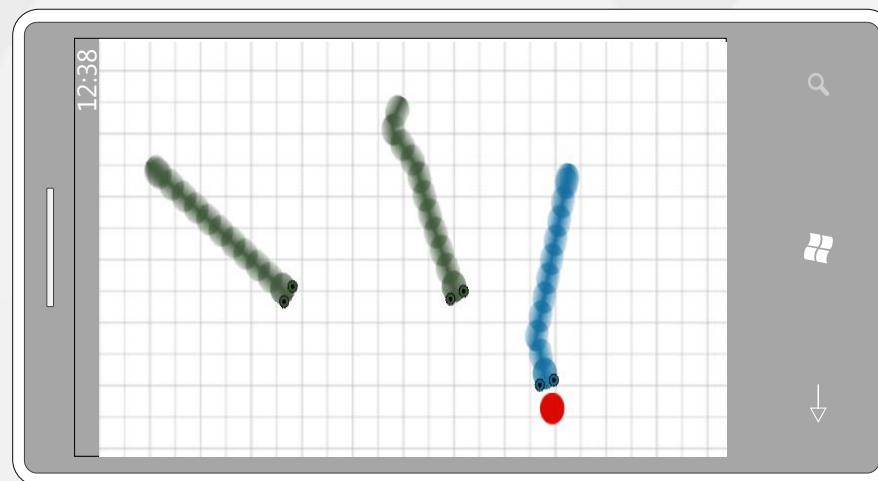


1.3、食物有不同种类

- 普通食物
 - 提供成长值
- BUFF食物
 - 提供成长值
 - 附加BUFF效果
 - 比如：双倍成长值，扩大视野，提高食物转化率

1.4、多人联机对战

- 通过局域网和帧同步的方式

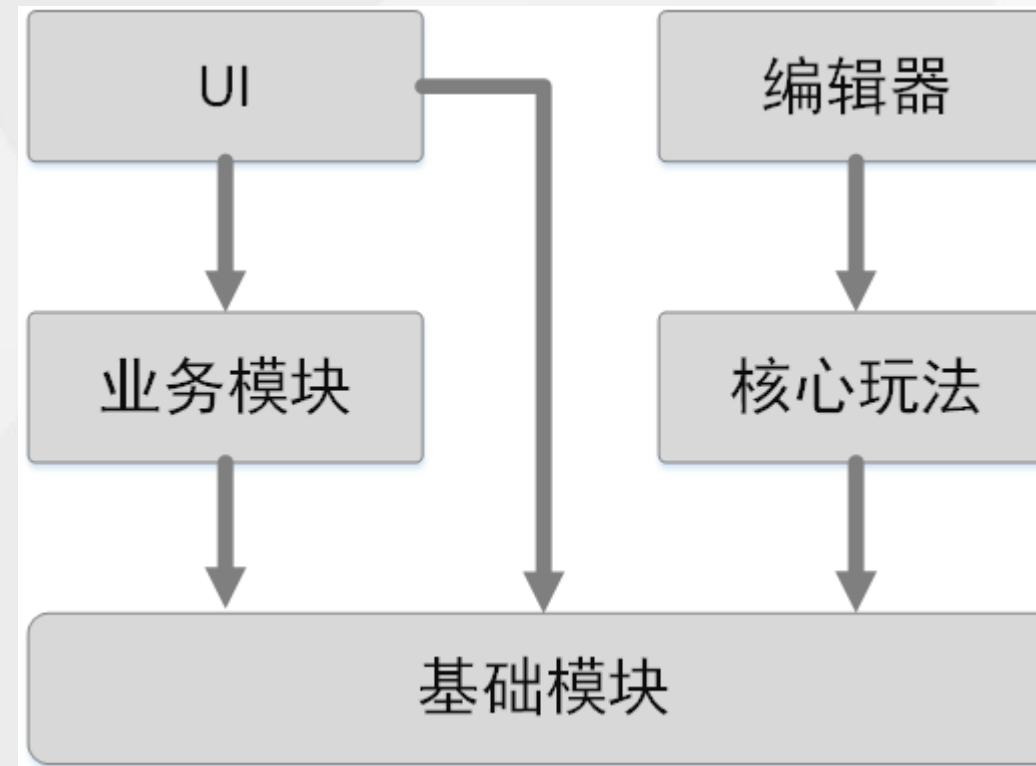


2、功能模块划分

- 2.1 模块依赖关系
- 2.2 模块功能细分

2、功能模块划分

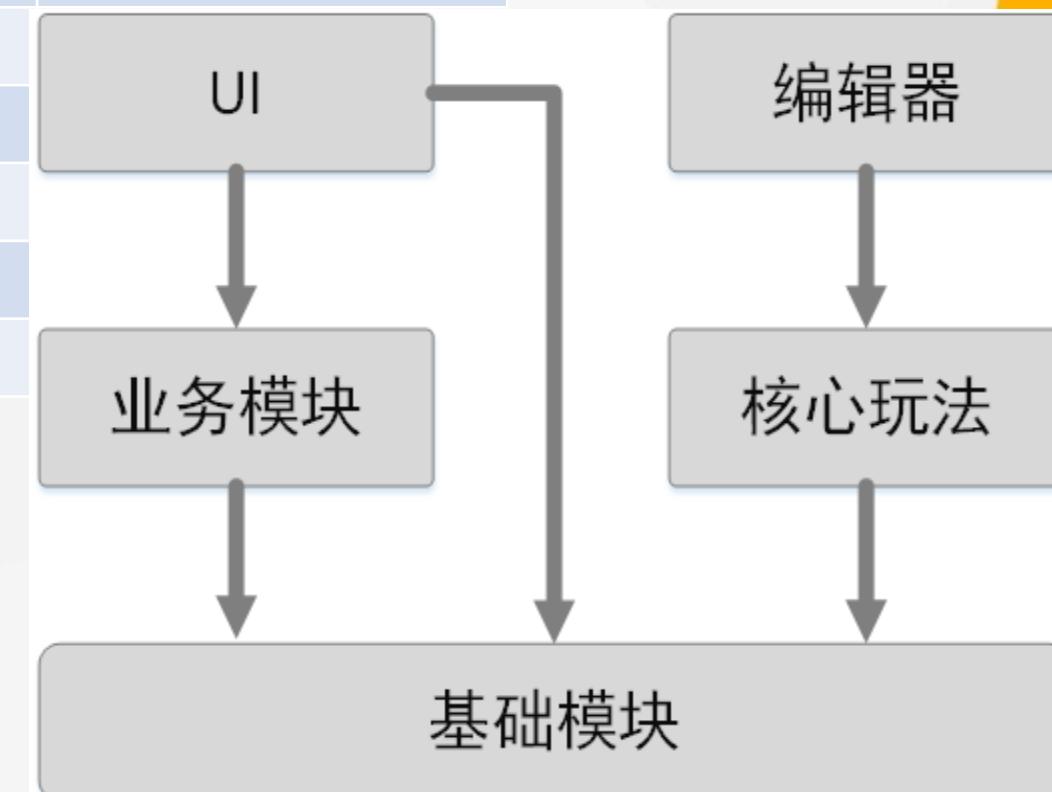
- 2.1 模块依赖关系



2、功能模块划分

• 2.2 模块功能细分

业务模块	基础模块	核心玩法	编辑器功能
注册与登录	模块管理	战斗单局	关卡编辑器
主城	UI管理	局域网匹配	皮肤编辑器
关卡	用户管理	技能系统	
背包	资源管理	BUFF系统	
商店	配置管理	AI系统	
任务	网络管理		
活动	支付管理		
	分享管理		

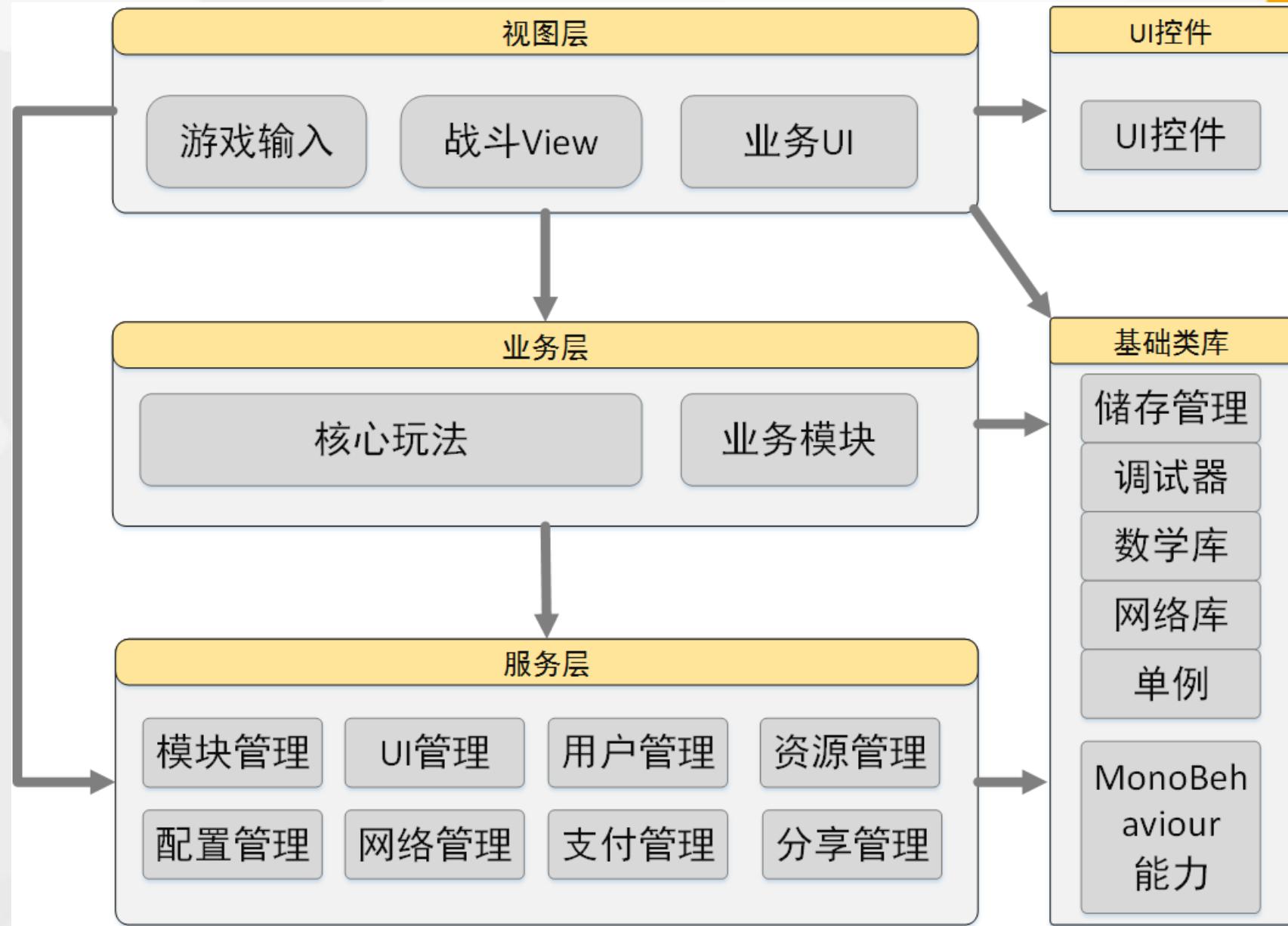


3、系统架构设计

- 3.1 设计思路
- 3.2 启动模块

3、系统架构设计

- 3.1 设计思路
 - 3.1.1 系统分层

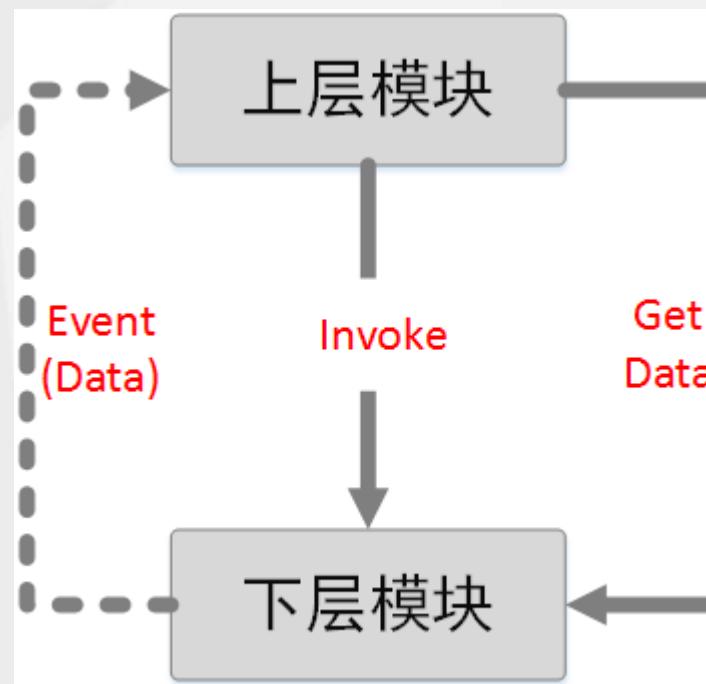


3、系统架构设计

• 3.1 设计思路

• 3.1.2 单向依赖

- 不同层级之间的模块单向依赖
- 仅允许上层模块依赖下层模块

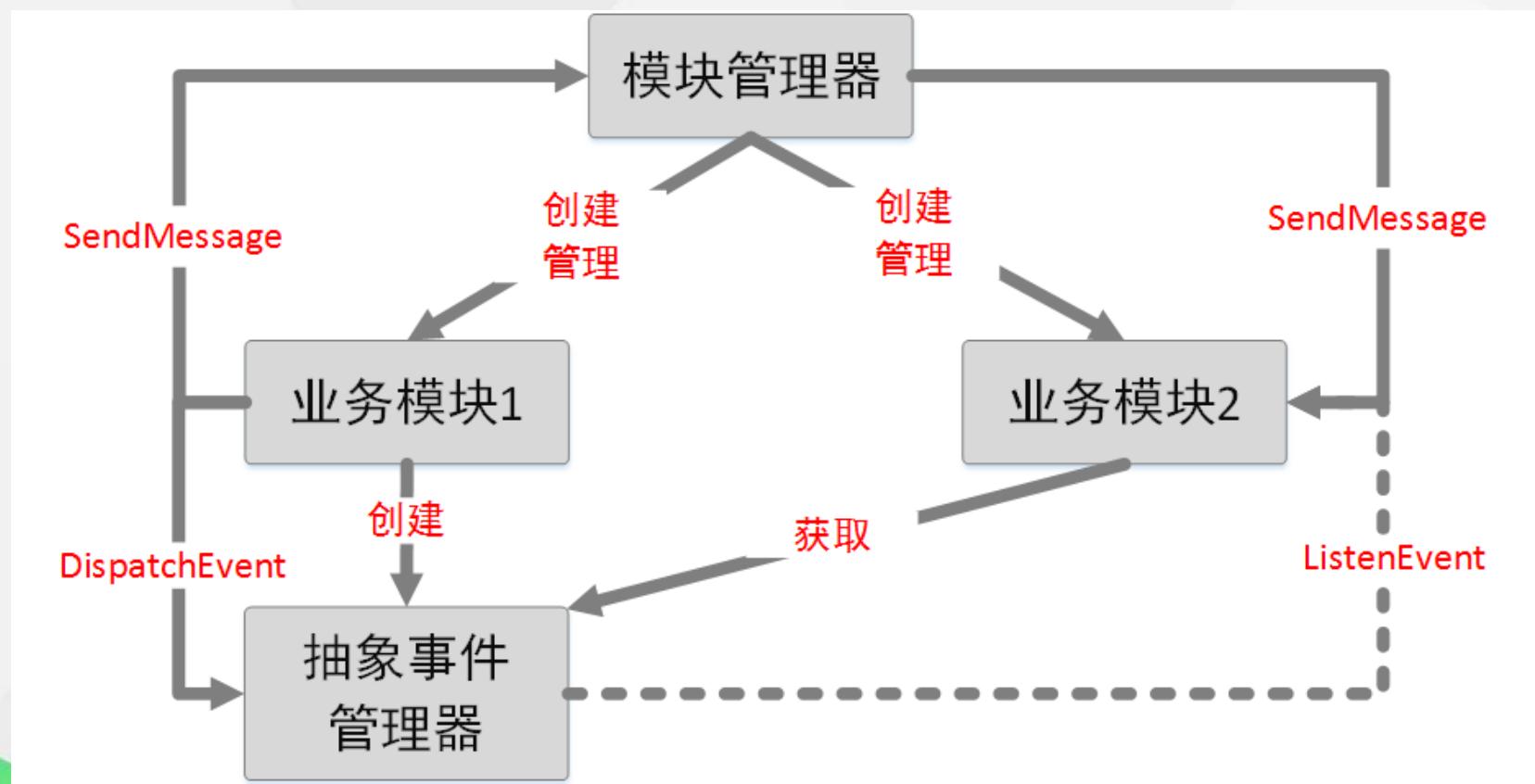


3、系统架构设计

• 3.1 设计思路

• 3.1.3 模块解耦

• 业务层模块通过【事件】与【消息】的方式通讯

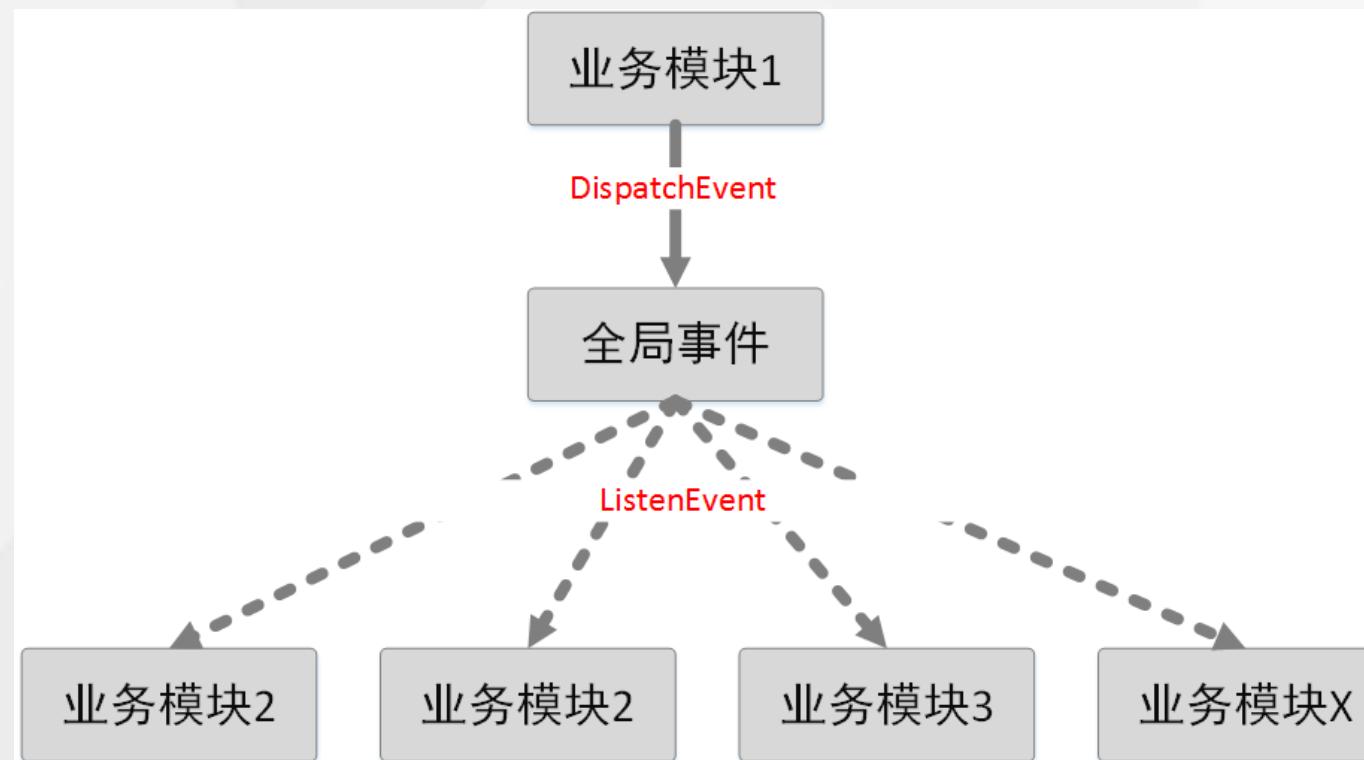


3、系统架构设计

• 3.1 设计思路

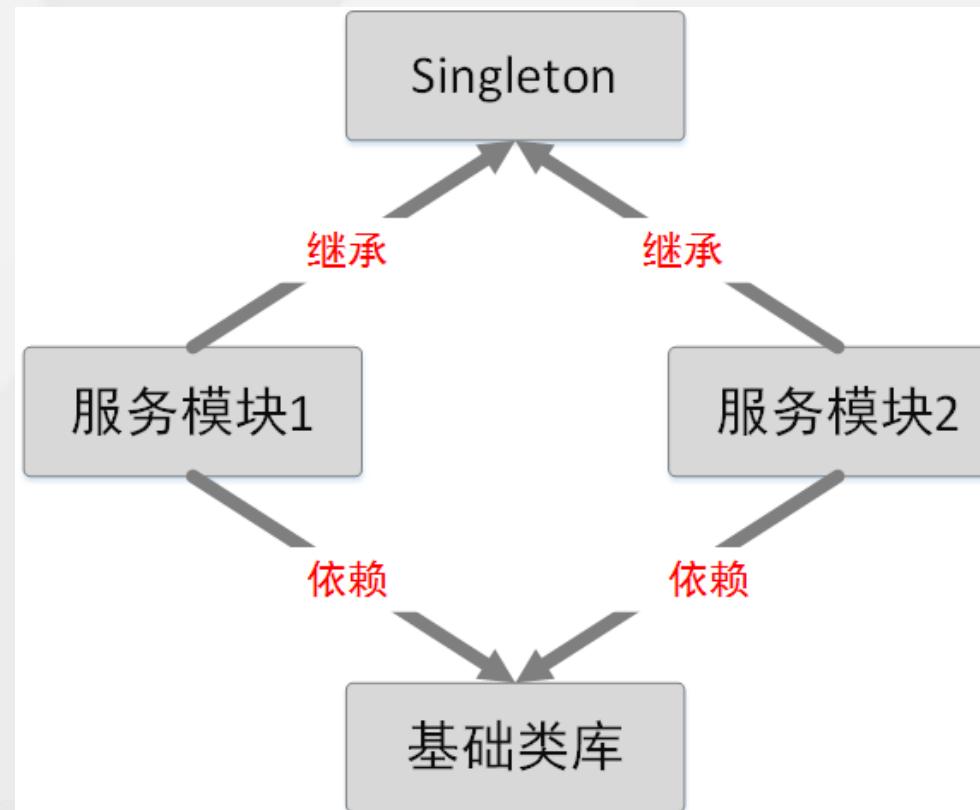
• 3.1.4 全局事件

- 有些事件不是从固定模块发出
- 有些事件模块影响全局逻辑
- (全局事件会系统可读性变差，但是又必不可少)



3、系统架构设计

- 3.1 设计思路
 - 3.1.5 模块独立
 - 服务层模块之间完全独立
 - 如果2个模块仍然必须有公共逻辑，将其再次抽象，放入基础类库



3、系统架构设计

- 3.1 设计思路

- 3.1.6 代码重用

- 业务层公共逻辑->服务层
 - 服务层公共逻辑->基础类库
 - 视图层公共逻辑->UI控件

3、系统架构设计

- 3.1 设计思路
 - 3.1.7 设计模式

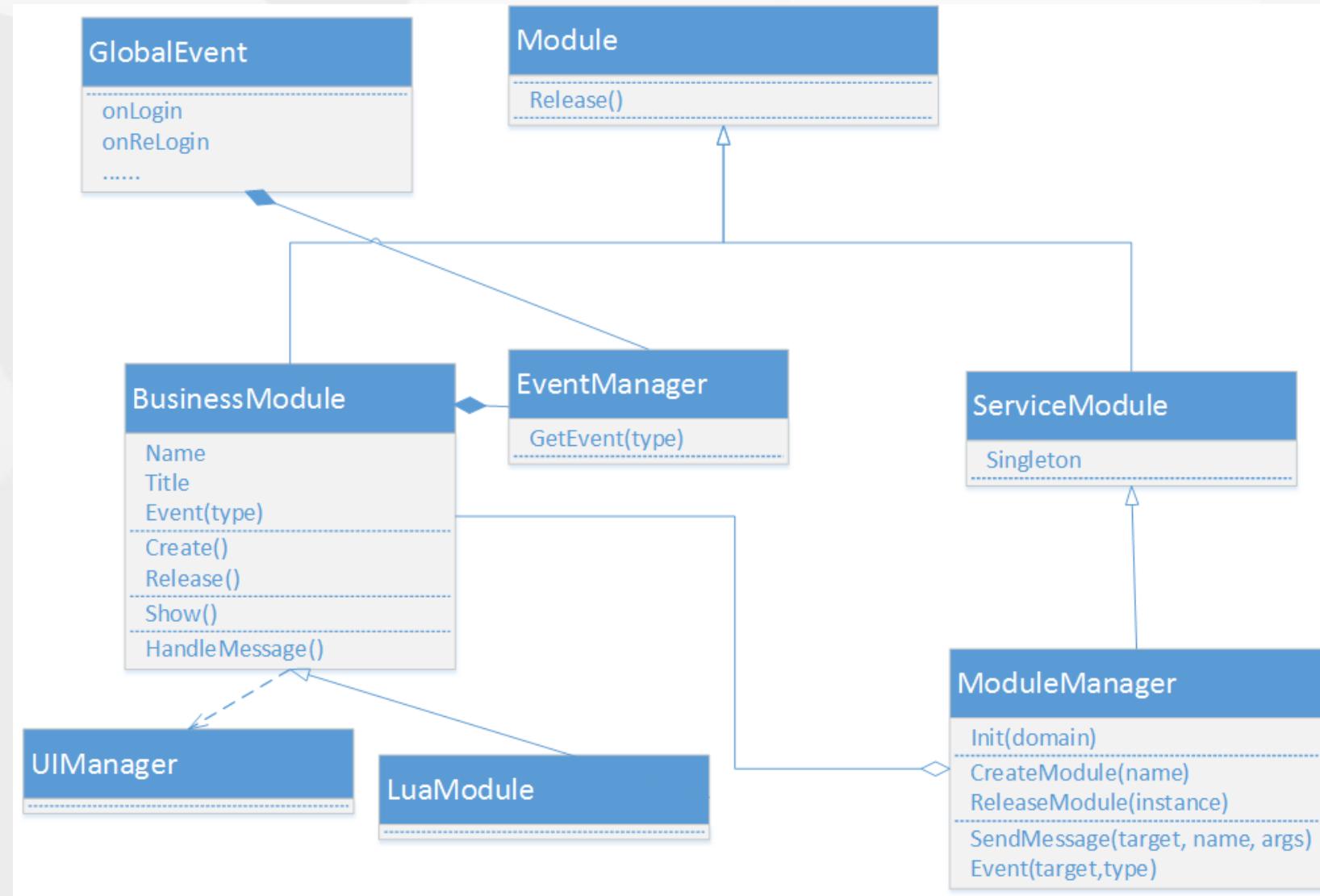
设计模式	应用场景		
观察者模式	业务层之间通讯	下层与上层通讯	客户端与服务器通讯
命令模式	业务层之间通讯		
单例模式	服务层为上层提供功能		
MVC模式	视图层与业务层通讯		
工厂模式	视图层实例的创建管理	核心玩法中角色的创建	特效的创建和管理

3、系统架构设计

• 3.2 启动模块

• 3.2.1 模块管理

- 整个系统都是从模块管理器开始启动

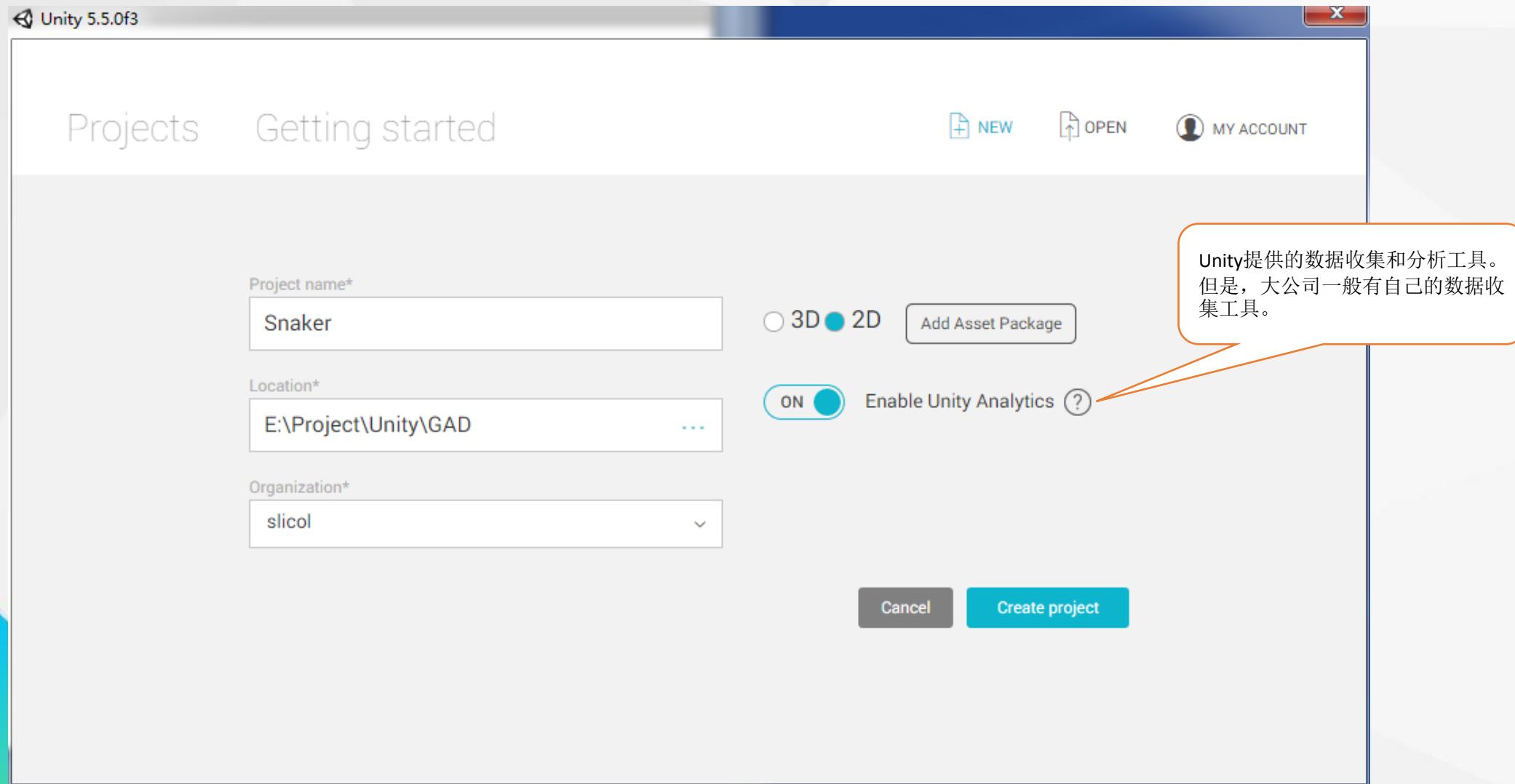


4、开始创建项目

- 4.1 创建Unity Project
- 4.2 创建目录
- 4.3 设置参数
- 4.4 创建场景及启动类

4、开始创建项目

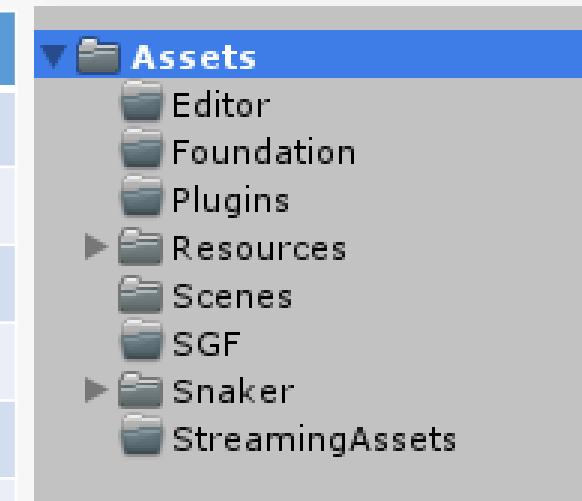
• 4.1 创建Unity Project



4、开始创建项目

- 4.2 创建目录
 - 4.2.1 一级目录

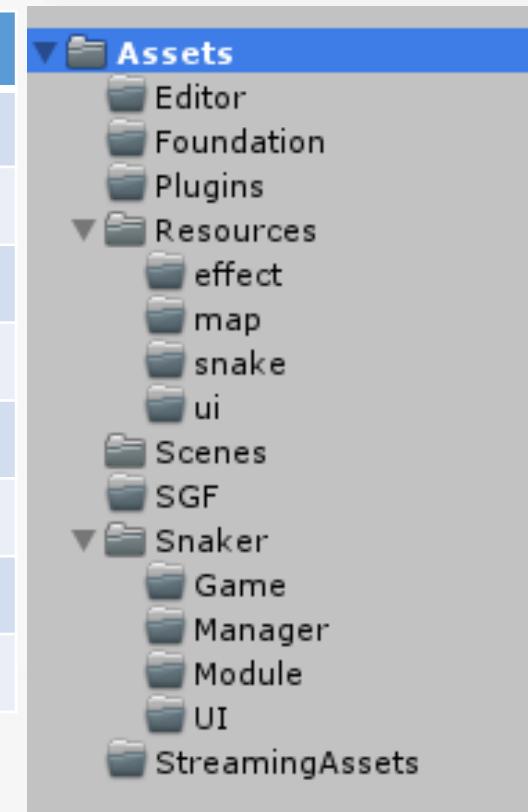
目录	系统目录	定义
Editor	Yes	用于放置编辑器扩展相关的类
Plugins	Yes	用于分平台放置第3方，或者自定义的任意SDK
Resources	Yes	用于放置游戏中不需要更新的资源
StreamingAssets	Yes	用于放置游戏中需要更新的资源和配置
Scenes	No	用于放置Unity场景文件，方便打包时全部引用
Snaker	No	用于放置游戏的视图层、业务层、服务层代码
SGF	No	用于放置基础类库源代码
Foundation	No	用于放置基础类库DLL（基于.Net框架的）



4、开始创建项目

- 4.2 创建目录
 - 4.2.2 二级目录

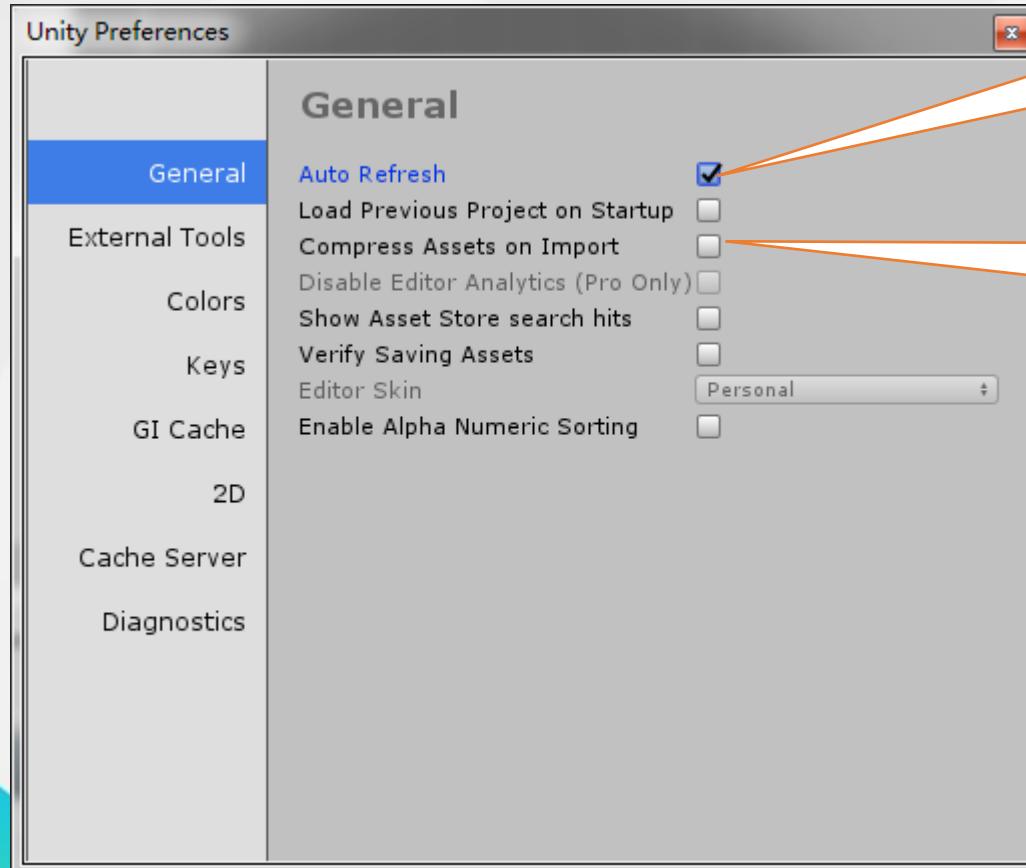
一级目录	二级目录	定义
Resources	effect	特效资源
	map	地图（关卡）资源
	snake	蛇的资源（皮肤资源）
	ui	各个业务模块的UI资源
Snaker	Game	游戏核心玩法的实现
	Service	服务层模块（以ServiceModule单例的形式提供功能）
	Module	各个业务模块（比如登录，商店等）
	UI	各个业务模块对应的UI实现



4、开始创建项目

• 4.3 设置参数

• 4.3.1 Edit/Preferences/General

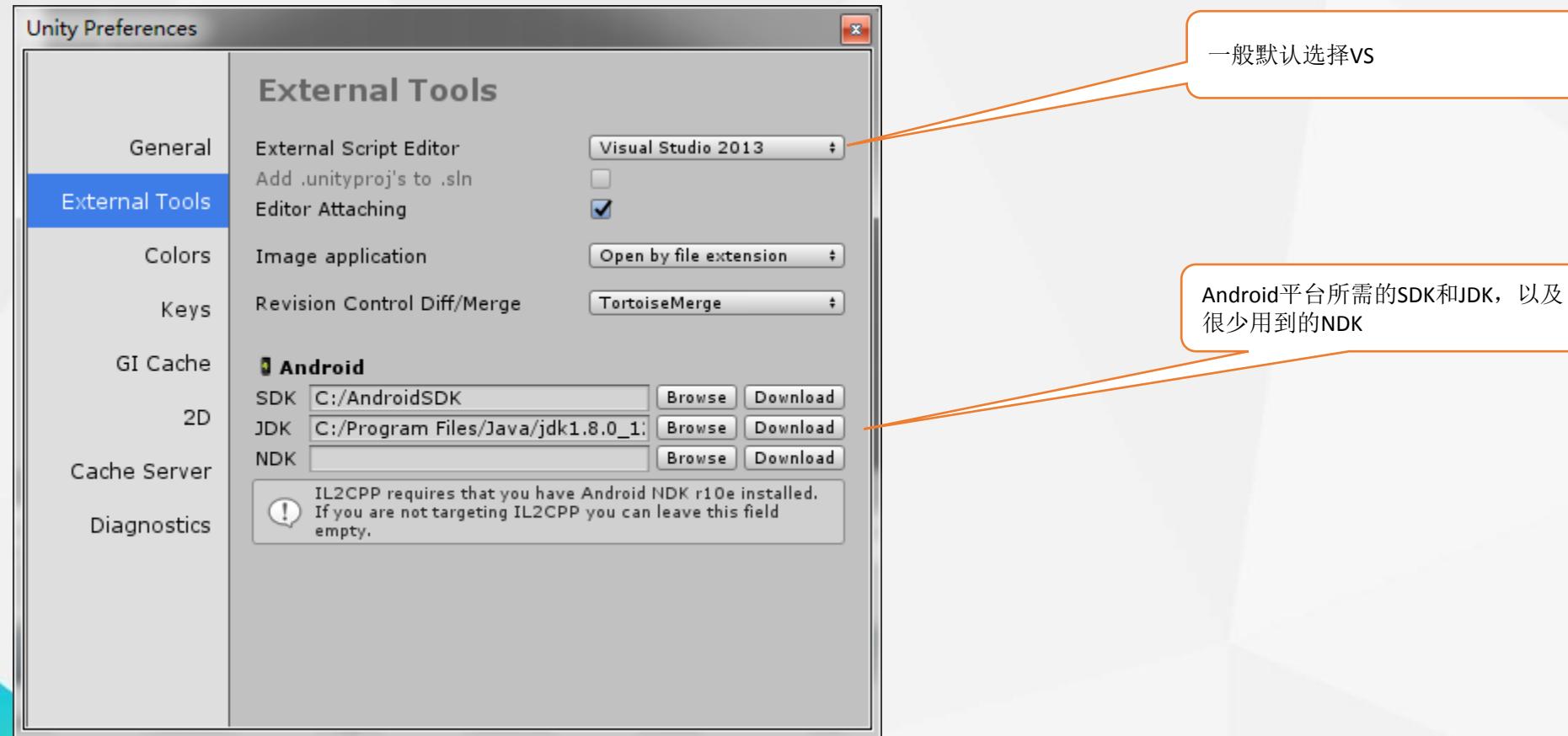


当代码或者资源发生变化时，当
Unity获得焦点时，会进行一次编
译和刷新操作。
视自己习惯决定是否勾选！

如果勾选，当Unity切换平台时，
会有一个漫长的资源压缩过程。
所以一般不勾选！

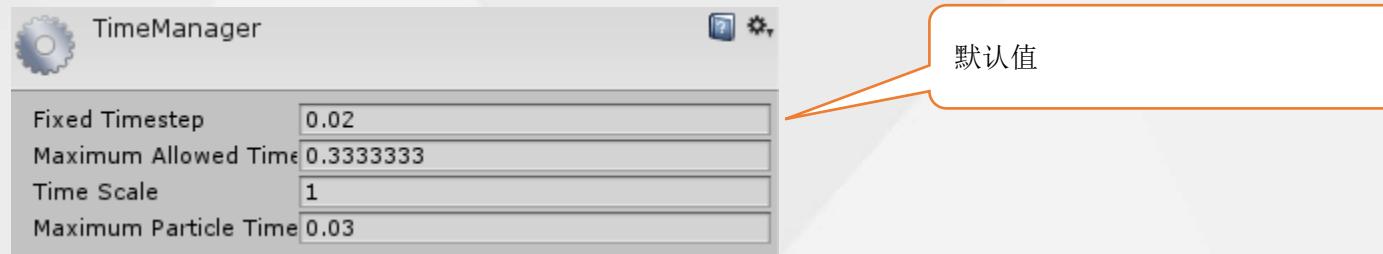
4、开始创建项目

- 4.3 设置参数
 - 4.3.2 Edit/Preferences/External Tools



4、开始创建项目

- 4.3 设置参数
 - 4.3.2 Edit/Project Settings/Time



4、开始创建项目

• 4.4 创建场景及启动类

- 4.4.1 将场景命名为Main，保存在Scenes目录下
- 4.4.2 新建App启动类AppMain
- 4.4.3 修改启动顺序 (Edit/Project Settings/Script Execution Order)



如果同一场景中多个MonoBehaviour挂在GameObject上，那么，AppMain将最先被实例化

5、让架构先跑起来

- 5.1 实现日志系统
- 5.2 实现模块管理器
- 5.3 验证架构是否正常工作

5、让架构先跑起来

• 5.1 实现日志系统

- 5.1.1 实现 : Debugger
- 5.1.2 实现 : DebuggerExtension

```
1 个引用
public virtual void Create()
{
    this.Log("Create()");
}

1 个引用
public virtual void Release()
{
    this.Log("Release");

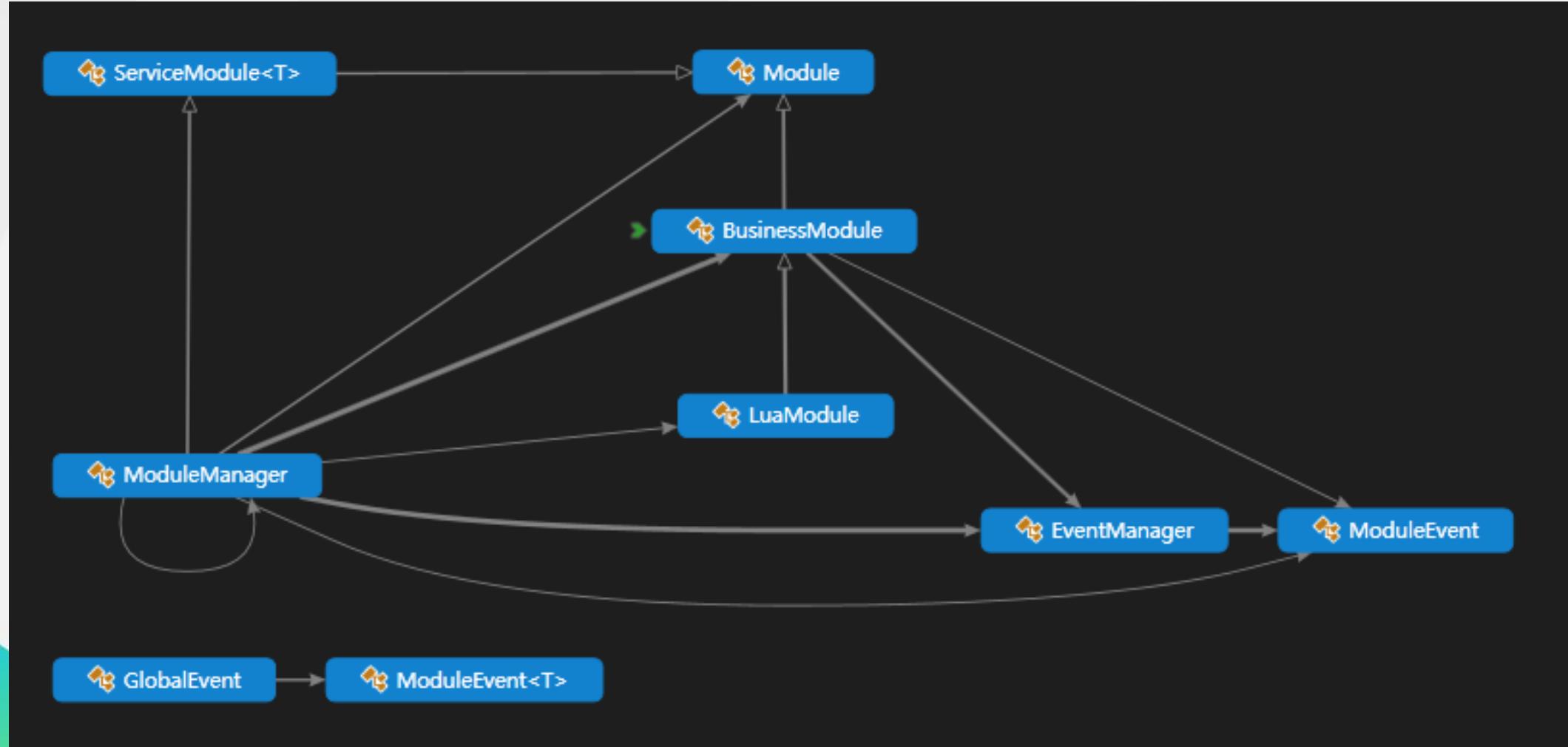
    if (m_mapEvents != null)
    {
        foreach (var @event in m_mapEvents)
        {
            @event.Value.RemoveAllListeners();
        }
        m_mapEvents.Clear();
    }
}
```

```
Clear | Collapse | Clear on Play | Error Pause | 
> 20:03:26.806 Module::Create()
UnityEngine.Debuger:Log(String, String)
! > 20:03:26.815 Module::Release
UnityEngine.Debuger:Log(String, String)
```

工欲善其事，必先利其器

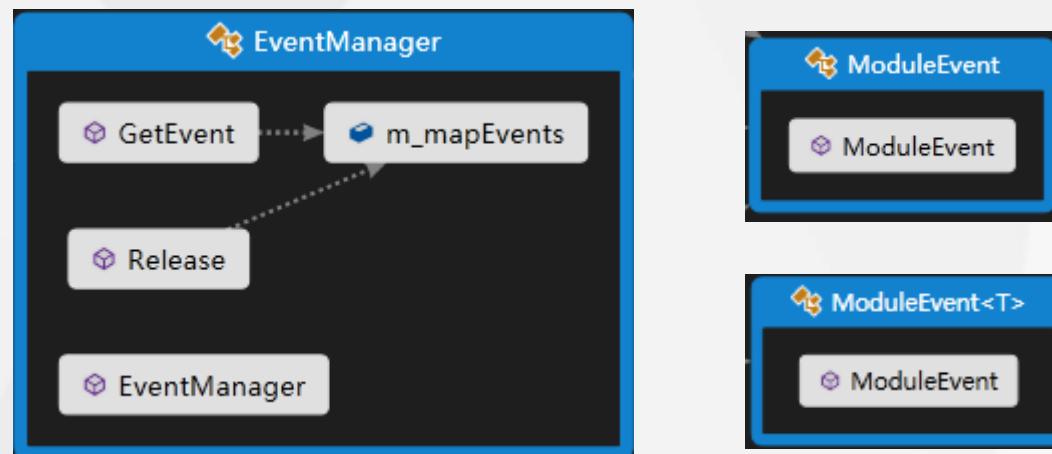
5、让架构先跑起来

- 5.2 实现模块管理器
 - 总体类图



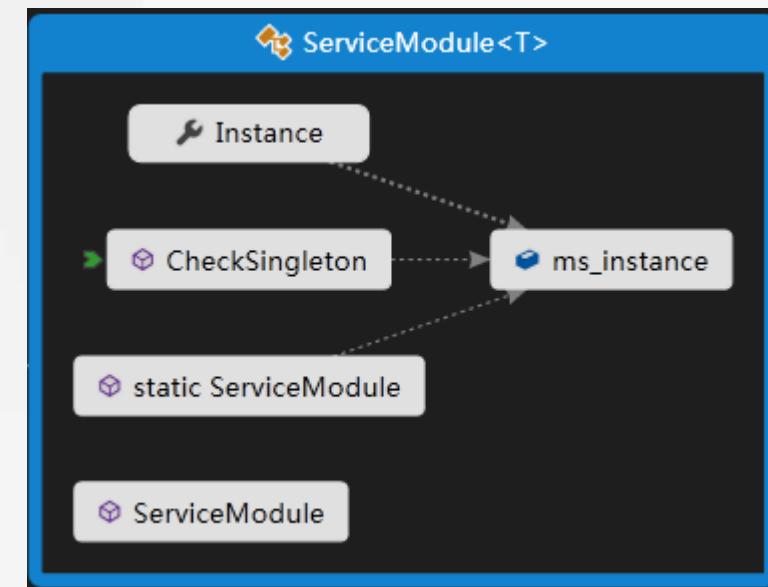
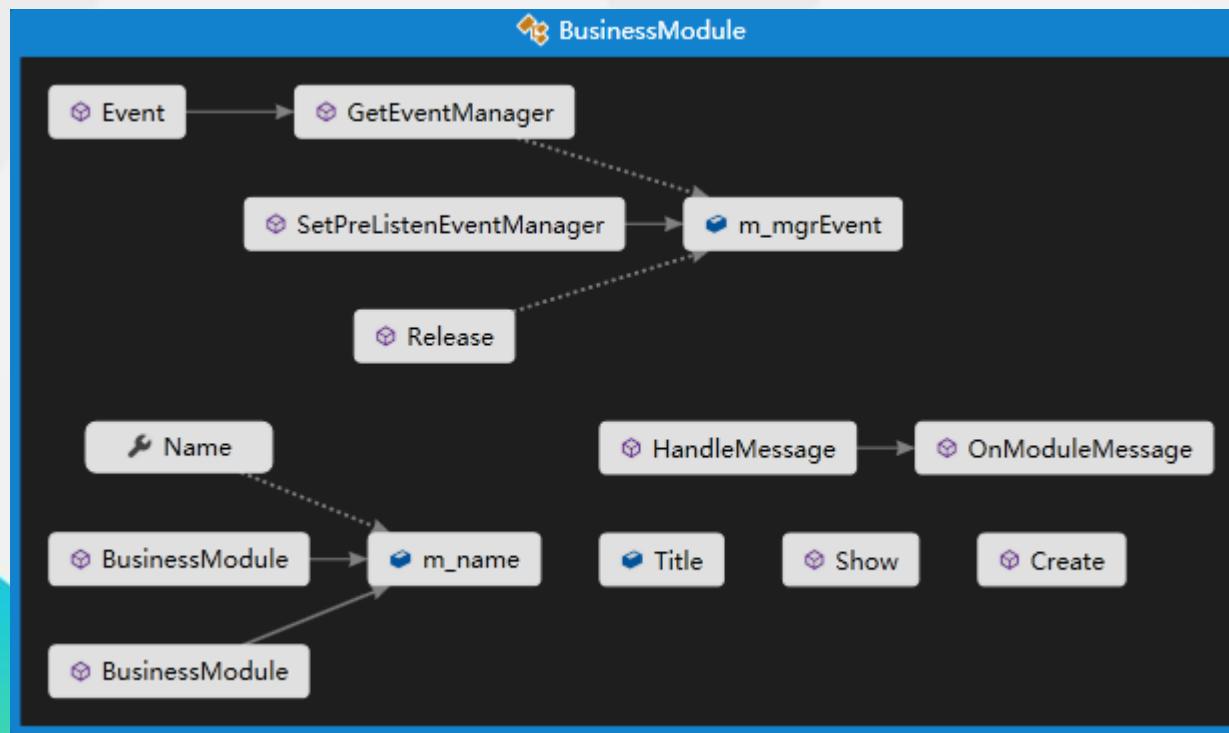
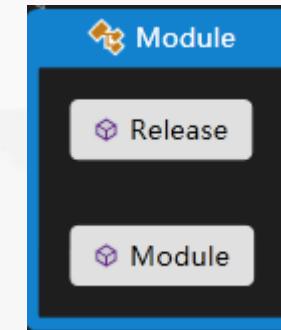
5、让架构先跑起来

- 5.2 实现模块管理器
 - 5.2.1 实现 : EventTable(EventManager)类



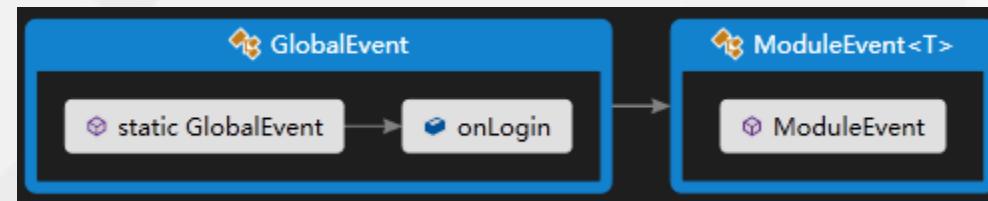
5、让架构先跑起来

- 5.2 实现模块管理器
 - 5.2.2 实现 : Module类
 - 5.2.3 实现 : BusinessModule类
 - 5.2.4 实现 : ServiceModule类



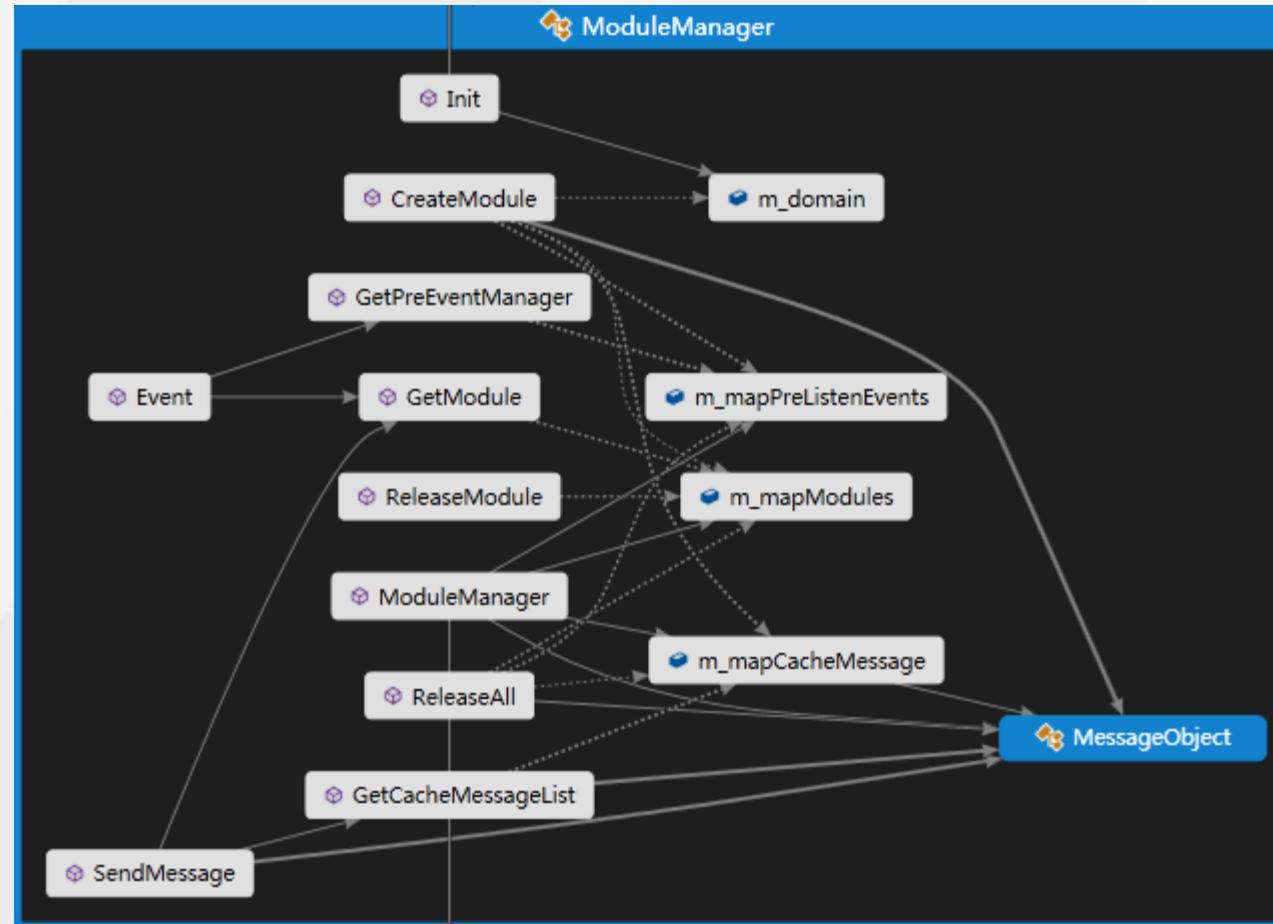
5、让架构先跑起来

- 5.2 实现模块管理器
 - 5.2.5 实现：GlobalEvent类



5、让架构先跑起来

- 5.2 实现模块管理器
 - 5.2.6 实现：ModuleManager类



5、让架构先跑起来

- 5.3 验证架构是否正常工作
 - 5.3.1 实现2个业务模块：ModuleA , ModuleB
 - 5.3.2 实现1个服务模块：ModuleC
 - 5.3.3 验证模块A、B、C之间的调用和通讯

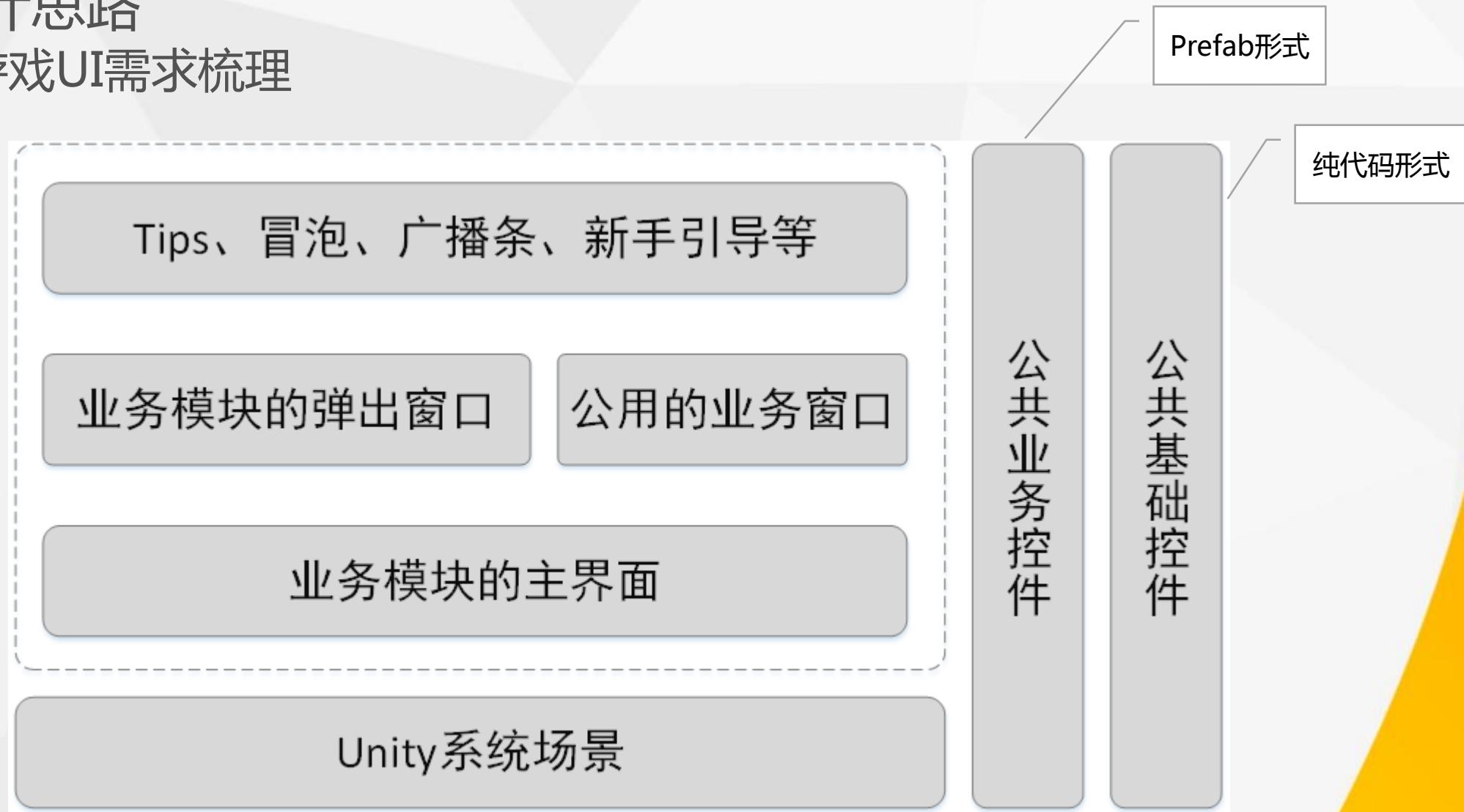
6、UI框架与MVC模式

- 6.1 UI设计思路
- 6.2 UI框架
- 6.3 MVC模式
- 6.4 编码实现及验证

6、UI框架与MVC模式

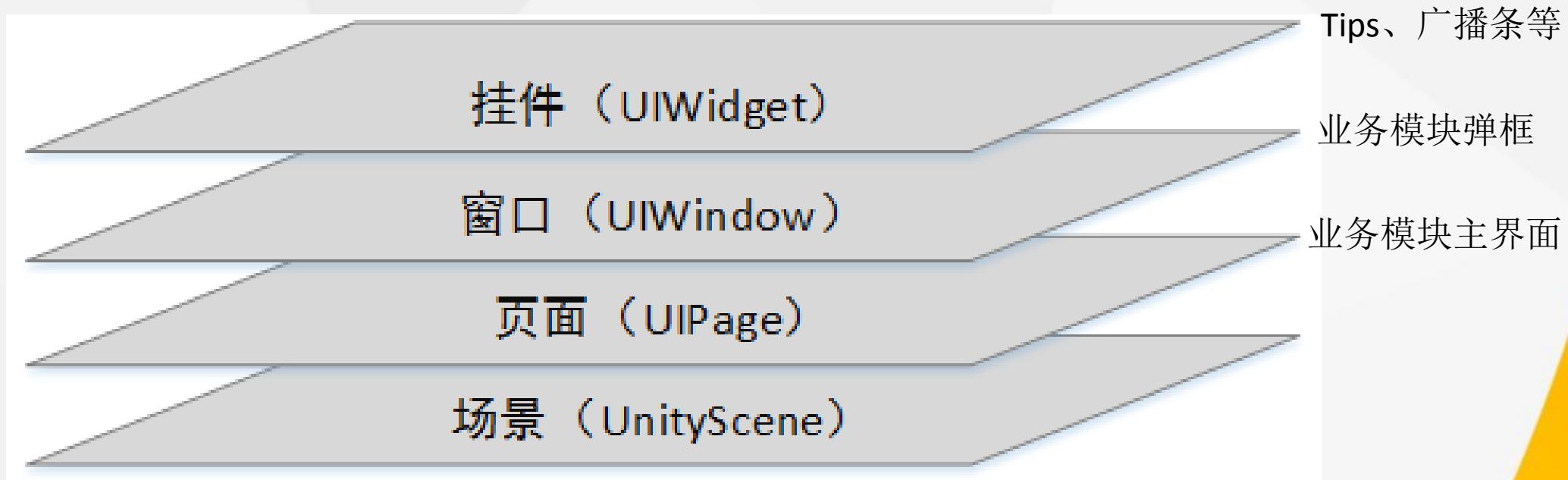
• 6.1 UI设计思路

• 6.1.1 游戏UI需求梳理



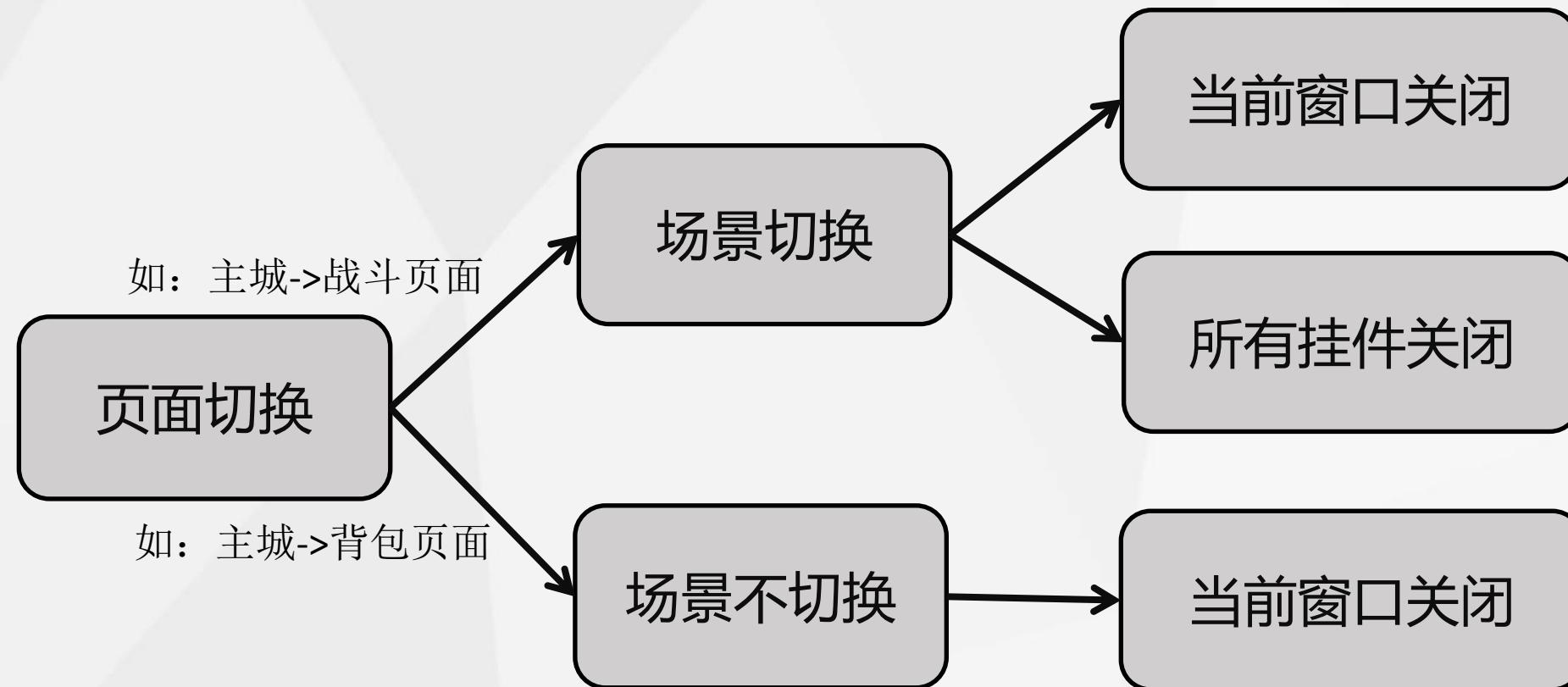
6、UI框架与MVC模式

- 6.1 UI设计思路
 - 6.1.2 UI的分类及层次结构



6、UI框架与MVC模式

- 6.1 UI设计思路
 - 6.1.2 页面切换



6、UI框架与MVC模式

- 6.1 UI设计思路
 - 6.1.2 页面切换



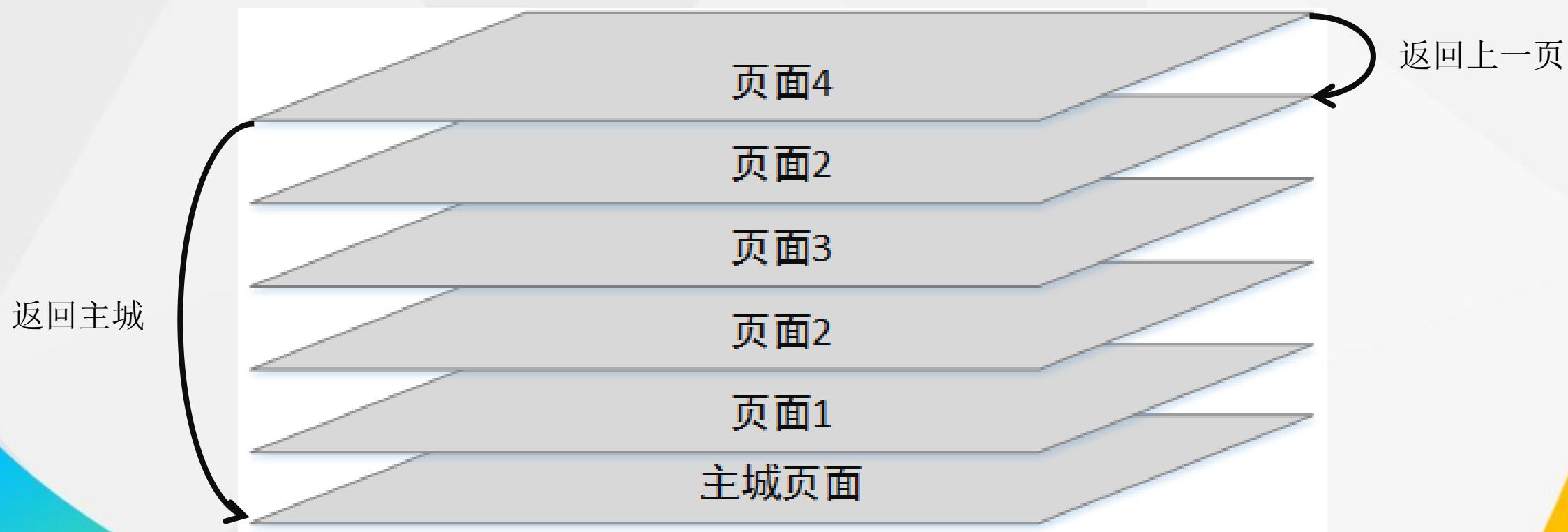
6、UI框架与MVC模式

- 6.1 UI设计思路
 - 6.1.2 页面切换



6、UI框架与MVC模式

- 6.1 UI设计思路
 - 6.1.3 页面返回



6、UI框架与MVC模式

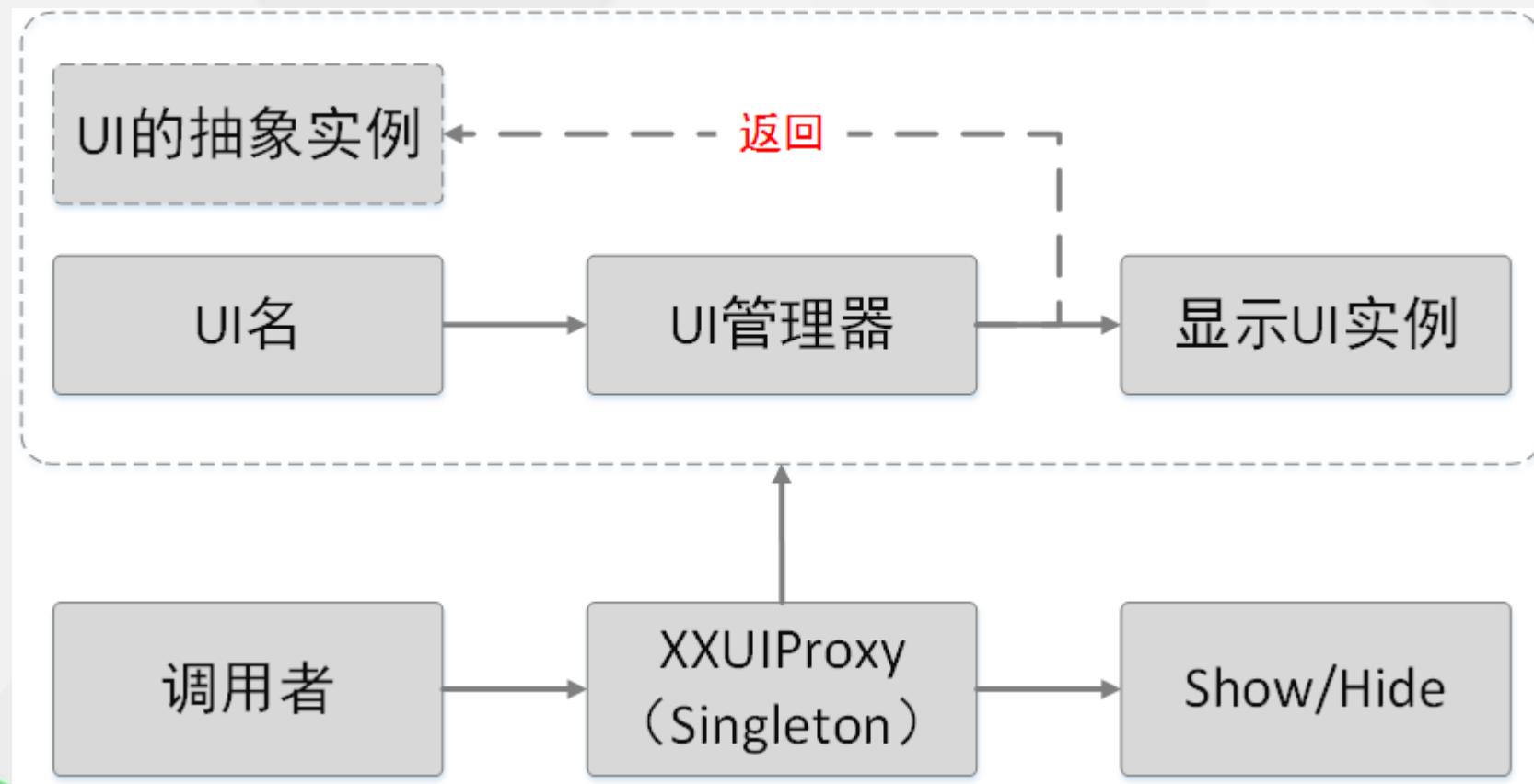
- 6.1 UI设计思路
 - 6.1.4 一般窗口及挂件显示
 - 工厂模式



6、UI框架与MVC模式

- 6.1 UI设计思路

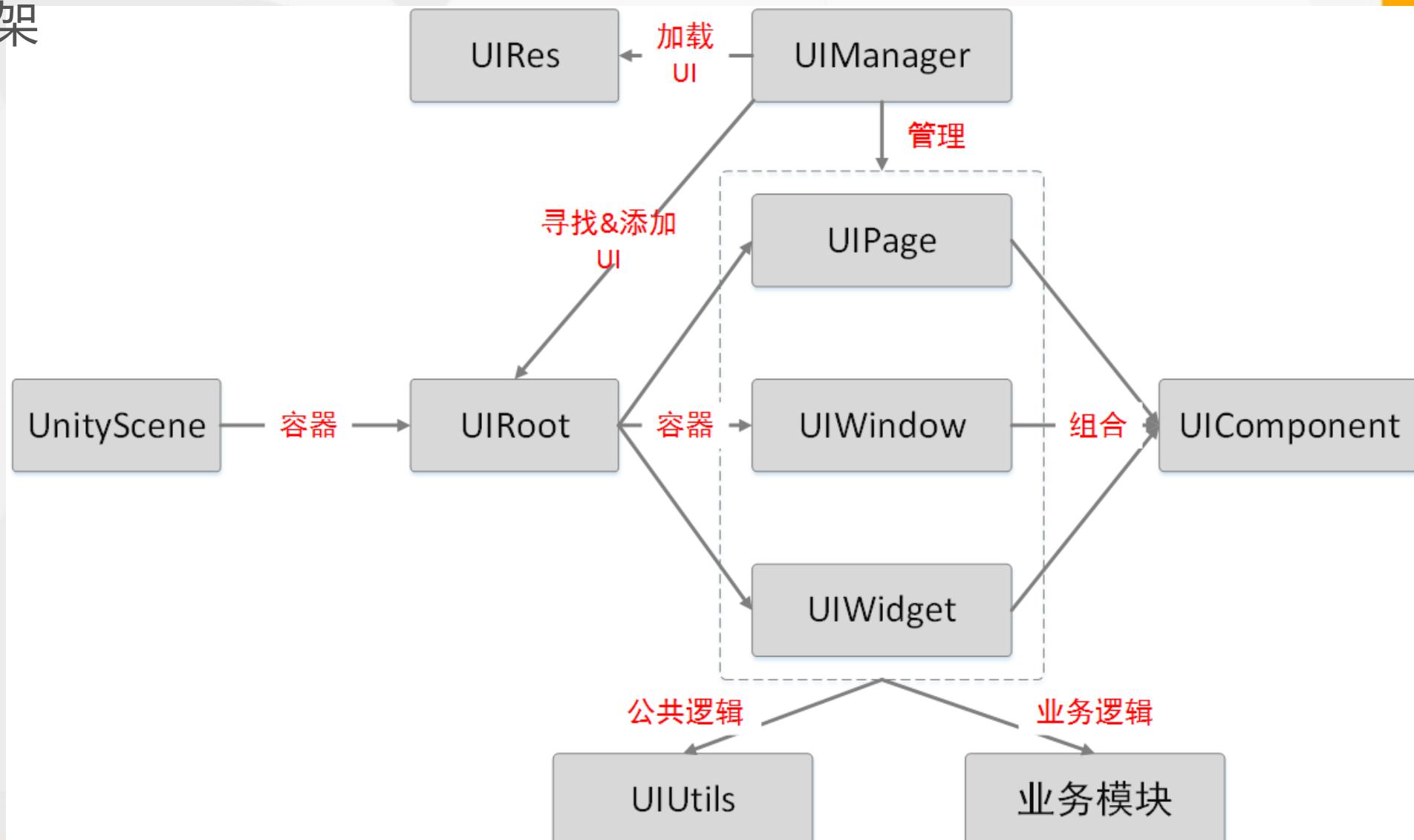
- 6.1.5 公用的UI显示
 - 代理模式+单例模式



6、UI框架与MVC模式

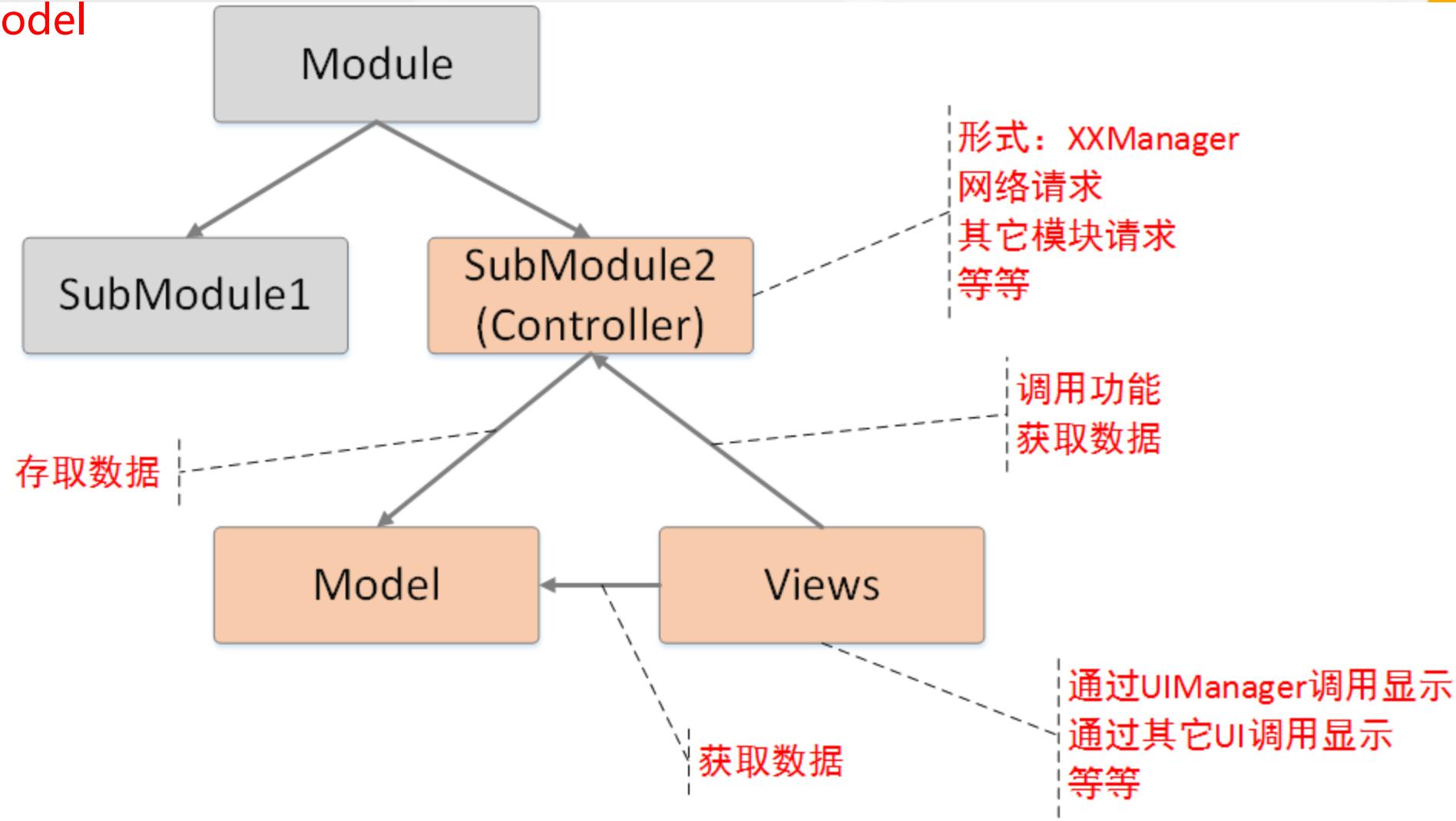
• 6.2 UI框架

• 6.2.1 UI整体框架



6、UI框架与MVC模式

- 6.3 MVC模式
 - 6.3.1 依赖示意图
 - Module != Model



6、UI框架与MVC模式

- 6.3 MVC模式
 - 6.3.2 MVC框架
 - 这是一个用烂的模式
 - 如果你非要弄一个框架出来，那就错了！

水无常形
请勿教条主义

6、UI框架与MVC模式

- 6.4 编码实现

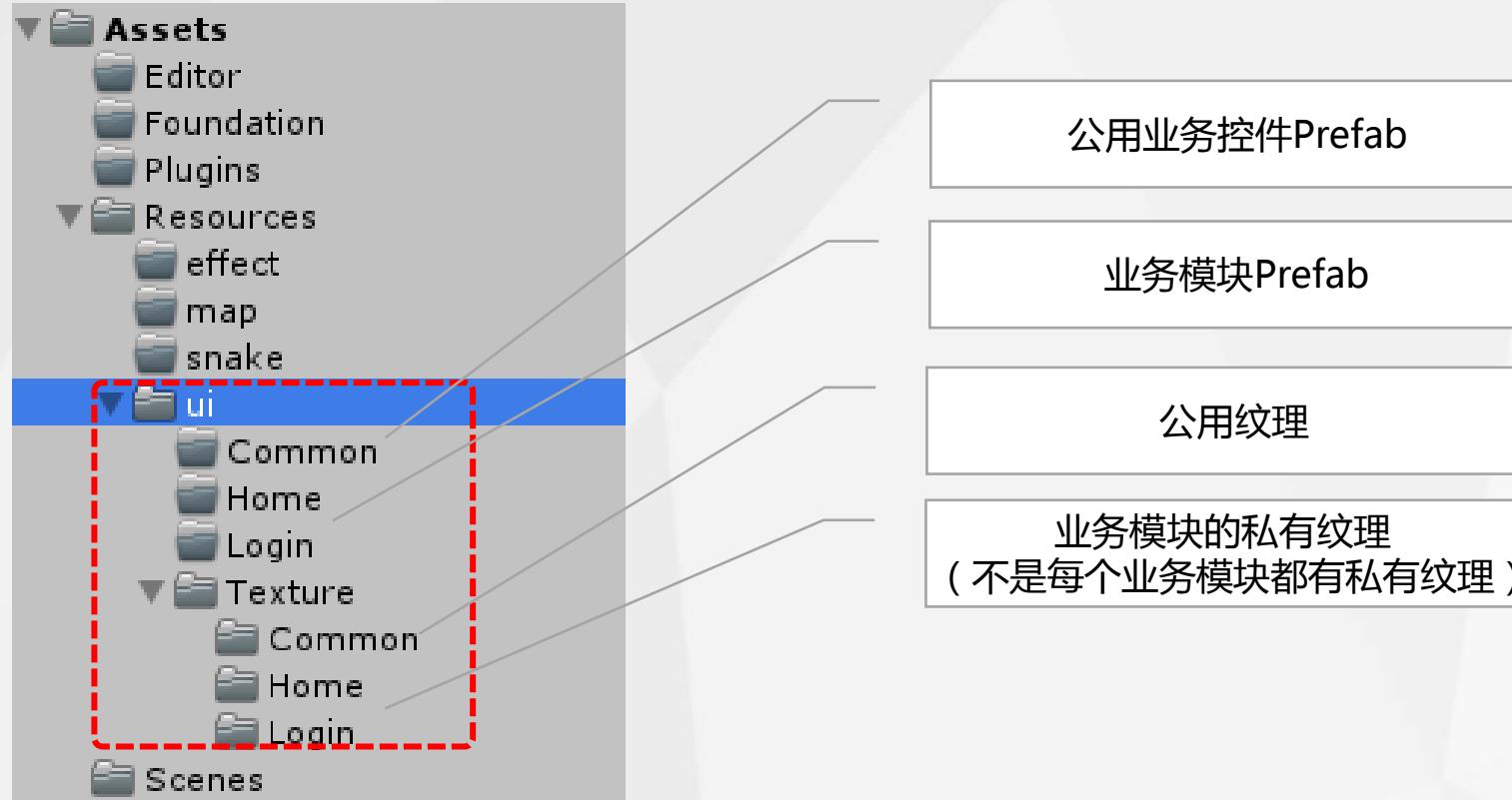
- 6.4.1 实现 : UIPanel, UIPage, UIWindow, UIWidget
- 6.4.2 实现 : UIRoot , UIRes , UIUtils
- 6.4.3 实现 : UIManager

7、主城及相关模块

- 7.1 资源导入
- 7.2 ProtoBuf的引用
- 7.3 主城及相关模块的关系
- 7.4 公用的UI之：UIMsgBox
- 7.5 工具类之：AppConfig
- 7.6 服务层模块之：用户管理模块
- 7.7 自下而上的设计：登录模块
- 7.8 自上而下的设计：主城模块

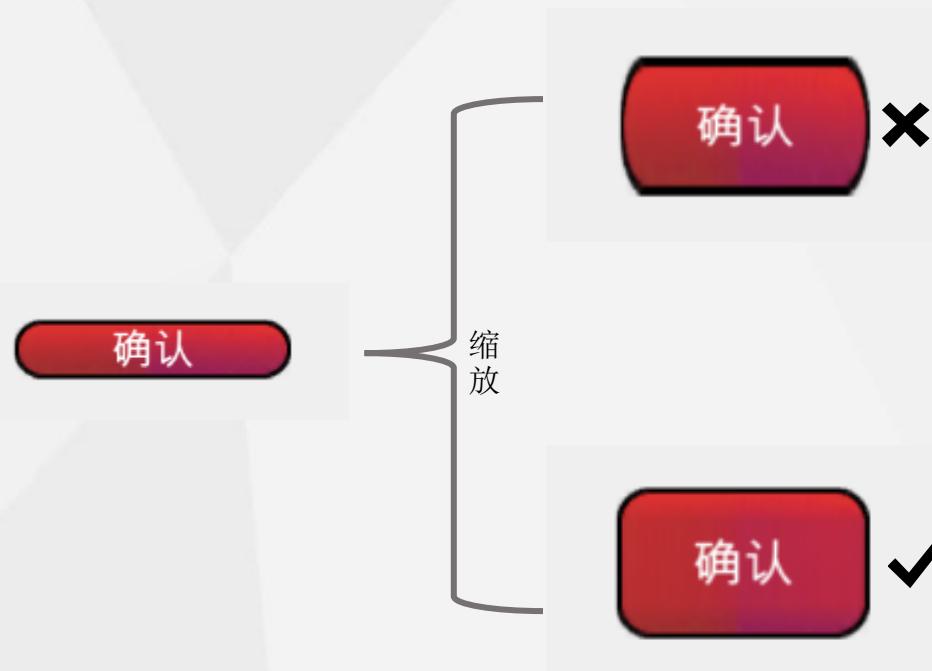
7、主城及相关模块

- 7.1 资源导入
 - 7.1.1 UI资源目录规划



7、主城及相关模块

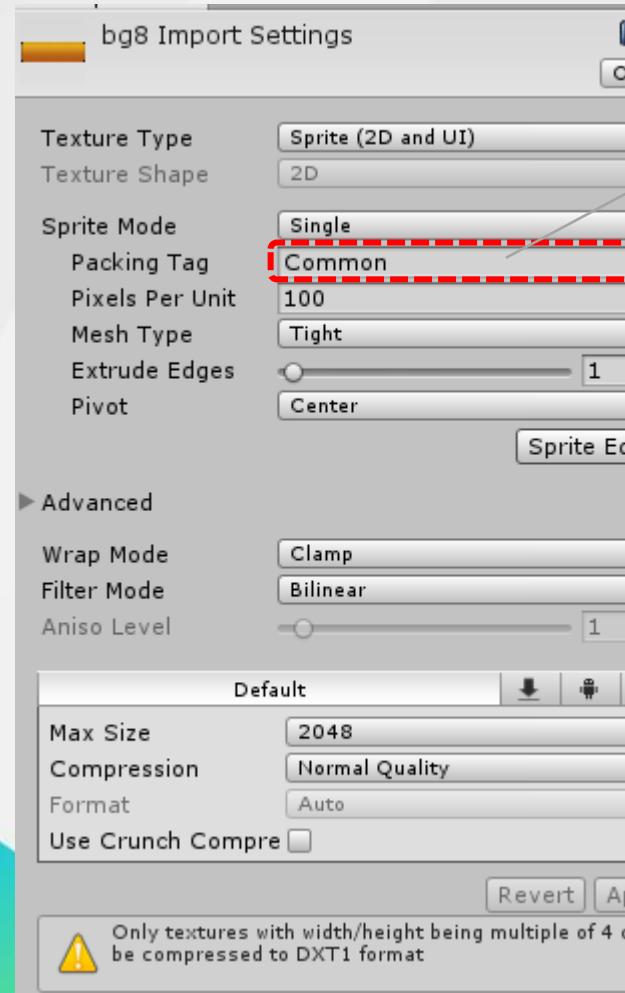
- 7.1 资源导入
 - 7.1.2 九宫格设置



7、主城及相关模块

• 7.1 资源导入

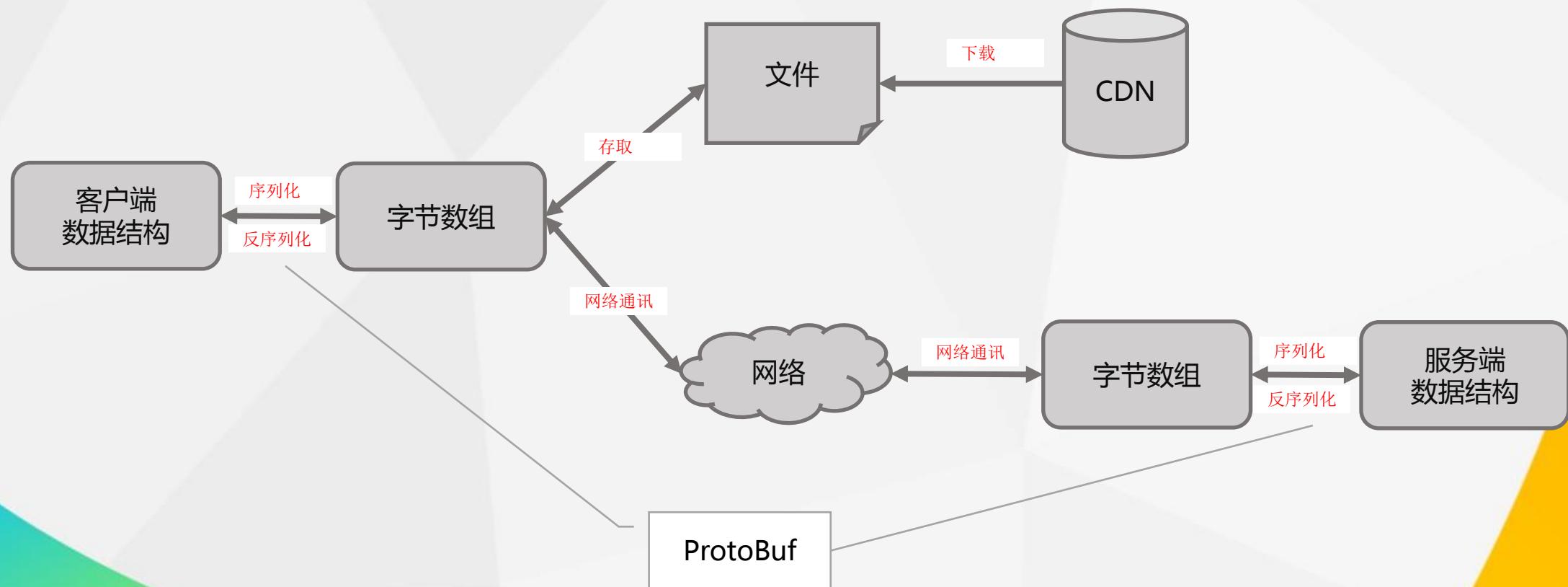
• 7.1.3 PackingTag (Unity Pro版本具有的功能)



用于SpritePacker打包时，将相关
Tag的打成同一个Atlas

7、主城及相关模块

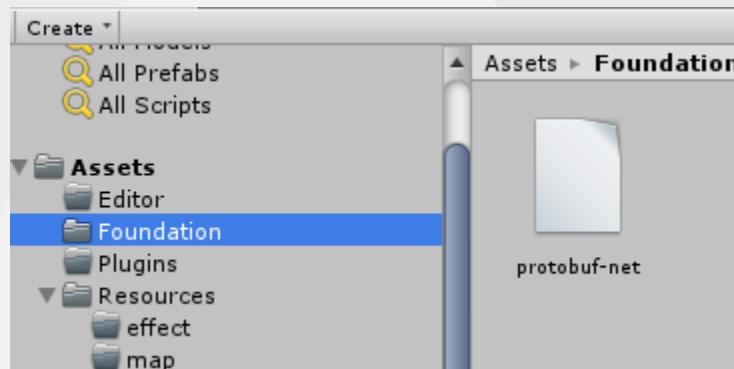
- 7.2 ProtoBuf的引用
 - 7.2.1 ProtoBuf是什么？为什么要使用ProtoBuf？



7、主城及相关模块

- 7.2 ProtoBuf的引用

- 7.2.2 从GitHub下载ProtoBuf的C#源码放入Unity工程
- 7.2.3 或者将编译好的DLL放入Unity工程

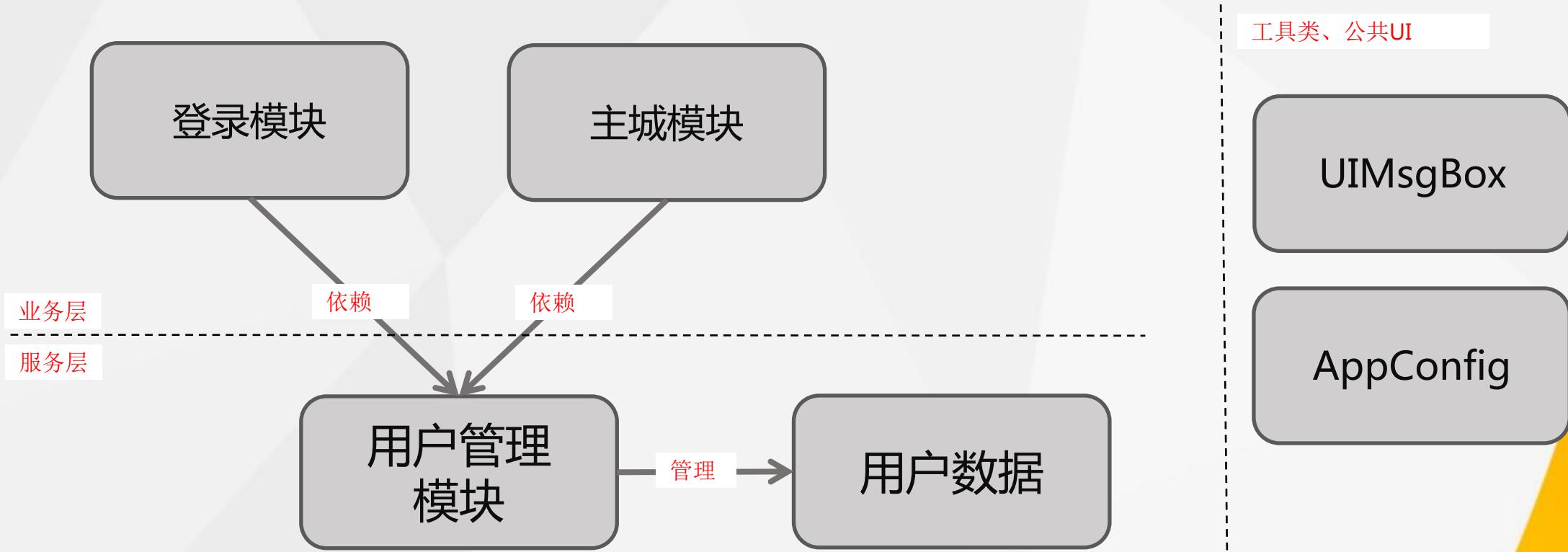


7、主城及相关模块

- 7.2 ProtoBuf的引用
 - 7.2.4 ProtoBuf序列化与反序列化工具类：PBSerializer
 - 7.2.5 ProtoBuf的使用示例

7、主城及相关模块

• 7.3 主城与相关模块的关系

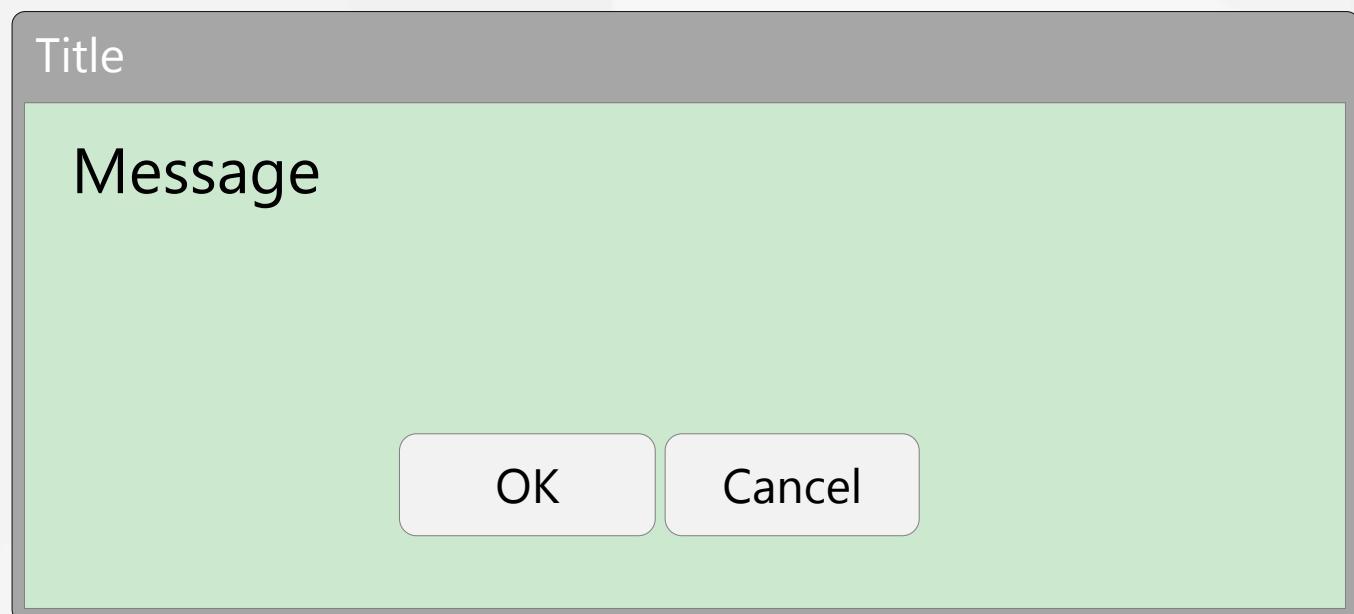


7、主城及相关模块

- 7.4 公用的UI之：UIMsgBox

- 7.4.1 UI定义

- Title : 可选参数
 - Content : 必要参数
 - 按钮的个数，以文本可设置
 - 当MsgBox关闭时，需要回调给调用者点了哪个按钮



7、主城及相关模块

- 7.4 公用的UI之：UIMsgBox
 - 7.4.2 回顾和完善之前的UI框架
 - 为了实现【当MsgBox关闭时，回调给调用者点了哪个按钮】，需要对UI框架进行完善：在CloseEvent里带上关闭的参数

7、主城及相关模块

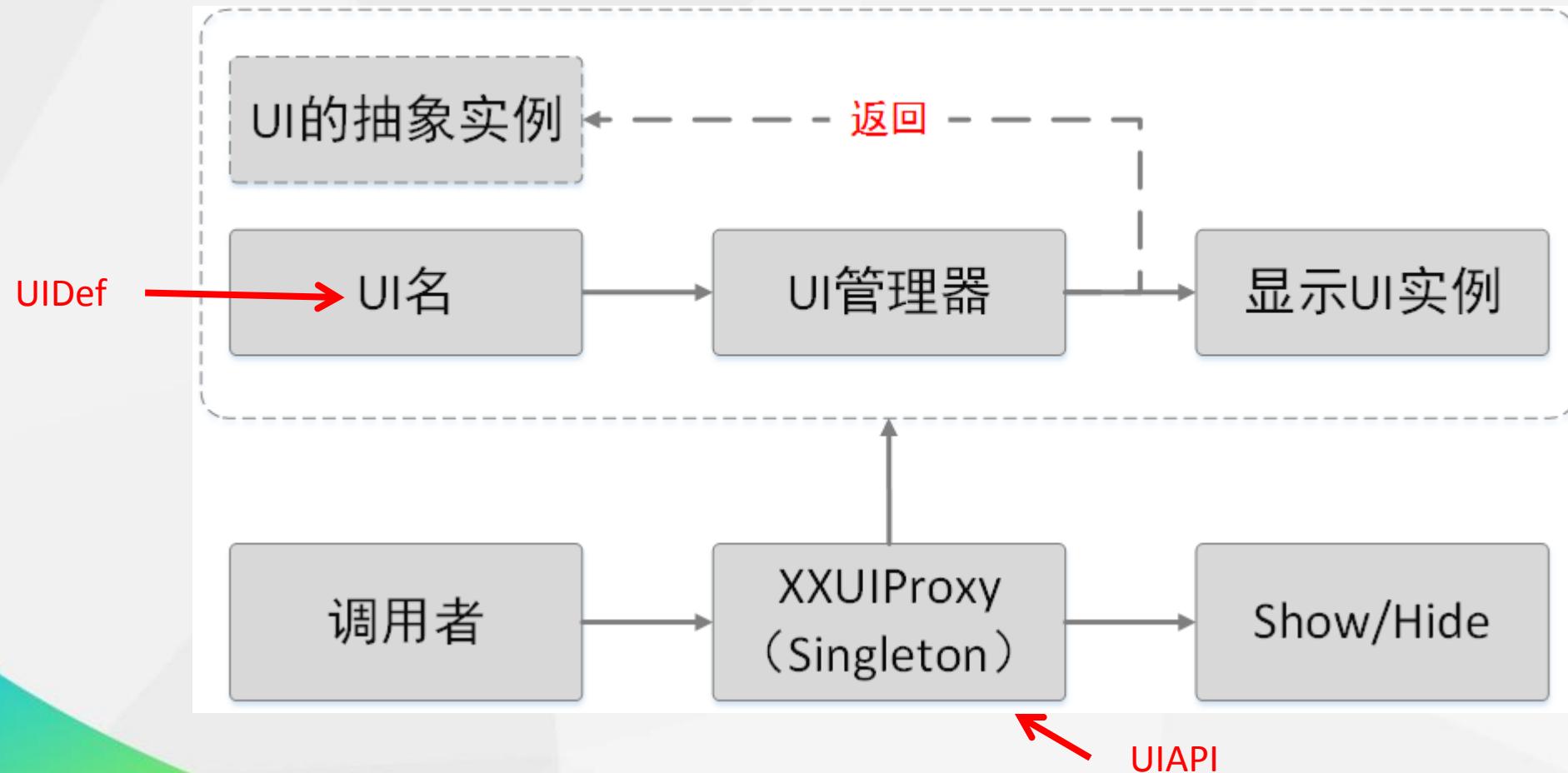
- 7.4 公用的UI之：UIMsgBox

- 7.4.3 UIMsgBox编码实现
- 7.4.4 UIMsgBox.prefab制作



7、主城及相关模块

- 7.4 公用的UI之：UIMsgBox
 - 7.4.5 UIAPI与UIDef



7、主城及相关模块

- 7.4 公用的UI之：UIMsgBox
 - 7.4.6 使用示例



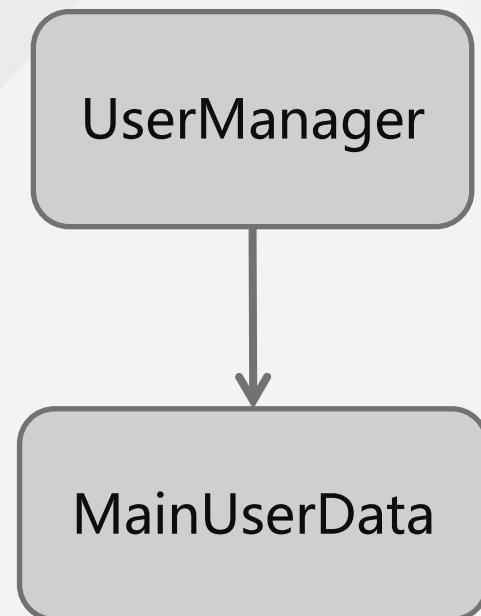
7、主城及相关模块

- 7.5 工具类之： AppConfig
 - 用来保存App相关配置，比如上一次登录的用户ID等

```
/// <summary>
/// App的配置定义
/// </summary>
[ProtoContract]
10 个引用
public class AppConfig
{
    /// <summary>
    /// 主用户数据
    /// </summary>
    [ProtoMember(1)] public UserData mainUserData = new UserData();
    [ProtoMember(2)] public bool enableBgMusic = true;
    [ProtoMember(3)] public bool enableSoundEffect = true;
```

7、主城及相关模块

- 7.6 服务层模块之：用户管理模块
 - 7.6.1 定义数据结构：UserData
 - 7.6.2 定义逻辑接口：UserManager



7、主城及相关模块

- 7.7 自下而上的设计：登录模块
 - 7.7.1 模块逻辑类：LoginModule
 - Login(id, name, pwd)

7、主城及相关模块

- 7.7 自下而上的设计：登录模块
 - 7.7.2 UI脚本类：UILoginPage
 - OnBtnLogin()

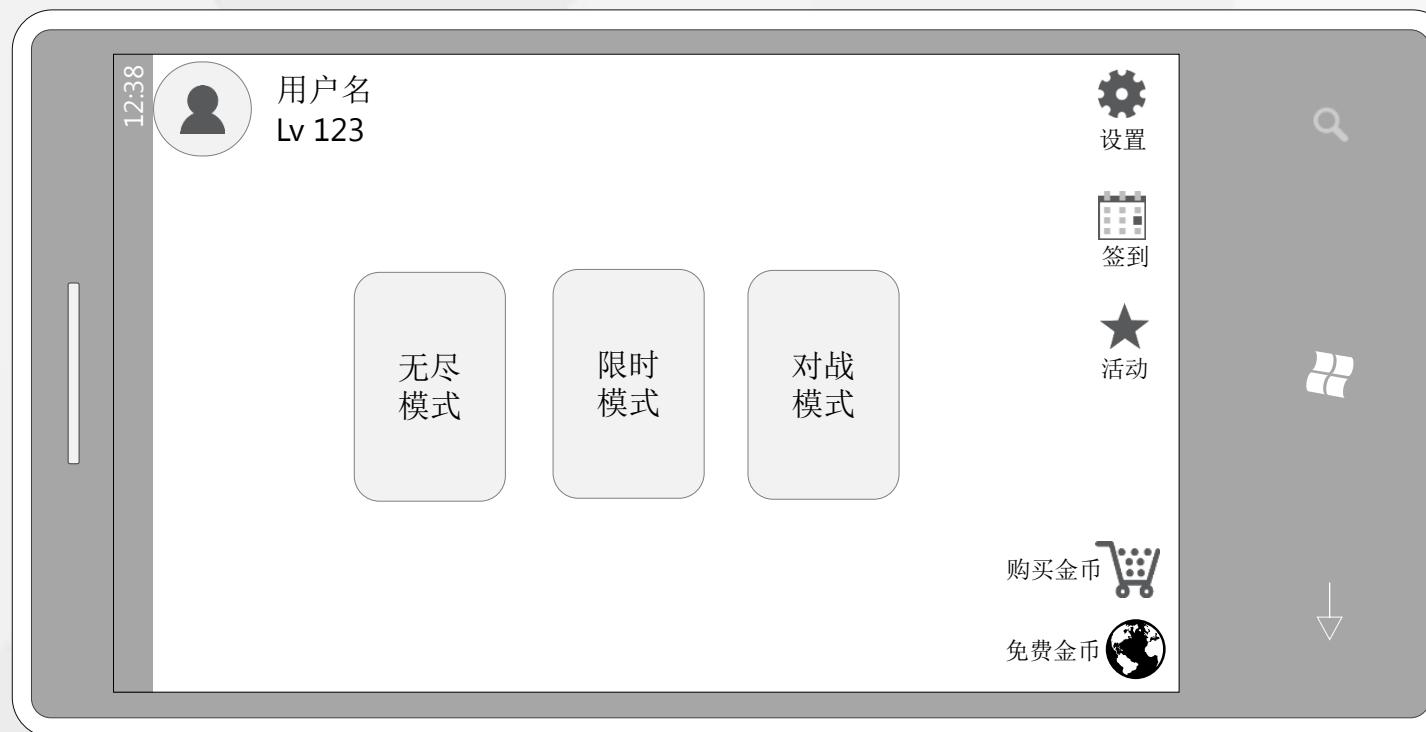
7、主城及相关模块

- 7.7 自下而上的设计：登录模块
 - 7.7.3 UI制作：UILoginPage.prefab



7、主城及相关模块

- 7.8 自上而下的设计：主城模块
 - 7.8.1 UI制作：UIHomePage.Prefab



7、主城及相关模块

- 7.8 自上而下的设计：主城模块

- 7.8.2 UI脚本：UIHomePage

- OnBtnSetting()
 - OnBtnDailyCheckIn()
 - OnBtnActivity()
 - OnBtnBuyCoin()
 - OnBtnFreeCoin()
 - OnBtnEndlessPVE()
 - OnBtnTimelimitPVE()
 - OnBtnPVP()
 - OnBtnUserInfo()

7、主城及相关模块

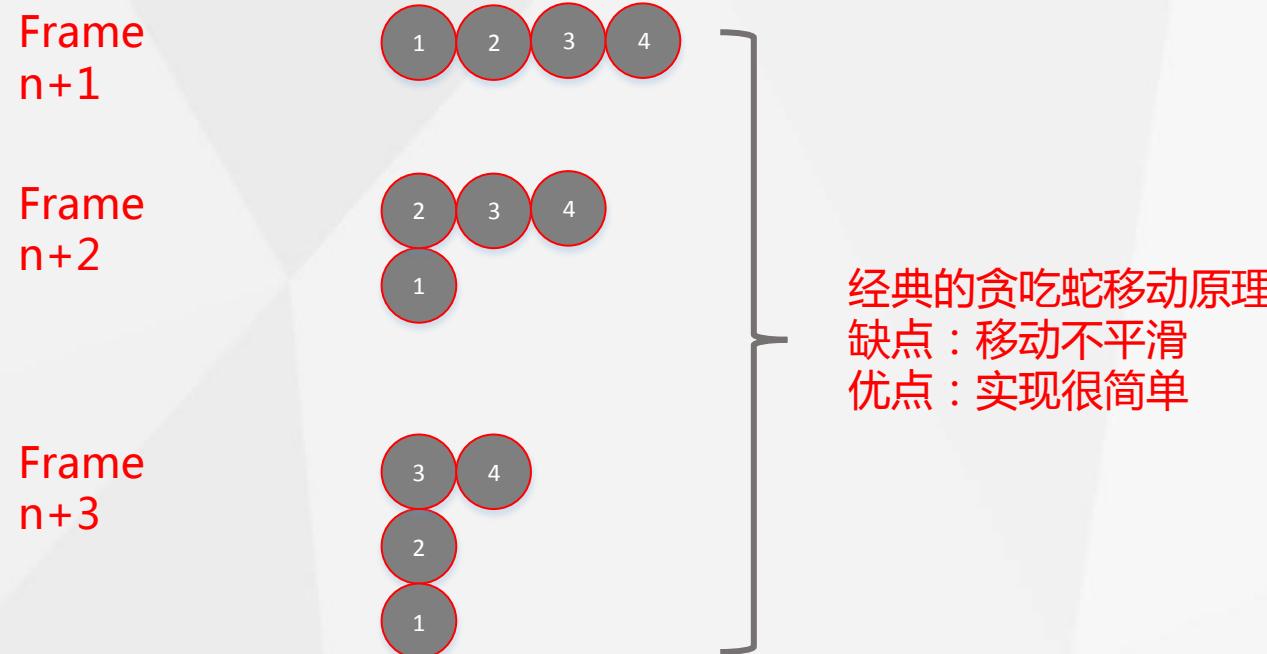
- 7.8 自上而下的设计：主城模块
 - 7.8.3 模块逻辑类：HomeModule
 - OpenModule(name)
 - 7.8.4 模块名定义：ModuleDef

8、核心玩法

- 8.1 设计思路
- 8.2 整体框架
- 8.3 数据定义
- 8.4 核心对象
- 8.5 玩家模块
- 8.6 地图模块
- 8.7 输入模块
- 8.8 游戏管理器
- 8.9 资源导入

8、核心玩法

- 8.1 设计思路
 - 8.1.1 蛇的移动
 - 蛇的后一个结点移到前一个结点坐标

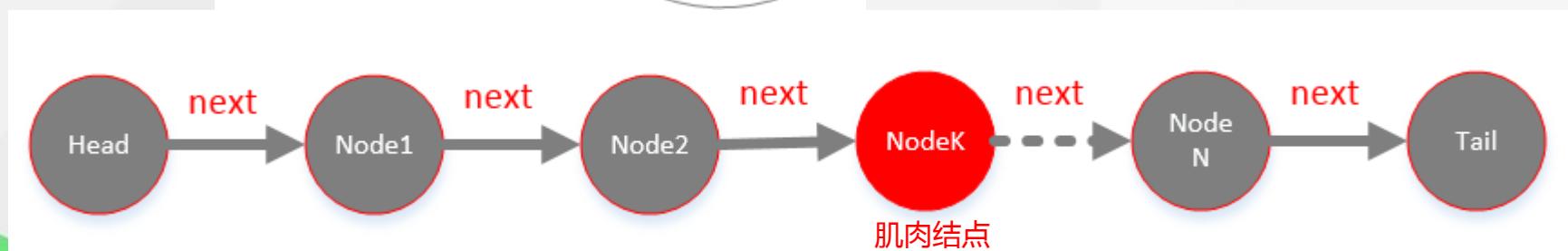
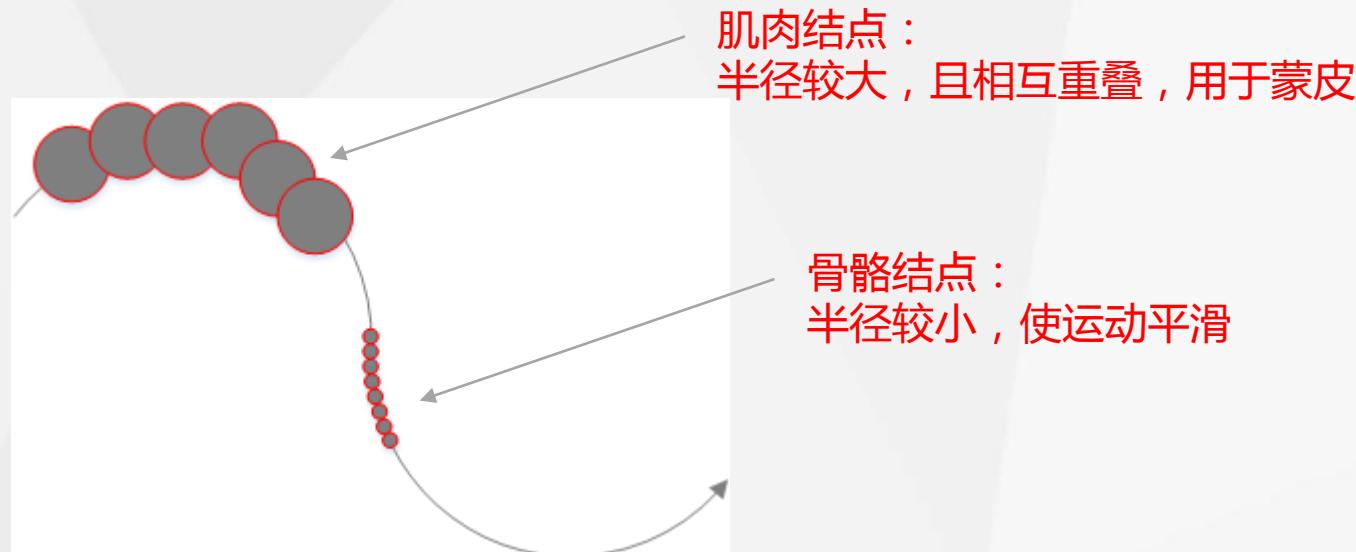


8、核心玩法

• 8.1 设计思路

• 8.1.2 蛇的结构

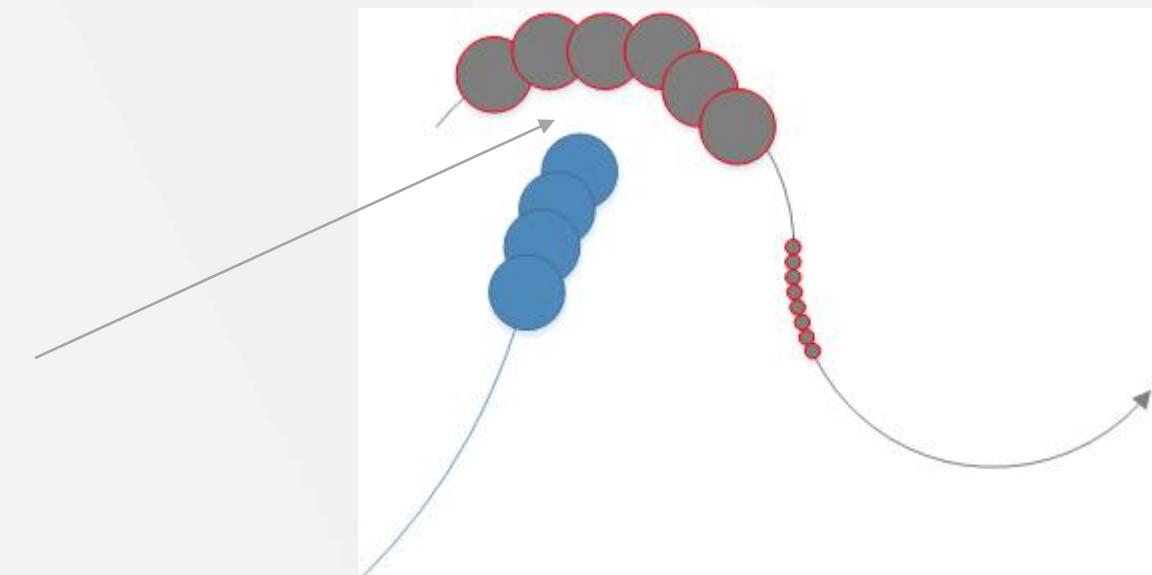
- 看不见的：骨骼链，控制运动过程
- 看得见的：肌肉链，控制视觉表现，以及碰撞



8、核心玩法

- 8.1 设计思路
 - 8.1.3 蛇的碰撞检测

以【肌肉结点】进行碰撞检测
采用【2个圆的相交算法】替代Unity物理引擎

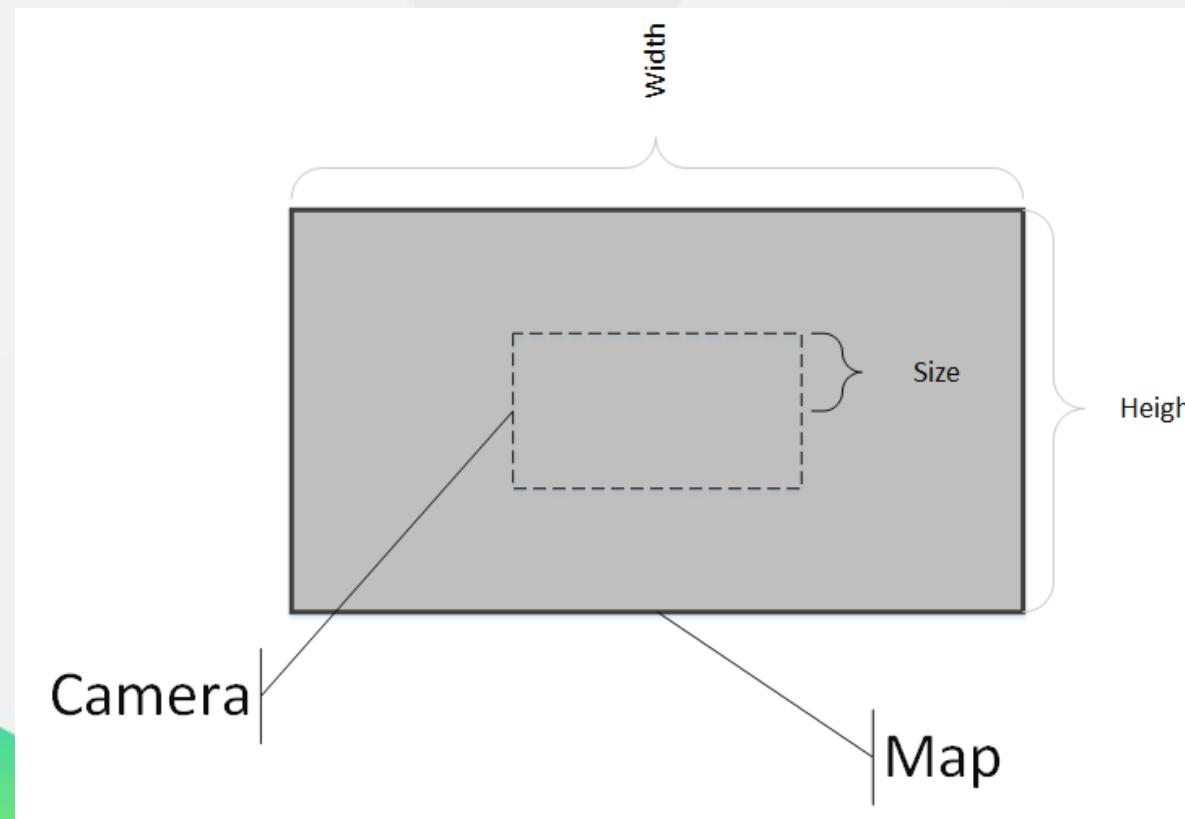


8、核心玩法

• 8.1 设计思路

• 8.1.4 地图的基准尺寸

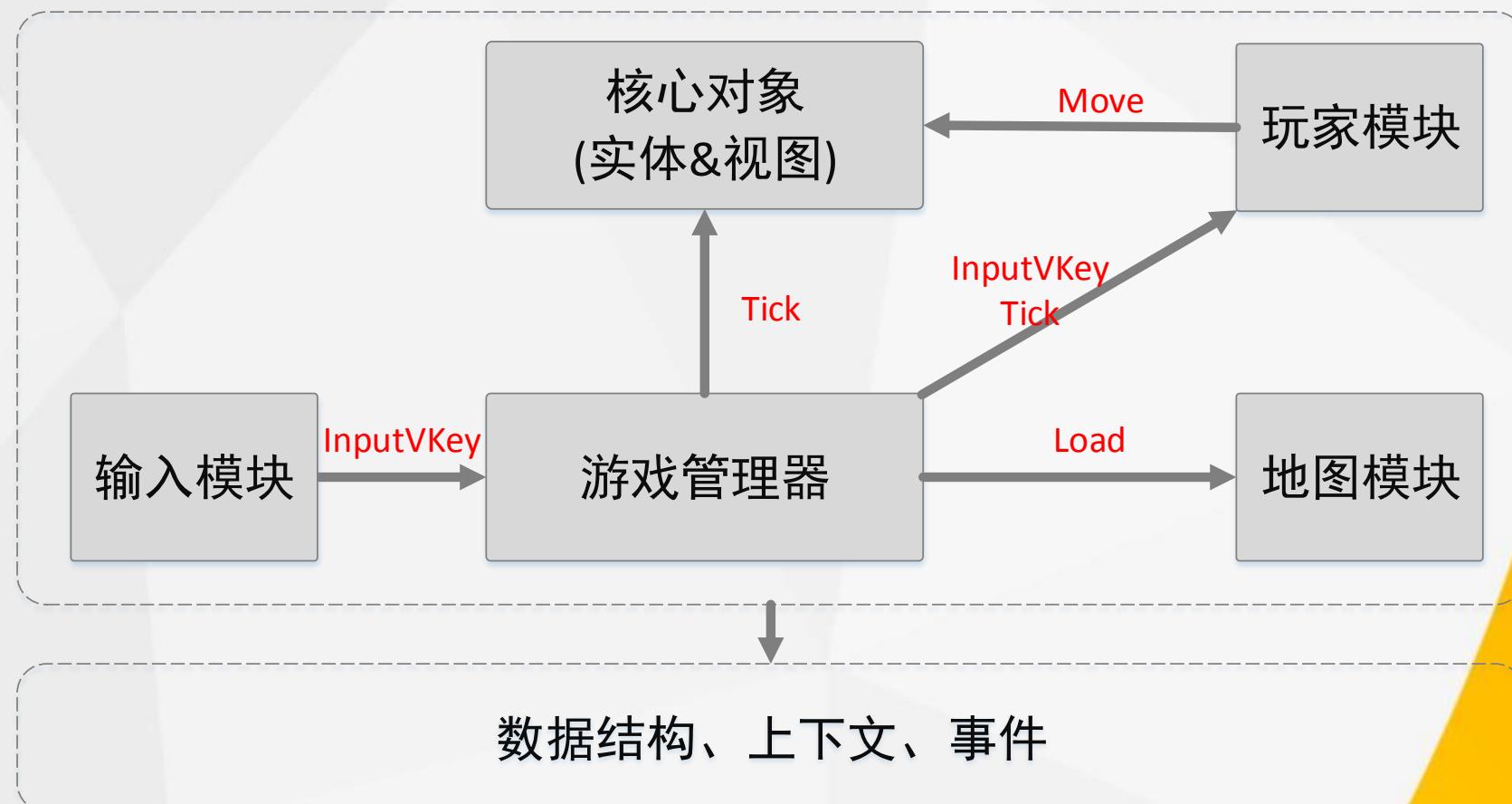
- 地图宽高为一屏的2倍，如下图，Camera一屏只显示地图的1/4。
- 一屏内纵向可显示最多30个蛇的结点，以达到视觉的饱满感
- 蛇的基准尺寸是32像素，地图的基准高为960像素



地图高=30*蛇视图结点半径*2

8、核心玩法

- 8.2 整体框架
 - 8.2.1 模块划分及依赖关系

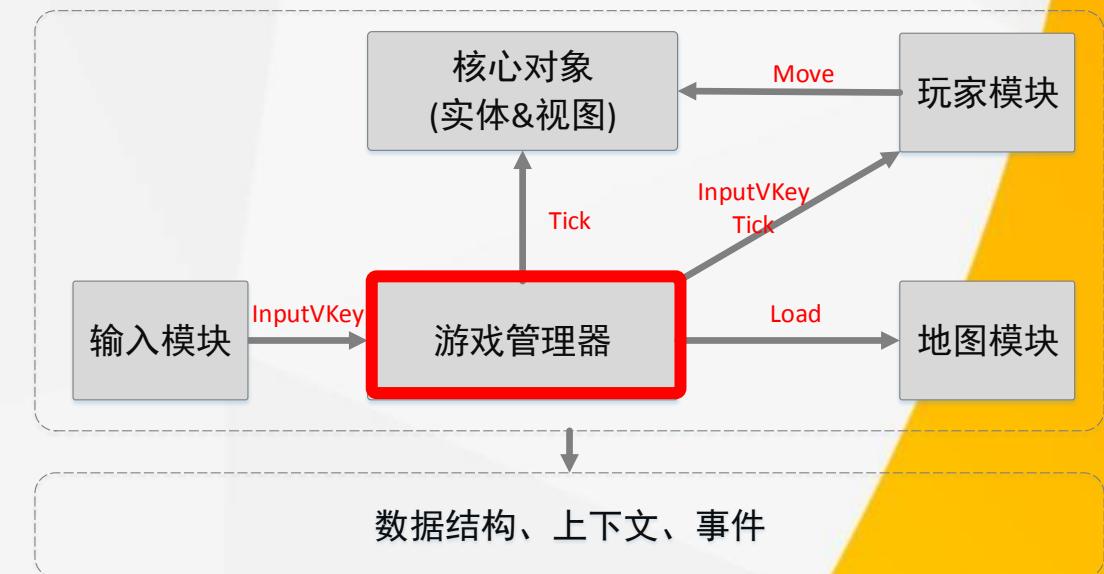


8、核心玩法

• 8.3 数据定义

• 8.3.1 GameParam

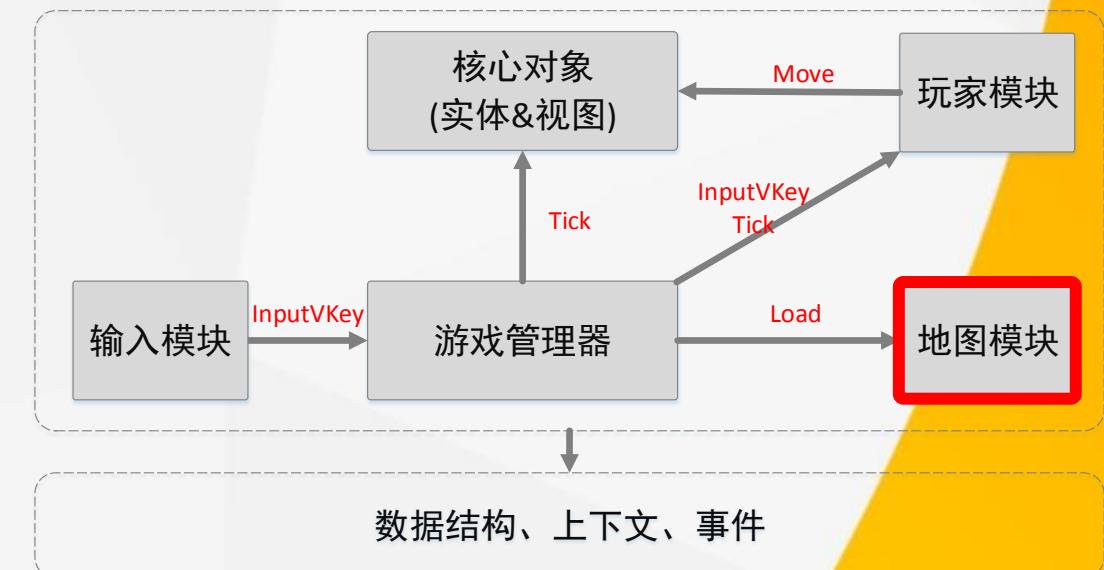
- 启动核心玩法所需要的参数
- id : 后台给每一局游戏的编号
- mapData : 地图数据
- randSeed : 随机种子 , 用来保证不同客户端随机序列一致
- mode : 游戏模式 , 比如PVP , PVE , 限时模式 , 无尽模式 , 等等
- limitedTime : 如果限时模式 , 限时时间



8、核心玩法

- 8.3 数据定义
 - 8.3.2 MapData

- 地图数据，目前暂时只需要这些数据
- id：地图的编号，通过它加载地图资源
- name：地图名称，用于快速显示在地图列表中

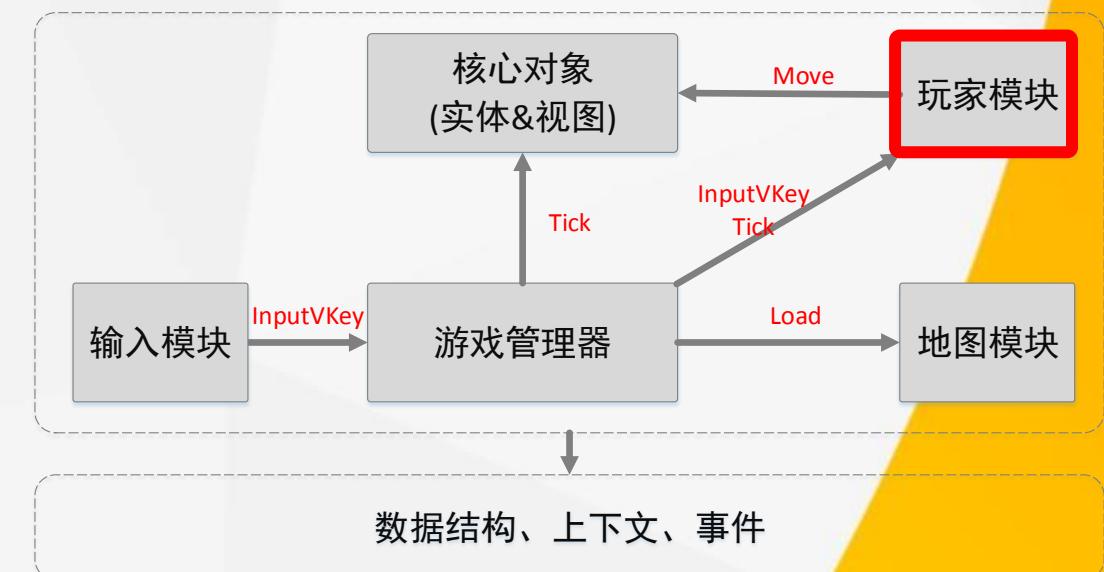


8、核心玩法

• 8.3 数据定义

• 8.3.3 PlayerData

- 玩家数据，目前暂时只需要这些数据
- Id：单局玩家ID，这个ID是单局分配的，从1开始
- userId：用户ID，这个ID是全局分配的，用于映射到UserData
- Name：玩家的名字，拷贝至UserData，为了在游戏中快速显示名字
- snakeData：蛇的数据，即核心对象相关的数据
- teamId：如果是组队游戏，则有一个组队ID
- Score：单局积分
- Ai：如果玩家是AI控制的，代表AI的编号

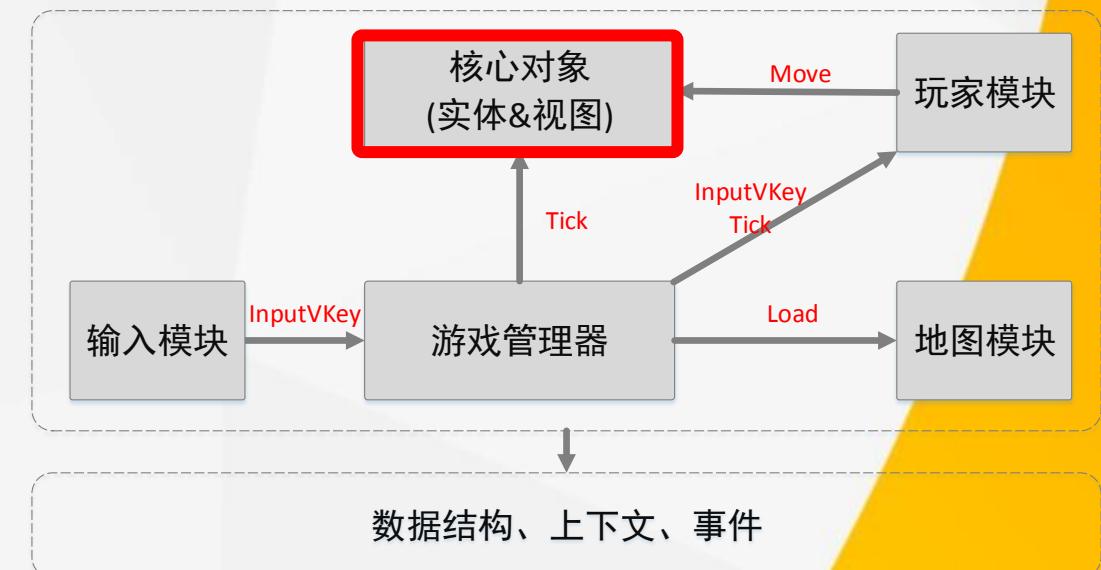


8、核心玩法

• 8.3 数据定义

• 8.3.4 SnakeData

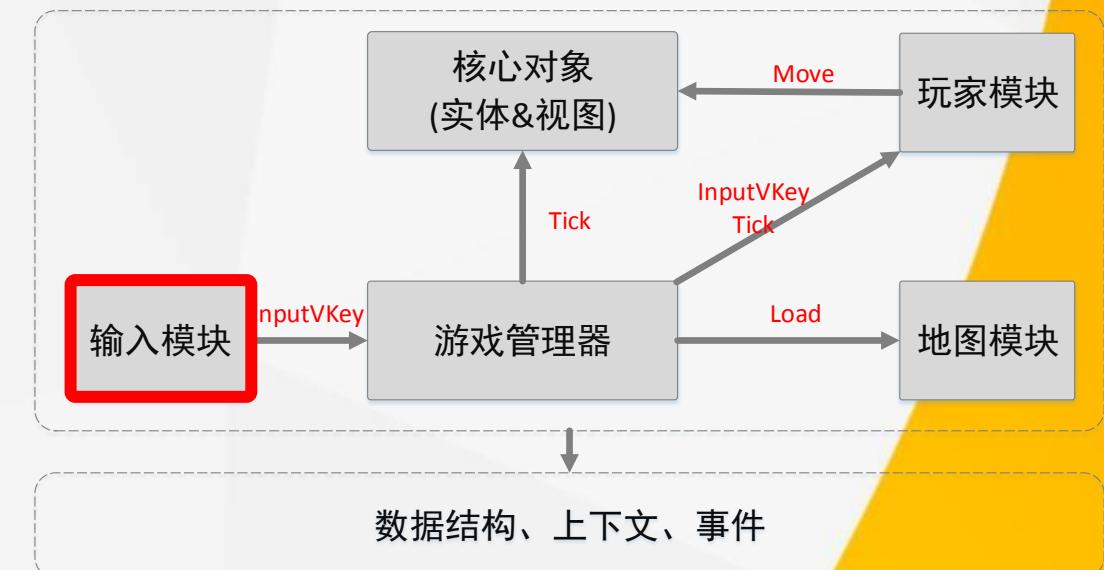
- 核心对象之一，蛇的数据
- Id：表示这是一条什么样的蛇，通过它加载蛇的资源和配置（如果有的话）
- Name：蛇的名字
- size：蛇的大小，即蛇的肌肉结点大小，基准值为32（像素）
- keyStep：关键结点的步长。它影响蛇的【视觉平滑】
- length：蛇的长度，即骨骼数量，也即链表的长度



8、核心玩法

- 8.3 数据定义
 - 8.3.5 GameVKey

- 将玩家的操作封装为Vkey【虚拟按键】
- MoveX
- MoveY
- SpeedUp
- CreatePlayer



8、核心玩法

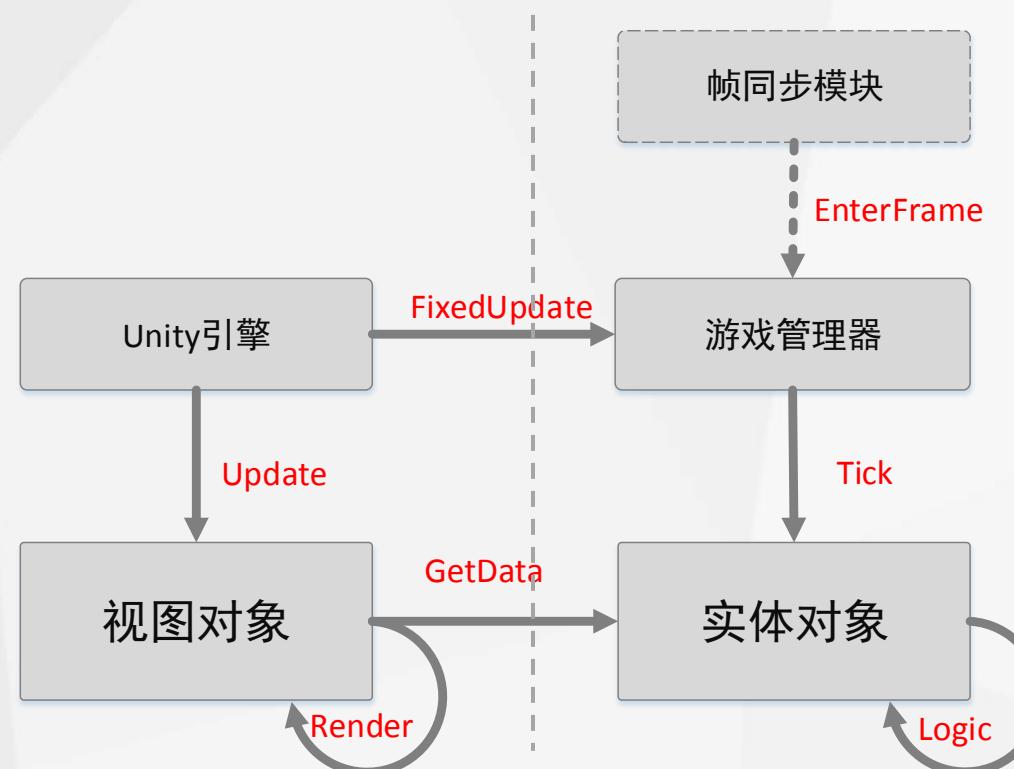
- 8.4 核心对象
 - 8.4.1 有哪些核心对象
 - 蛇（实体、视图）
 - 食物（实体、视图）

8、核心玩法

• 8.4 核心对象

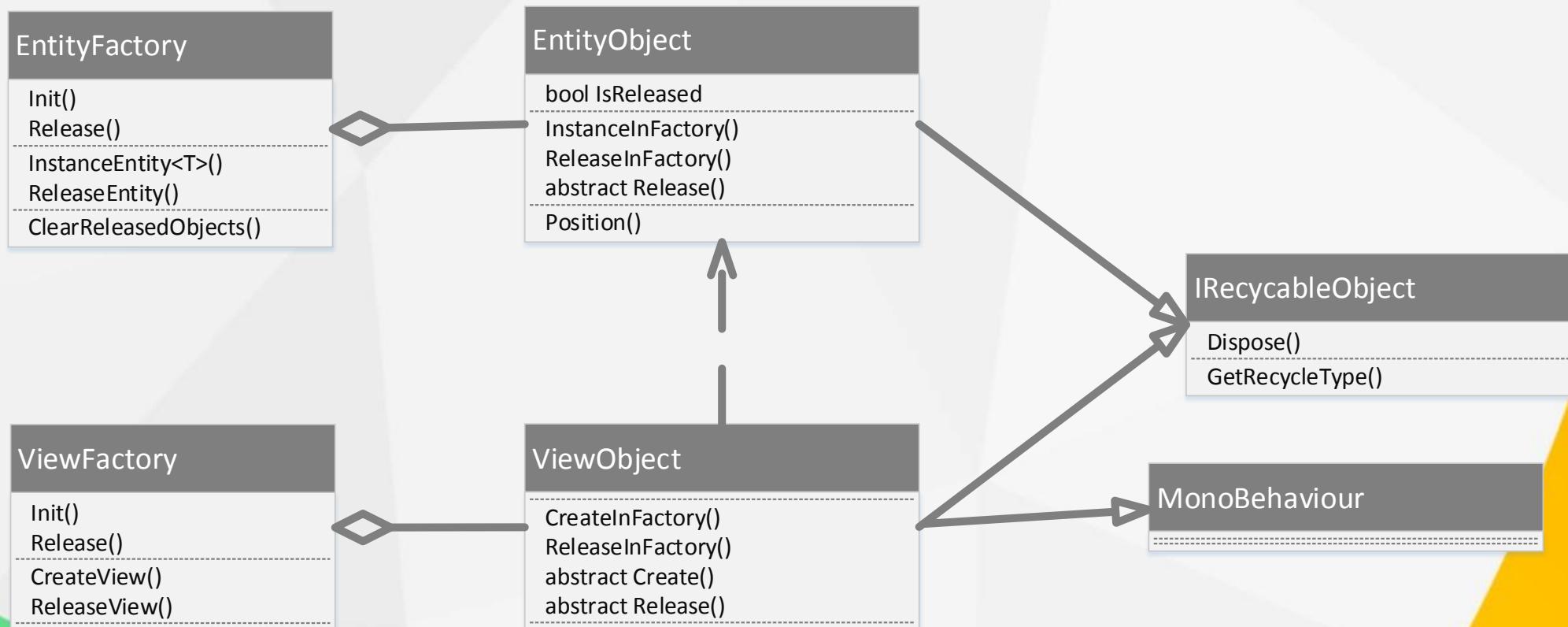
• 8.4.2 实体对象与视图对象

- 【游戏逻辑】由【实体对象】封装和执行
- 【实体对象】从【对象工厂】创建和释放【视图对象】
- 【视图对象】从【实体对象】获取数据进行渲染



8、核心玩法

- 8.4 核心对象
 - 8.4.3 对象工厂
 - 实体对象工厂
 - 视图对象工厂



8、核心玩法

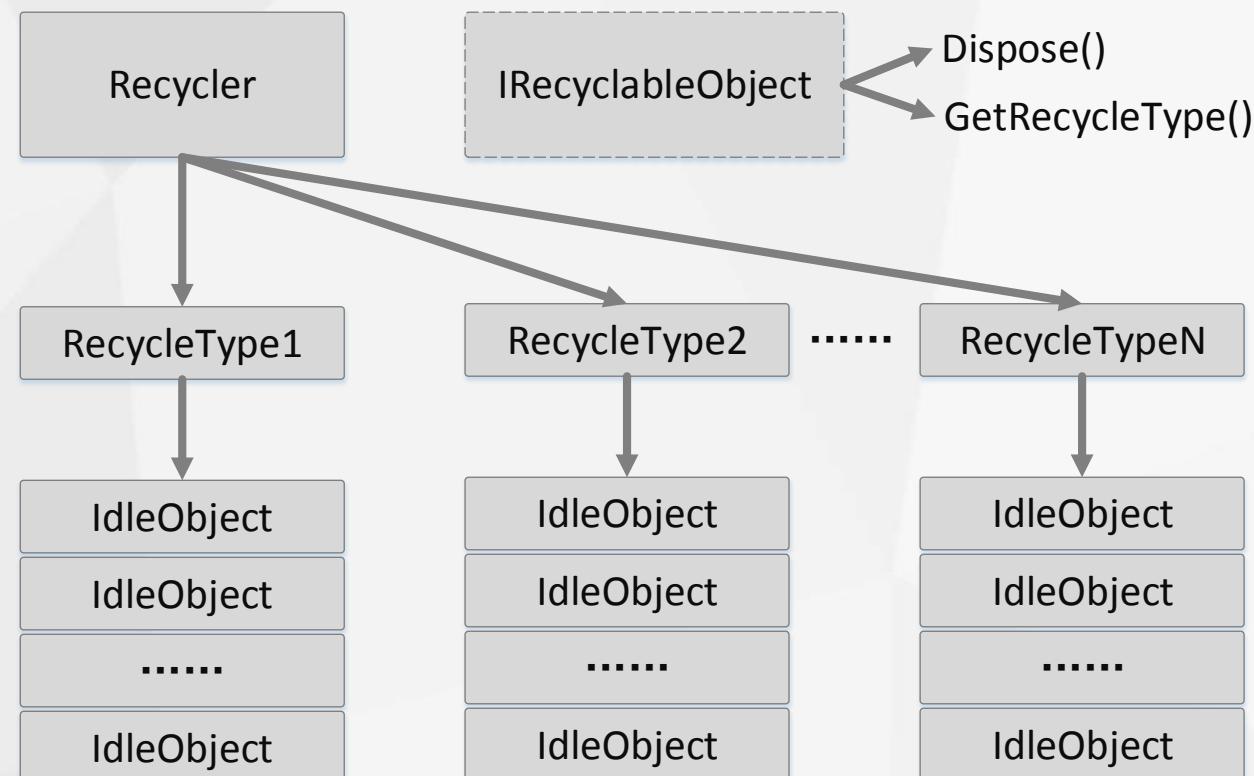
- 8.4 核心对象

- 8.4.4 对象回收器

- 对象回收类型 (RecycleType)

- 实体对象：类名

- 视图对象：资源路径

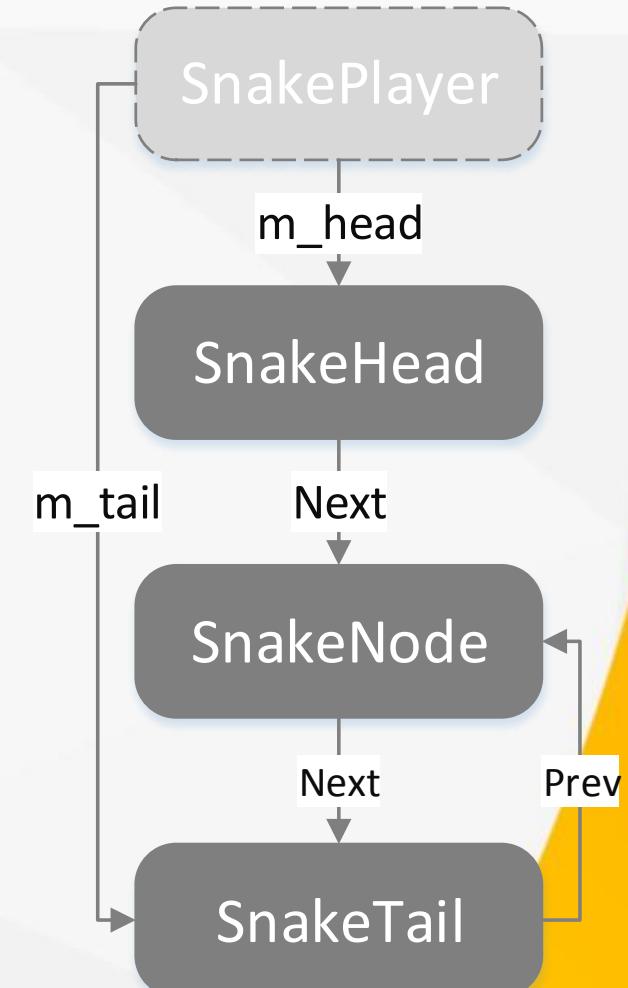


8、核心玩法

• 8.4 核心对象

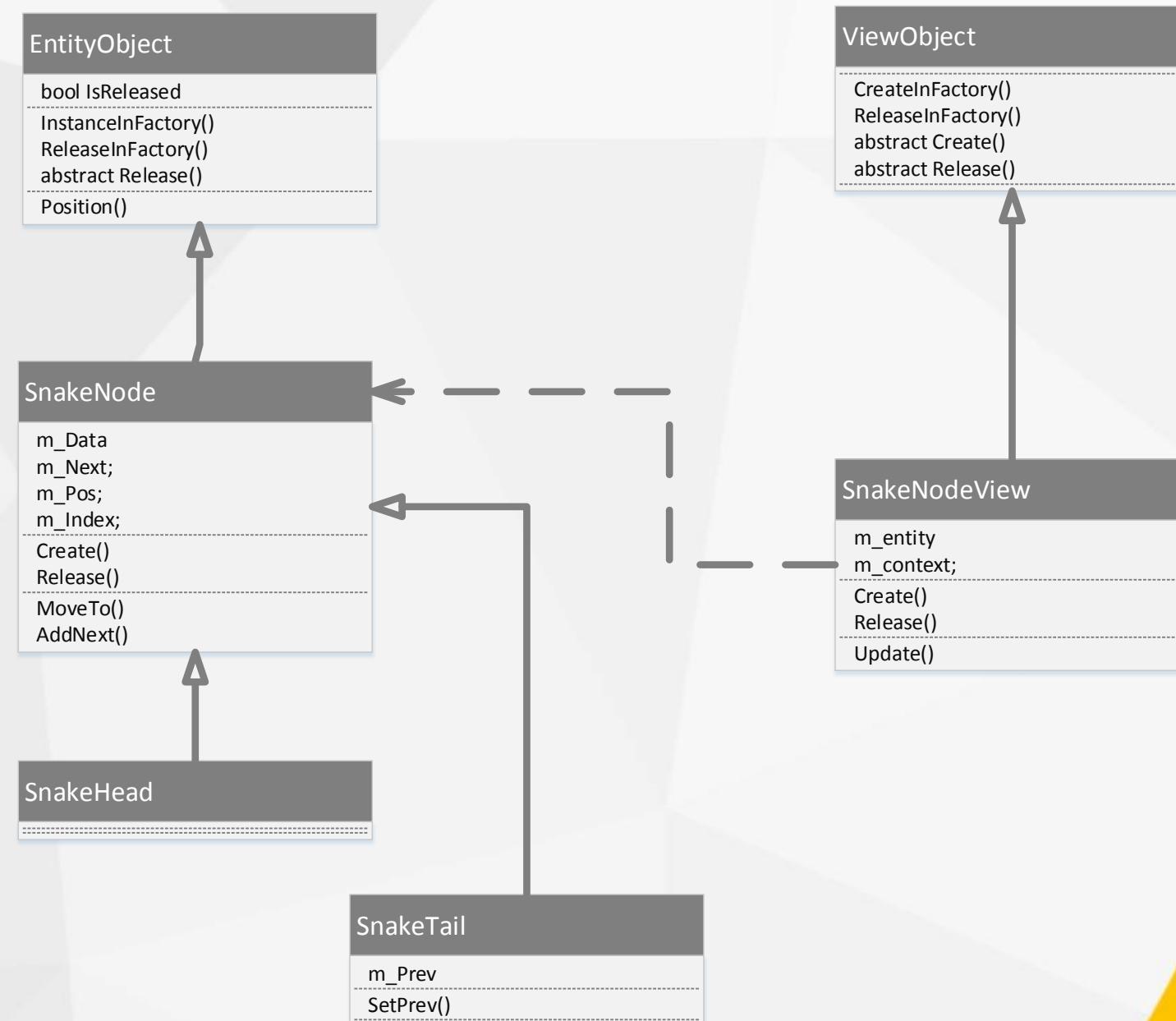
• 8.4.5 蛇的逻辑结构

- SnakePlayer，是【玩家模块】用来管理和维持蛇的逻辑结构
- SnakePlayer指向SnakeTail和SnakeTail指向倒数第2个Node，
- 是为了能够快速在SnakeTail之前插入Node



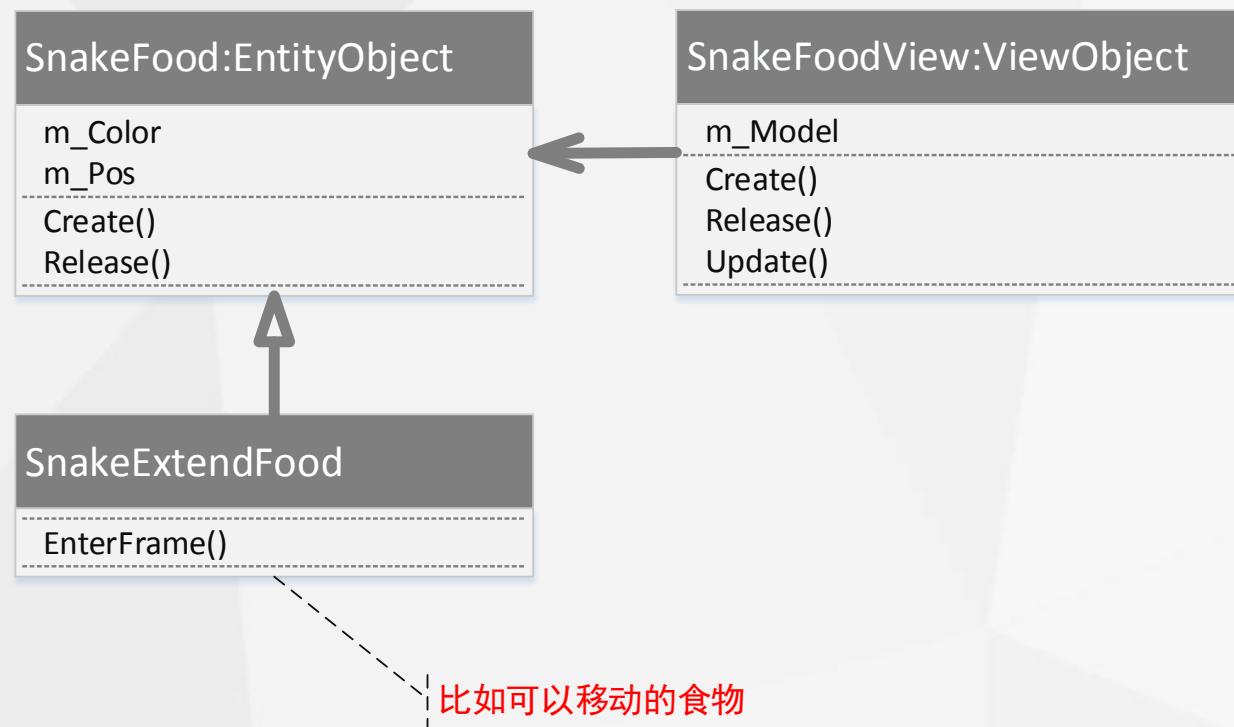
8、核心玩法

- 8.4 核心对象
 - 8.4.6 蛇的UML



8、核心玩法

- 8.4 核心对象
 - 8.4.7 食物的UML

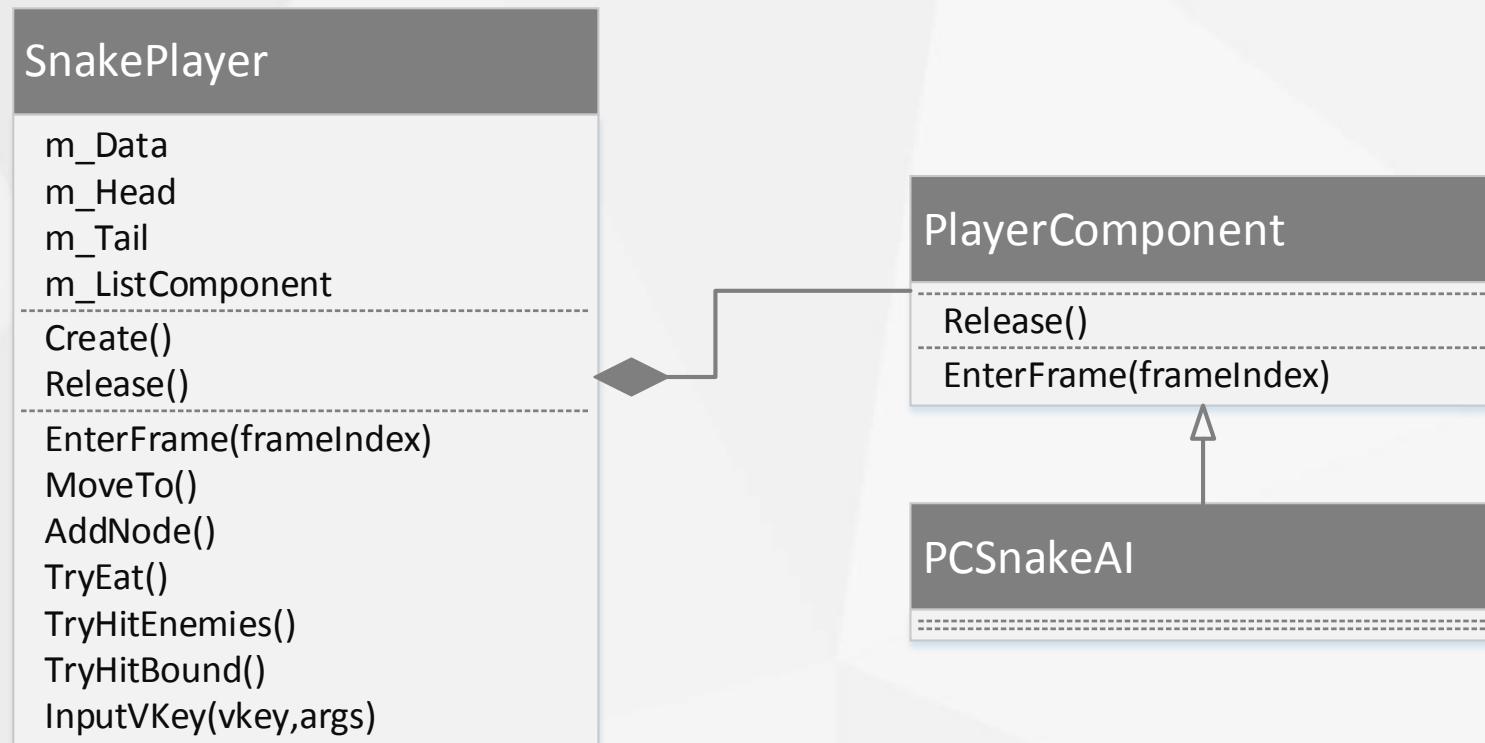


8、核心玩法

- 8.5 玩家模块

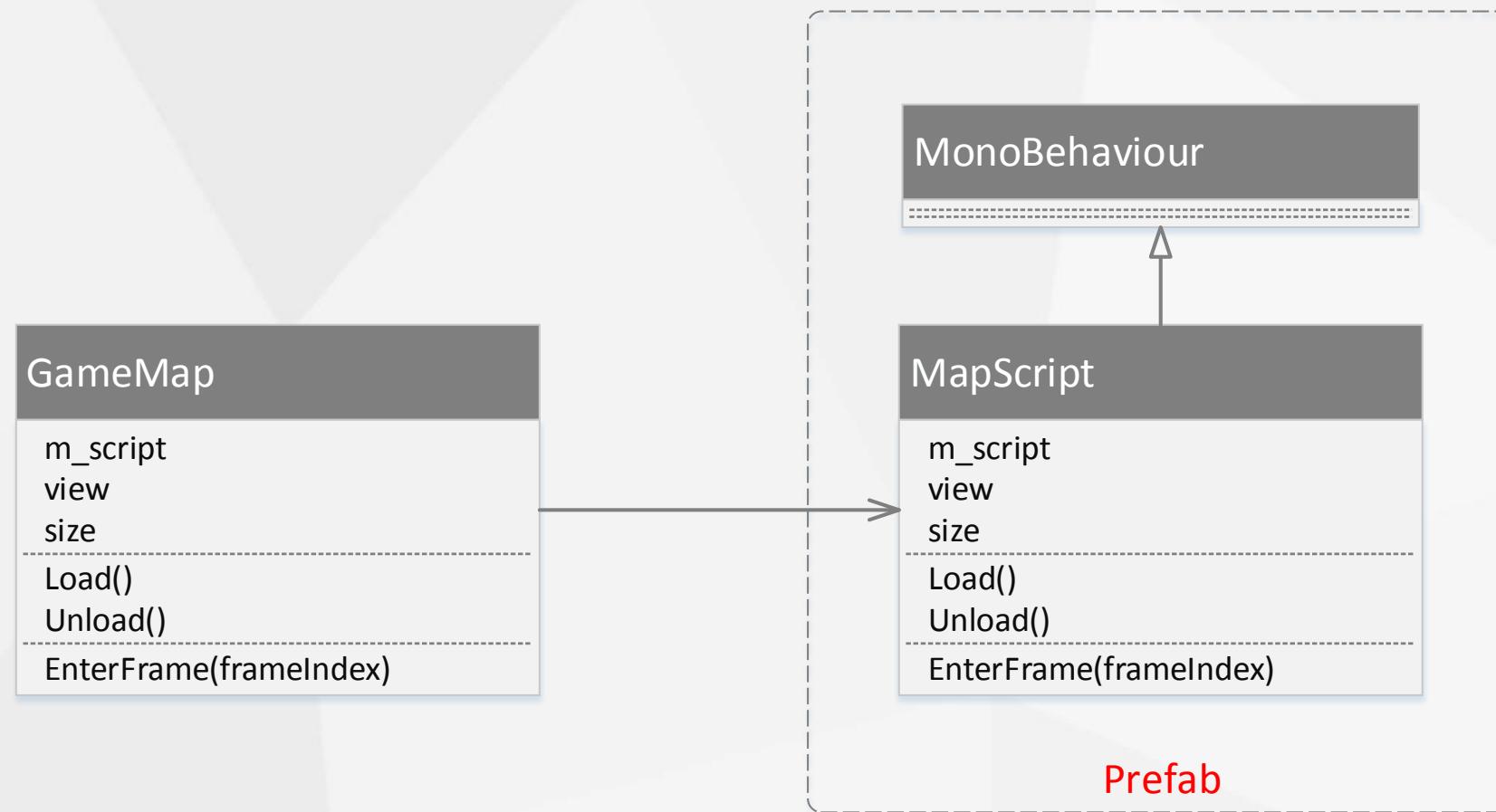
- 8.5.1 UML

- 管理蛇的逻辑结构
 - HitTest
 - HandleVkey
 - 组件管理



8、核心玩法

- 8.6 地图模块
 - 8.6.1 UML

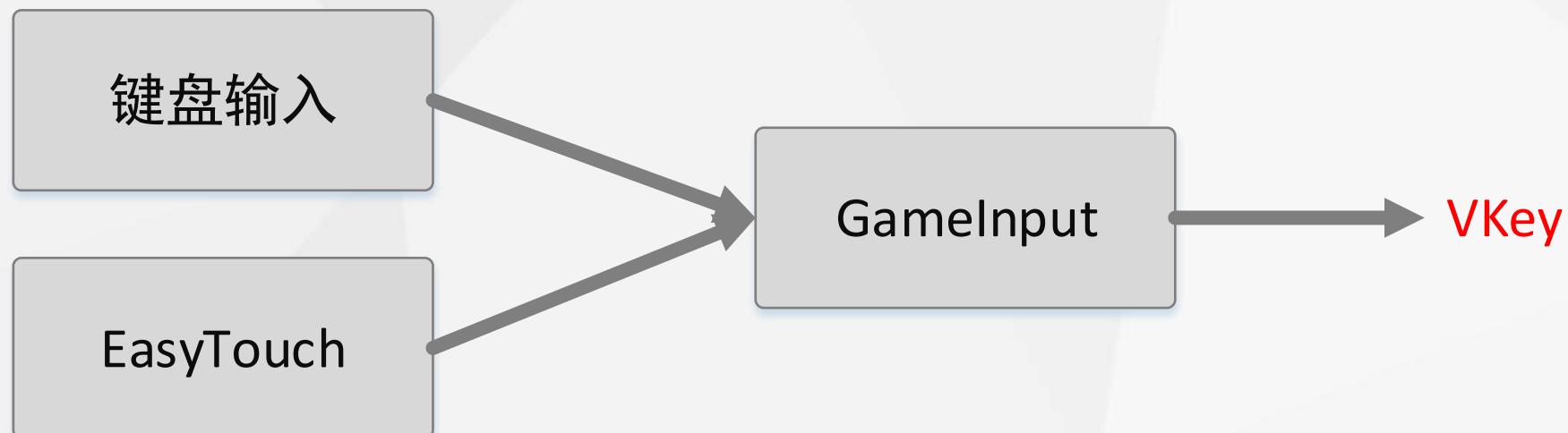


8、核心玩法

- 8.7 输入模块
 - 8.7.1 EasyTouch接入

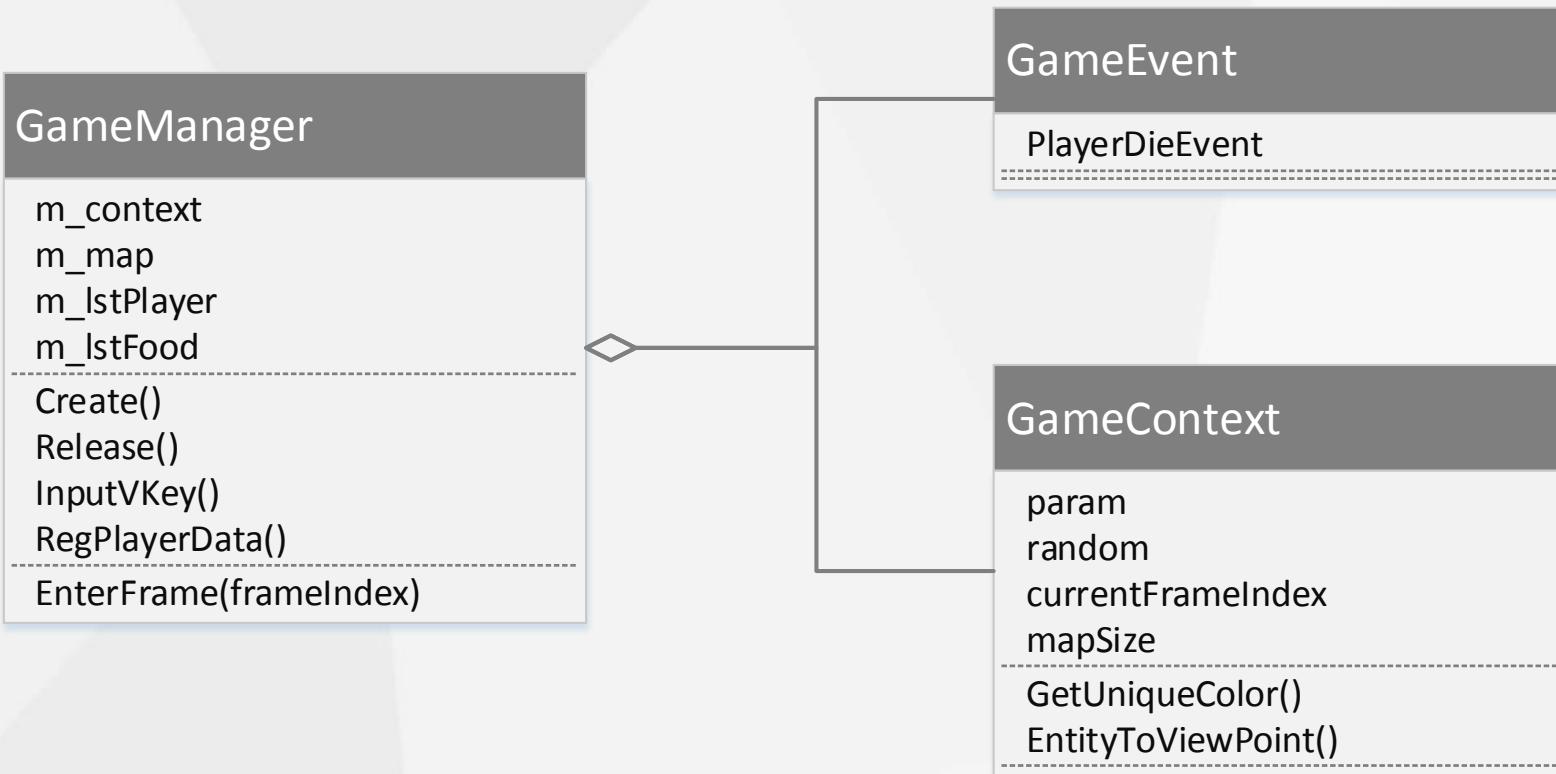
8、核心玩法

- 8.7 输入模块
 - 8.7.2 EasyTouch与键盘输入的封装



8、核心玩法

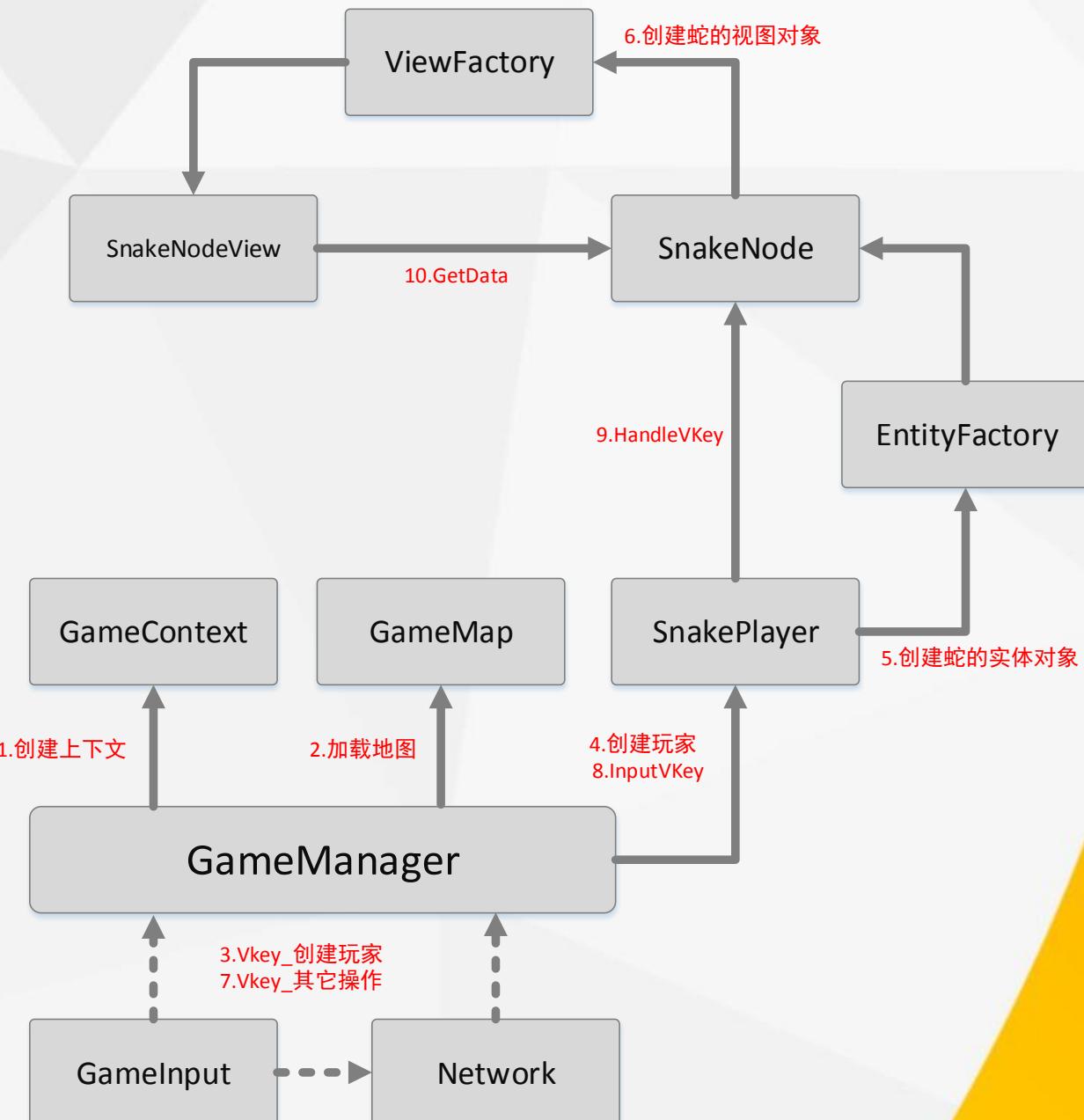
- 8.8 游戏管理器
 - 8.8.1 UML



8、核心玩法

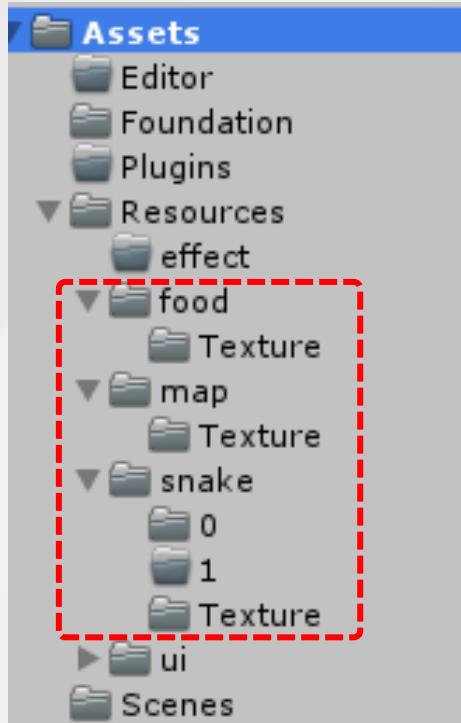
- 8.8 游戏管理器

- 8.8.2 主体逻辑示意图

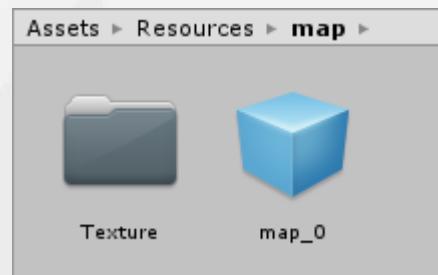


8、核心玩法

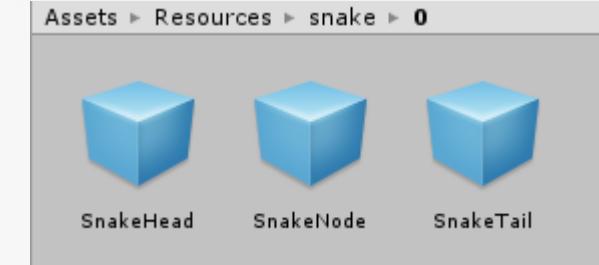
- 8.9 资源导入
 - 8.9.1 资源目录规划



食物按类型ID定义
Prefab



地图按地图ID定义
Prefab



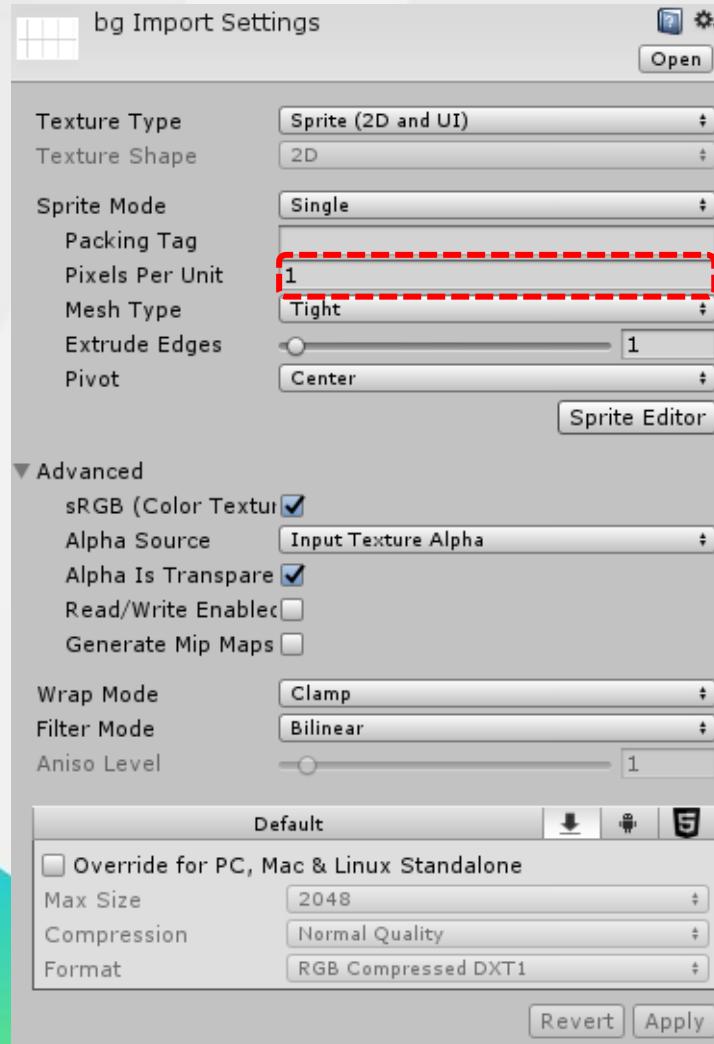
蛇按类型ID定义
Prefab

相同【类型】的食物和蛇，其【颜色】可能不同

8、核心玩法

• 8.9 资源导入

• 8.9.2 Pixels Per Unit设置



作为2D游戏，建议设为1，这样游戏中所有单位都是像素，好理解

8、核心玩法

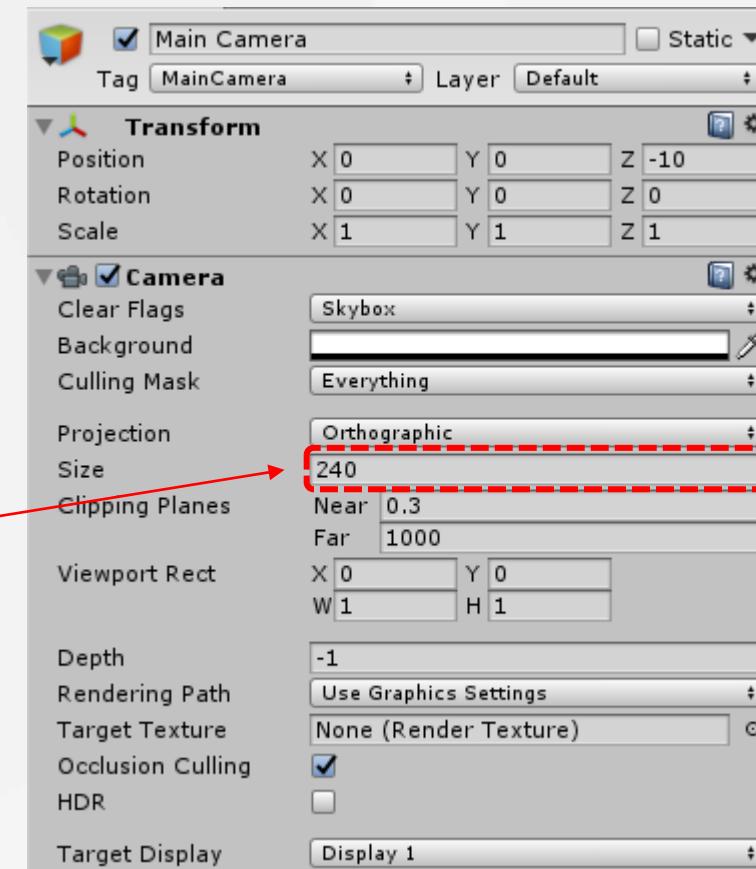
- 8.9 资源导入
 - 8.9.3 基准尺寸定义及CameraSize

可以将蛇的结点基准尺寸定为半径16像素，则



视口高=30*16=480

那么, Camera.Size = 240



9、帧同步 (Lite版)

- 9.1 关于帧同步
- 9.2 基本原理
- 9.3 网络通讯
- 9.4 整体框架
- 9.5 数据定义
- 9.6 客户端模块
- 9.7 服务器模块
- 9.8 演示Demo

9、帧同步 (Lite版)

• 9.1 关于帧同步

• 9.1.1 什么情况下适合采用帧同步？

- 游戏类型
 - FPS/TPS ? RTS/MOBA ? ACG?
 - 单局PVP , PVE模式
- 必要条件
 - 完备信息（无不可控输入），单局时长有限
- 充分条件
 - 高实时、高一致
 - 弱网络
 - 开发周期短

9、帧同步 (Lite版)

• 9.1 关于帧同步

• 9.1.2 帧同步优缺点对比

• 优点

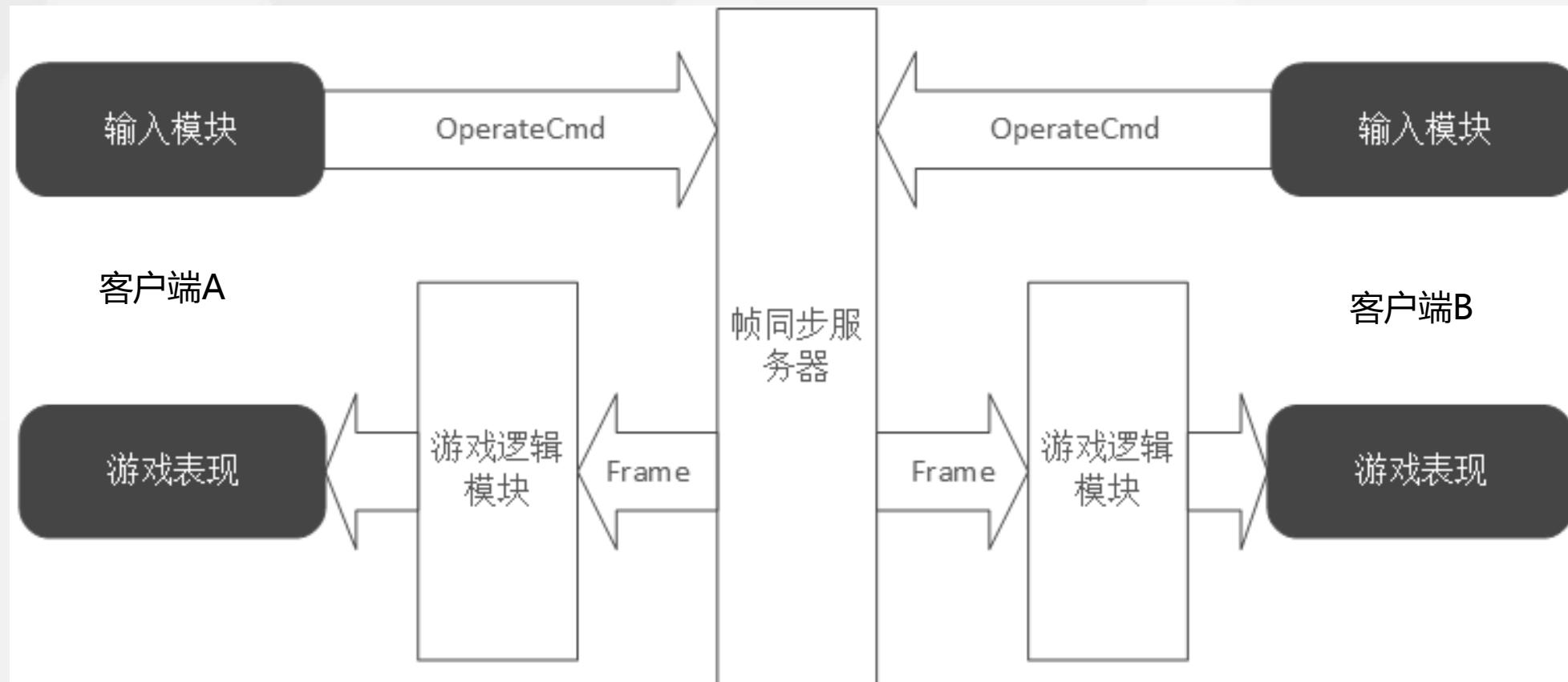
- 开发效率高，接近单机游戏开发的效率
- 100%完全同步，无法直接作弊
- 流量低，服务器负载低
- 通用单局服务器，几乎不需要重写后台代码
- 天然支持观战，录像，回放等功能
- 可支持海量游戏单位（因为再多单位，输入也是10个手指）

• 缺点

- 开发错误，逻辑计算不确定，等将导致不同步
- 无法避免感知外挂（客户端具有完备信息）

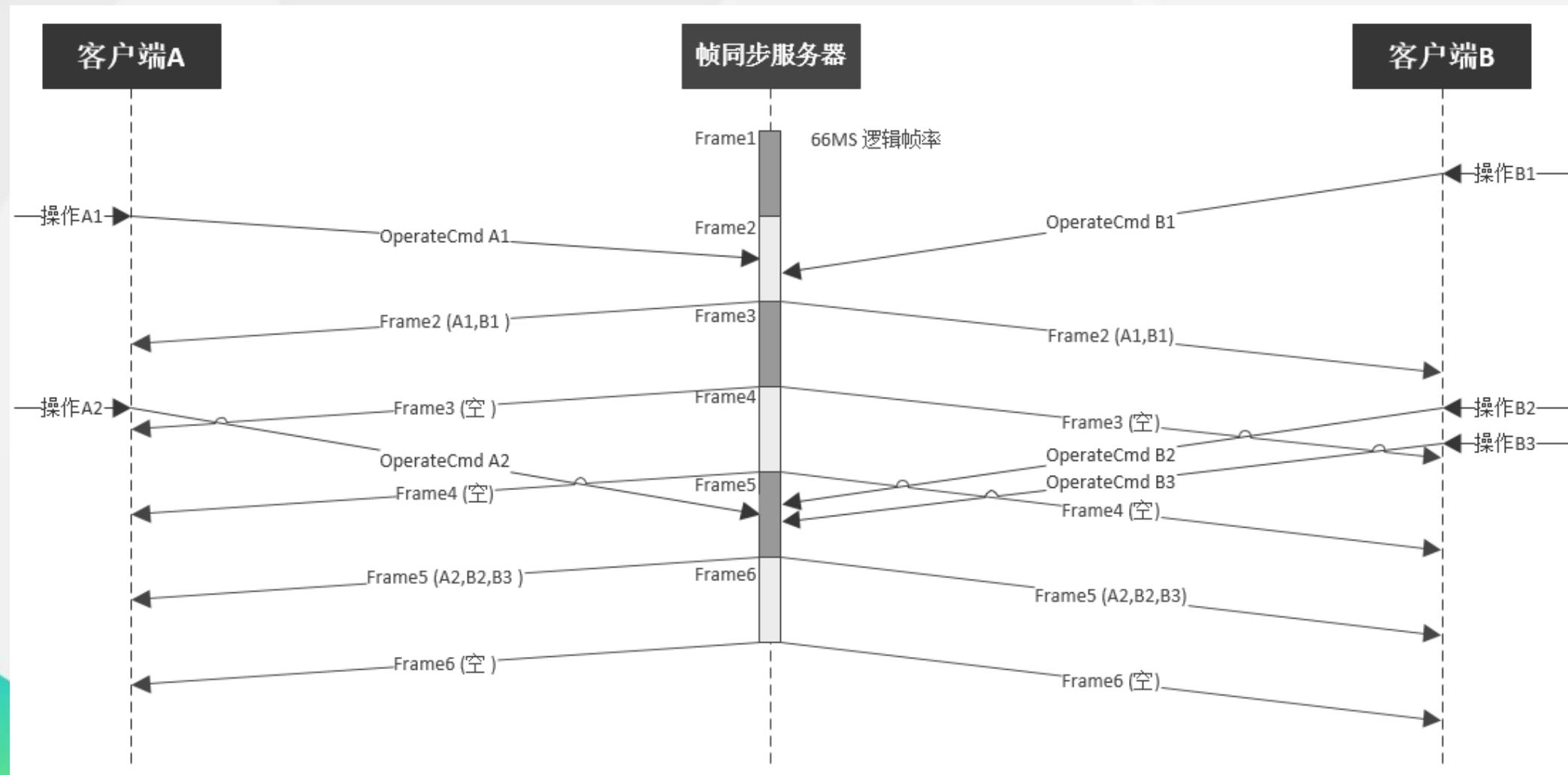
9、帧同步 (Lite版)

- 9.2 基本原理
 - 9.2.1 原理示意图



9、帧同步 (Lite版)

- 9.2 基本原理
- 9.2.2 同步的一致性



9、帧同步 (Lite版)

- 9.2 基本原理
 - 9.2.3 逻辑的确定性
 - 基于帧同步的游戏框架
 - 视图对象与逻辑对象单向解耦
 - 视图对象无法修改逻辑对象的值
 - 定点数数学库
 - 或者采用精度可控的浮点数
 - $(\text{int})(0.123456789 * 100000) / 100000$
 - 具备确定性的物理引擎
 - 或者用到的部分物理算法

9、帧同步 (Lite版)

• 9.3 网络通讯

• 9.3.1 使用TCP还是UDP ?

- 通过实际测试， UDP在弱网络下的延迟表现比TCP好
- 因此我采用UDP作为基础通讯协议

网络环境	TCP平均延迟	RUDP平均延迟	结论
网络良好	133	100	RUDP略优
5%丢包	173	143	RUDP略优
50%丢包	262	115	RUDP优势明显
50ms抖动	198	130	RUDP优势明显

9、帧同步 (Lite版)

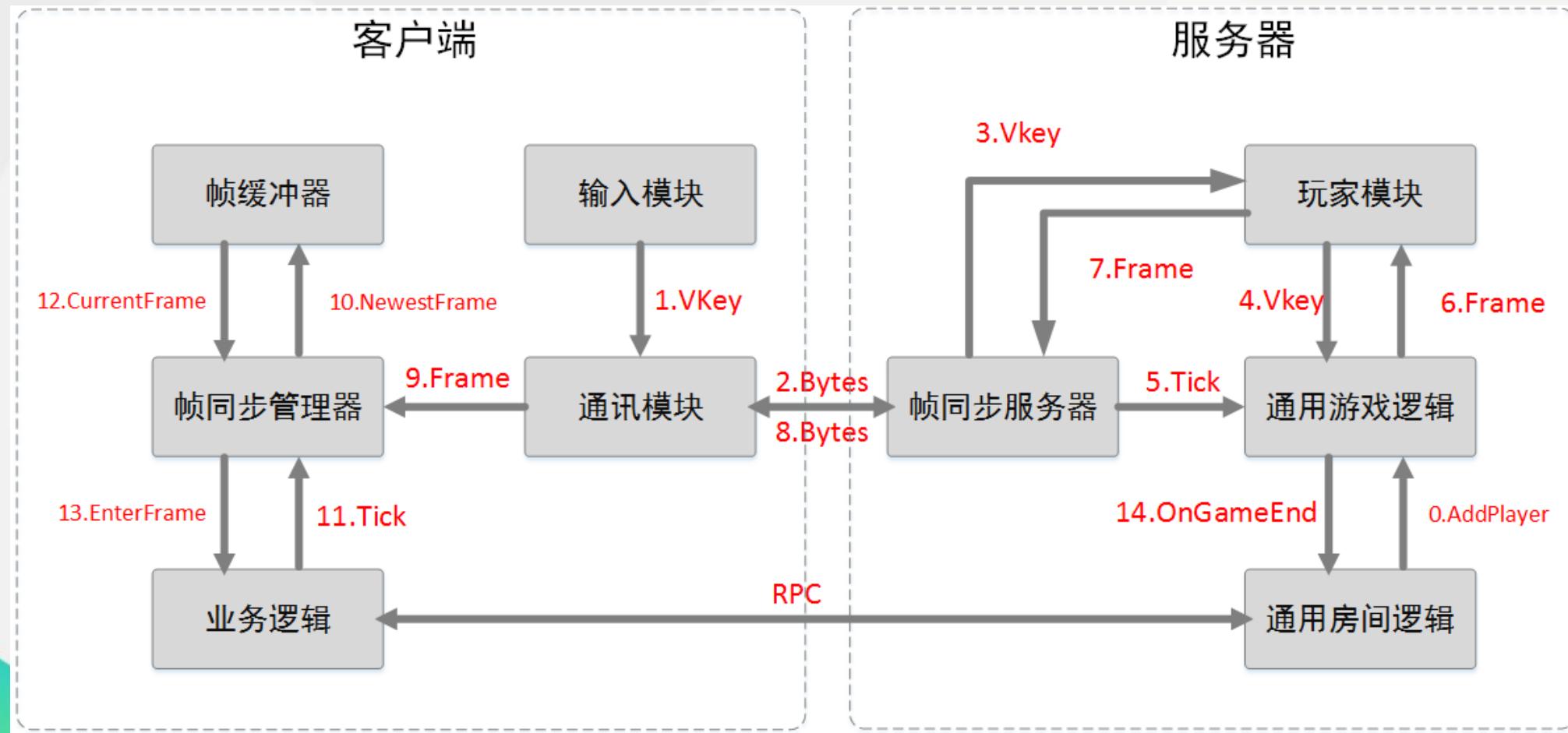
- 9.3 网络通讯

- 9.3.2 UDP如何保证可靠性？

- 由于RUDP的实现，不是本课程的重点，我采用了一个开源方案：KCP
 - 对其针对Unity客户端开发进行封装：KCPSocket
 - 在使用上，可以像使用UDP一样方便，但是又像TCP一样可靠。
 - 开源代码：<https://github.com/slicol/KCP-Socket>

9、帧同步 (Lite版)

- 9.4 整体框架
 - 9.4.1 模块划分



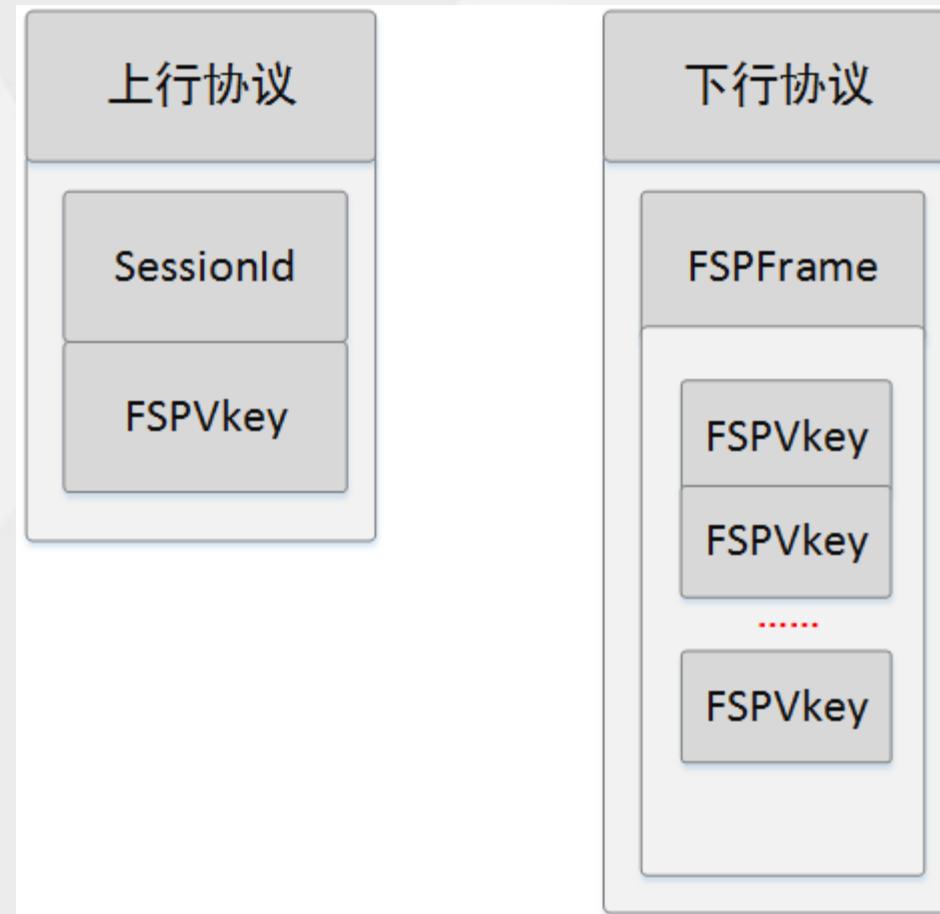
9、帧同步 (Lite版)

- 9.5 数据定义
 - 9.5.1 启动参数

参数	定义
Host	服务器IP
Port	服务器端口
Sid	会话ID，如果是局域网服务器，则等于PlayerId
serverFrameInternal	服务器 1 帧的时间
serverTimeout	服务器判断客户端掉线的超时
clientFrameRateMultiple	客户端与服务器帧率倍数

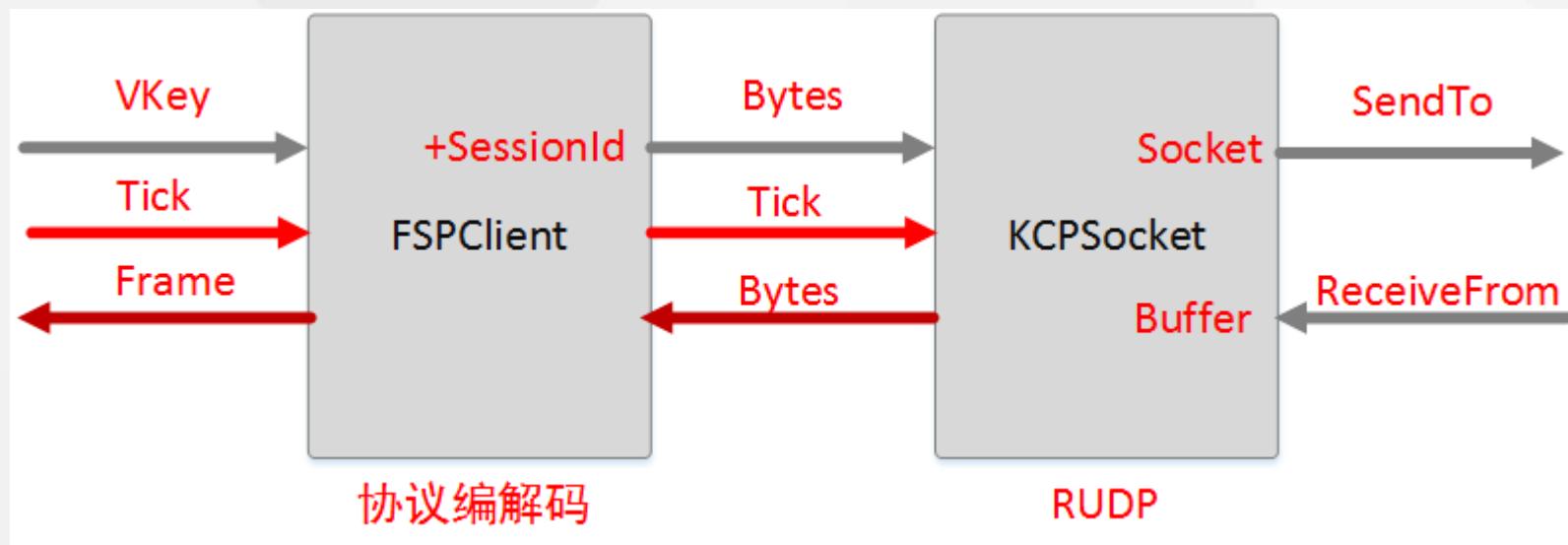
9、帧同步 (Lite版)

- 9.5 数据定义
 - 9.5.2 上下行协议 (基于ProtoBuf定义)



9、帧同步 (Lite版)

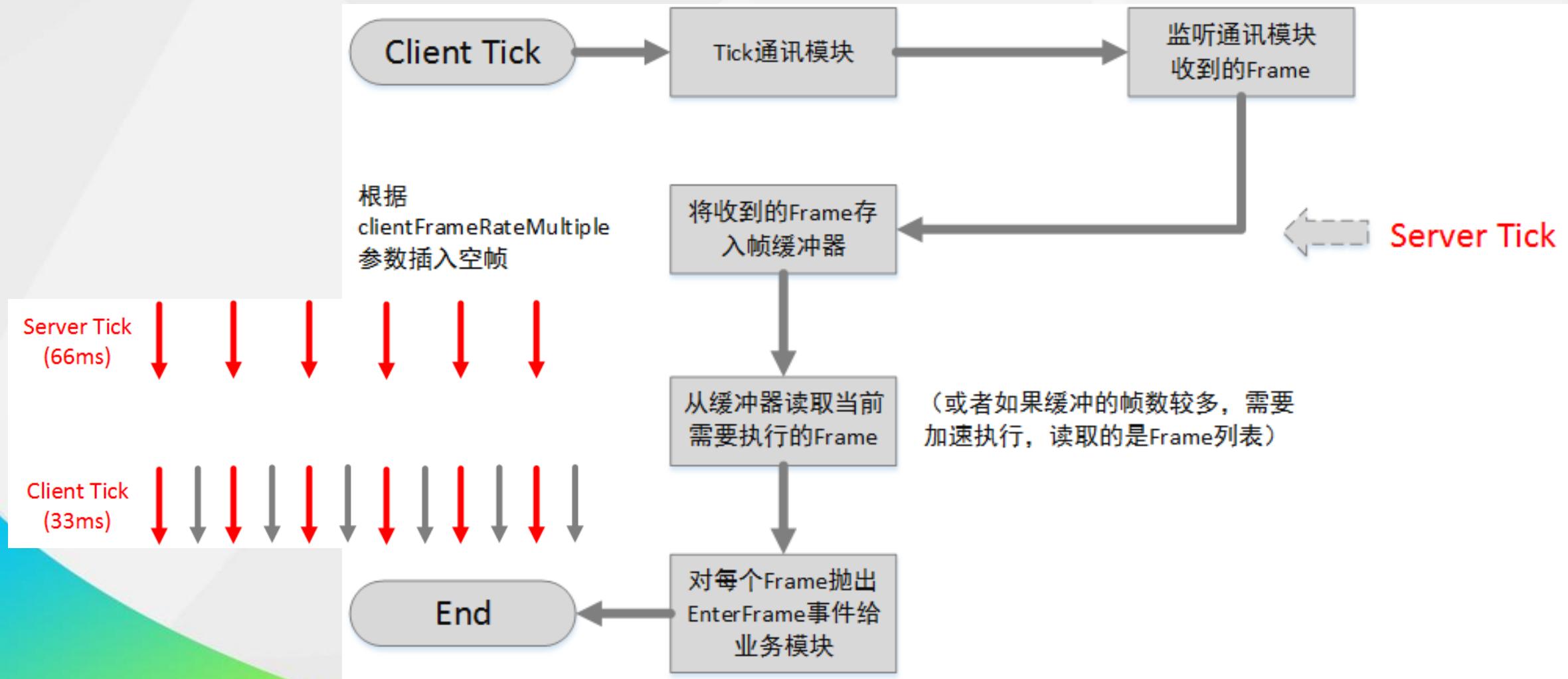
- 9.6 客户端设计
 - 9.6.1 客户端通讯模块



9、帧同步 (Lite版)

• 9.6 客户端设计

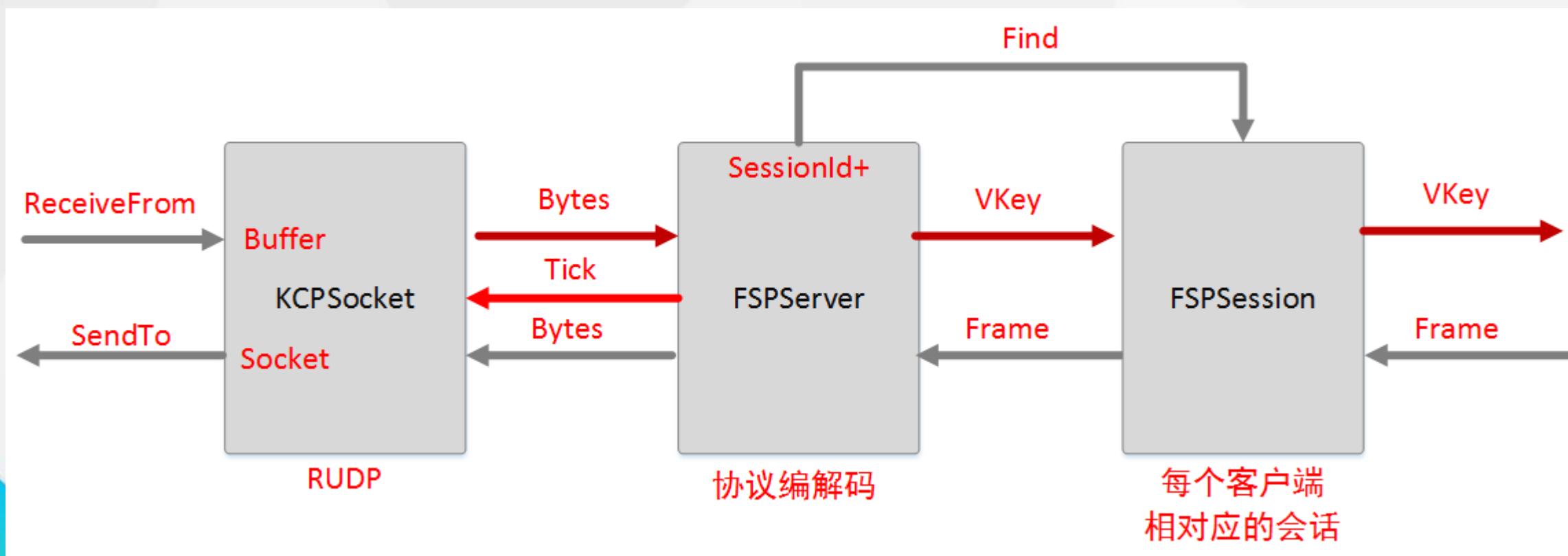
• 9.6.2 帧同步管理器主流程



9、帧同步 (Lite版)

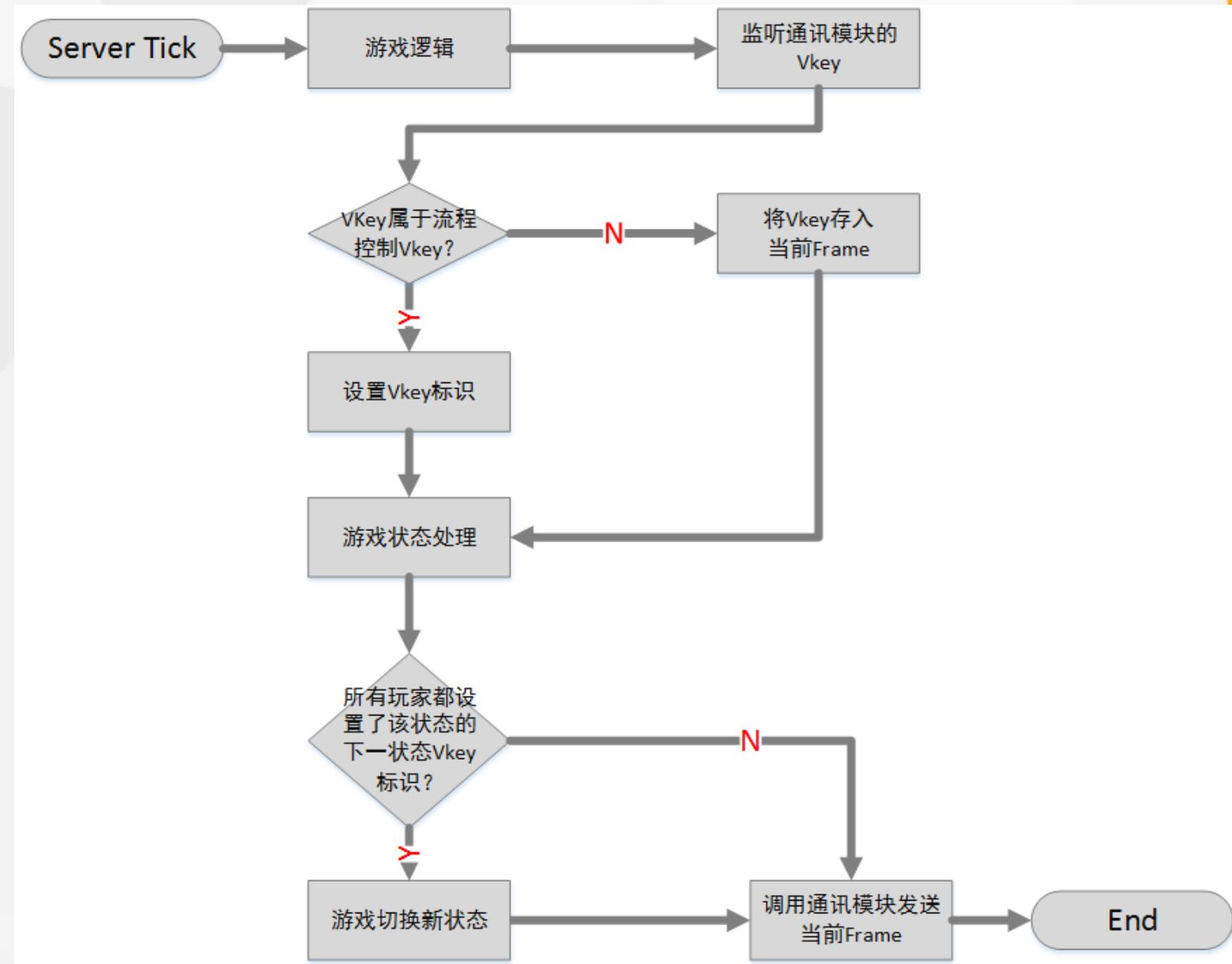
• 9.7 服务器设计

• 9.7.1 服务器通讯模块



9、帧同步 (Lite版)

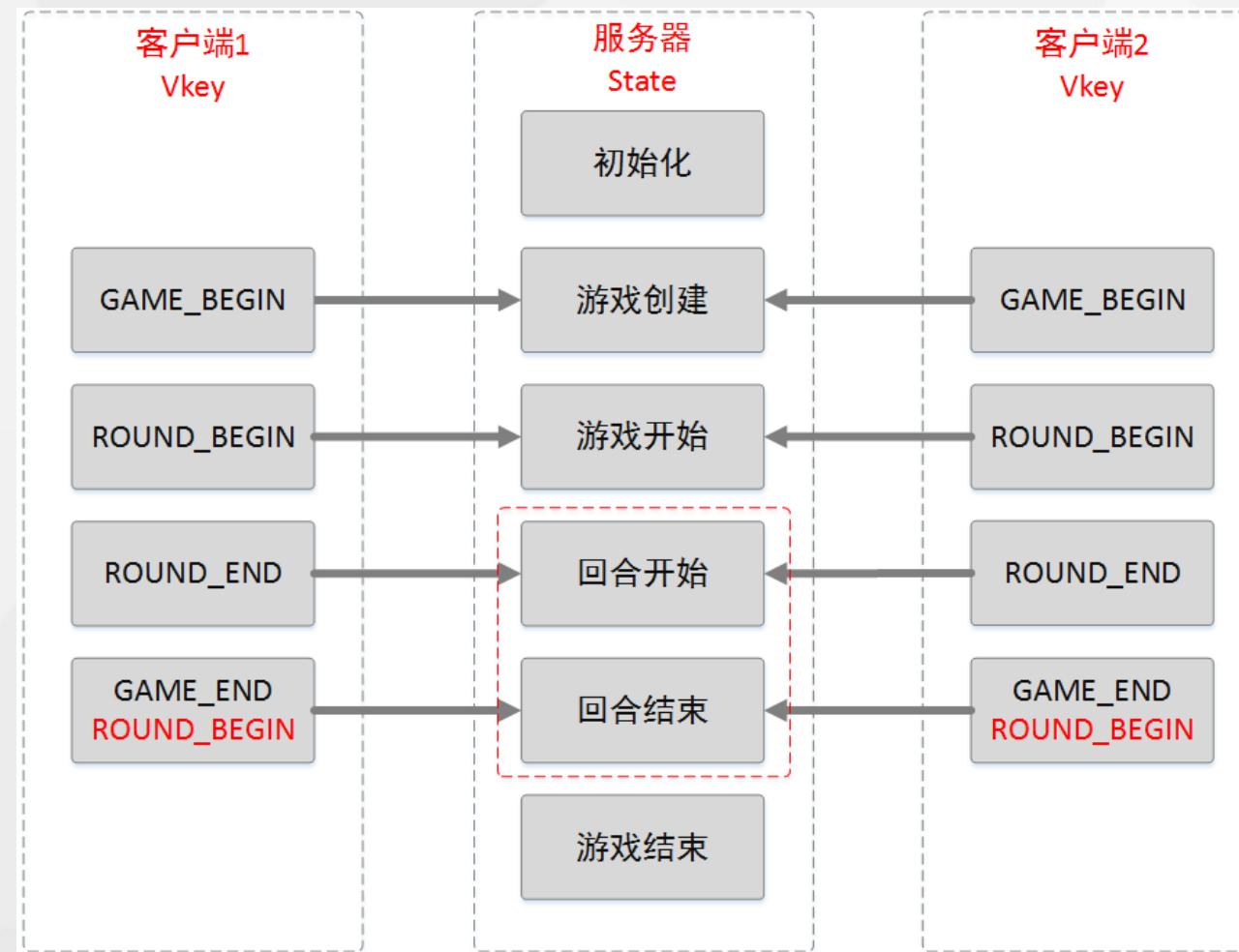
- 9.7 服务器设计
 - 9.7.2 通用游戏逻辑主流程



9、帧同步 (Lite版)

• 9.7 服务器设计

• 9.7.3 服务器状态机



9、帧同步 (Lite版)

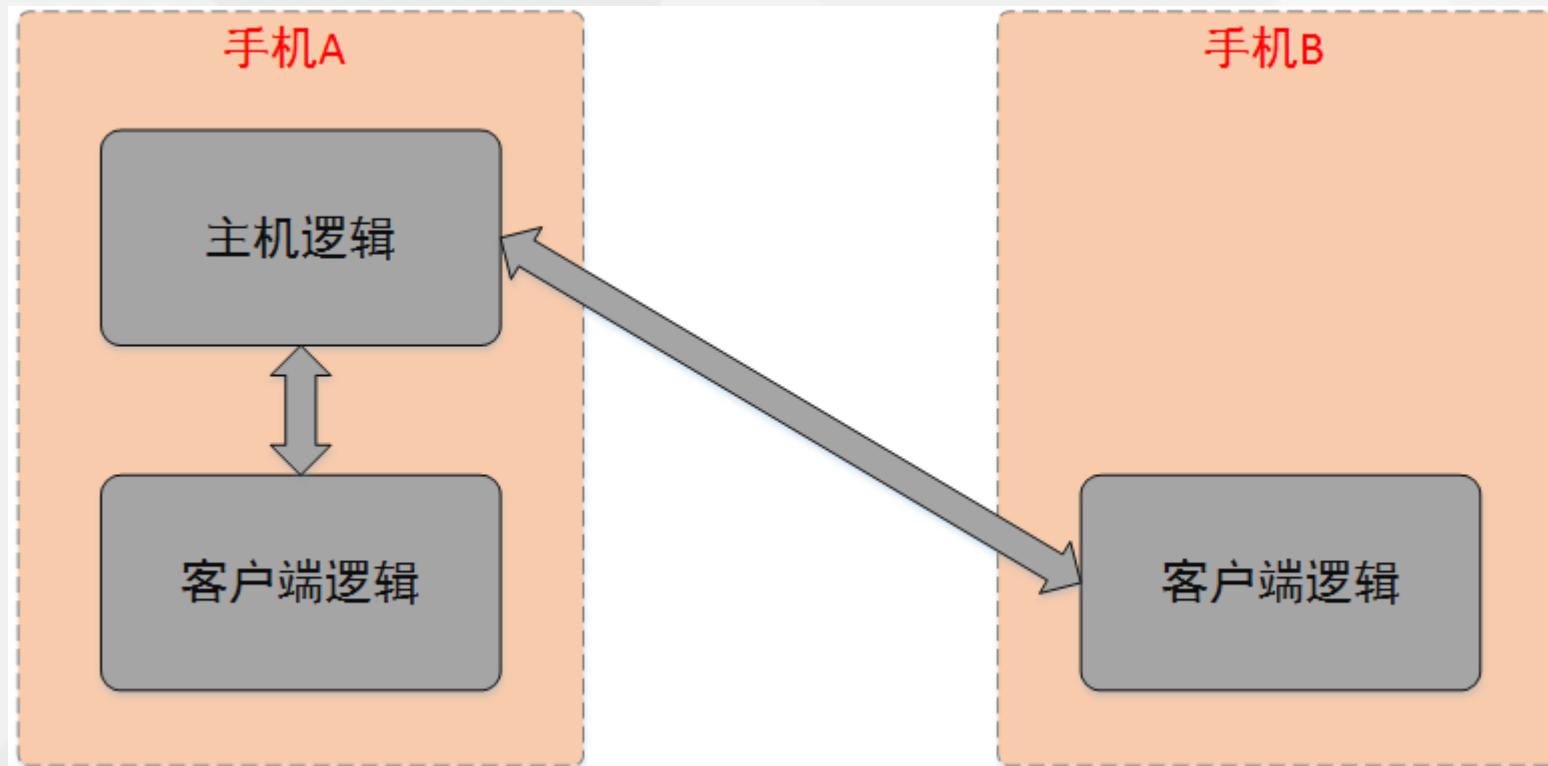
- 9.8 演示Example

10、局域网对战

- 10.1 局域网对战原理
- 10.2 局域网配对方案
- 10.3 RPC框架（Lite版）

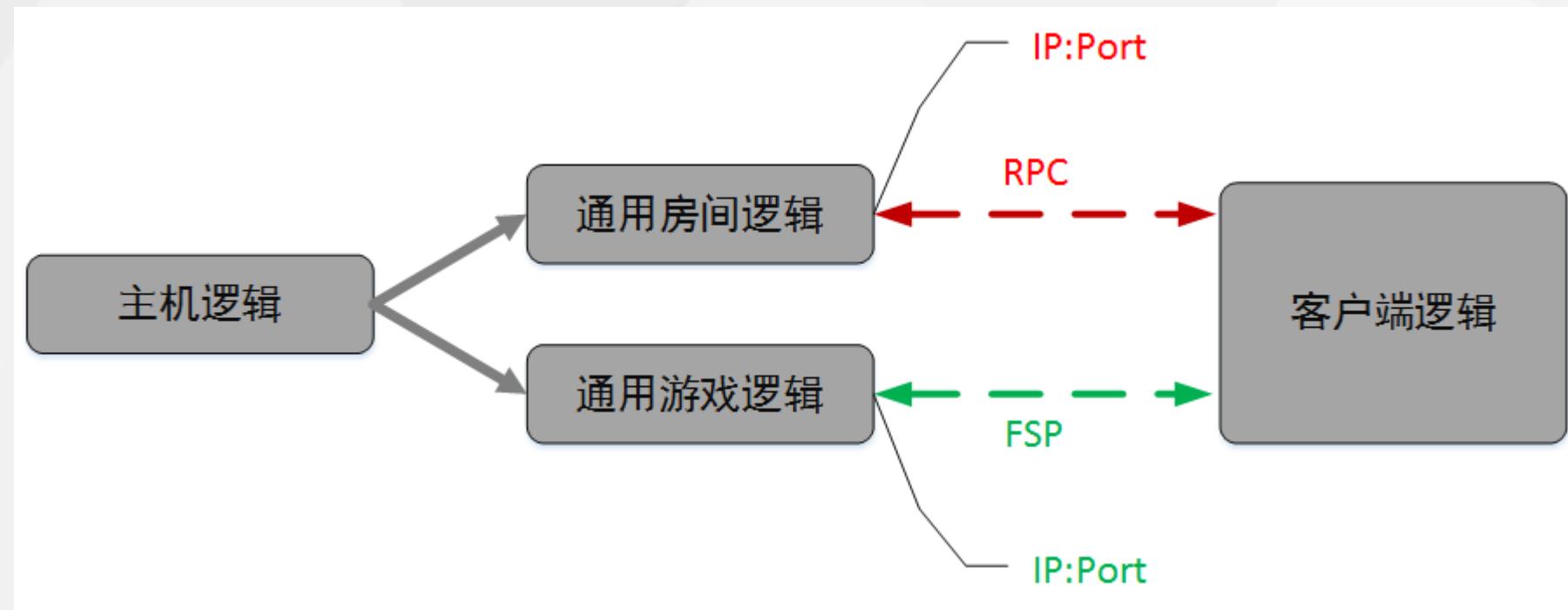
10、局域网对战

- 10.1 局域网对战原理
 - 10.1.1 主机逻辑与客户端逻辑



10、局域网对战

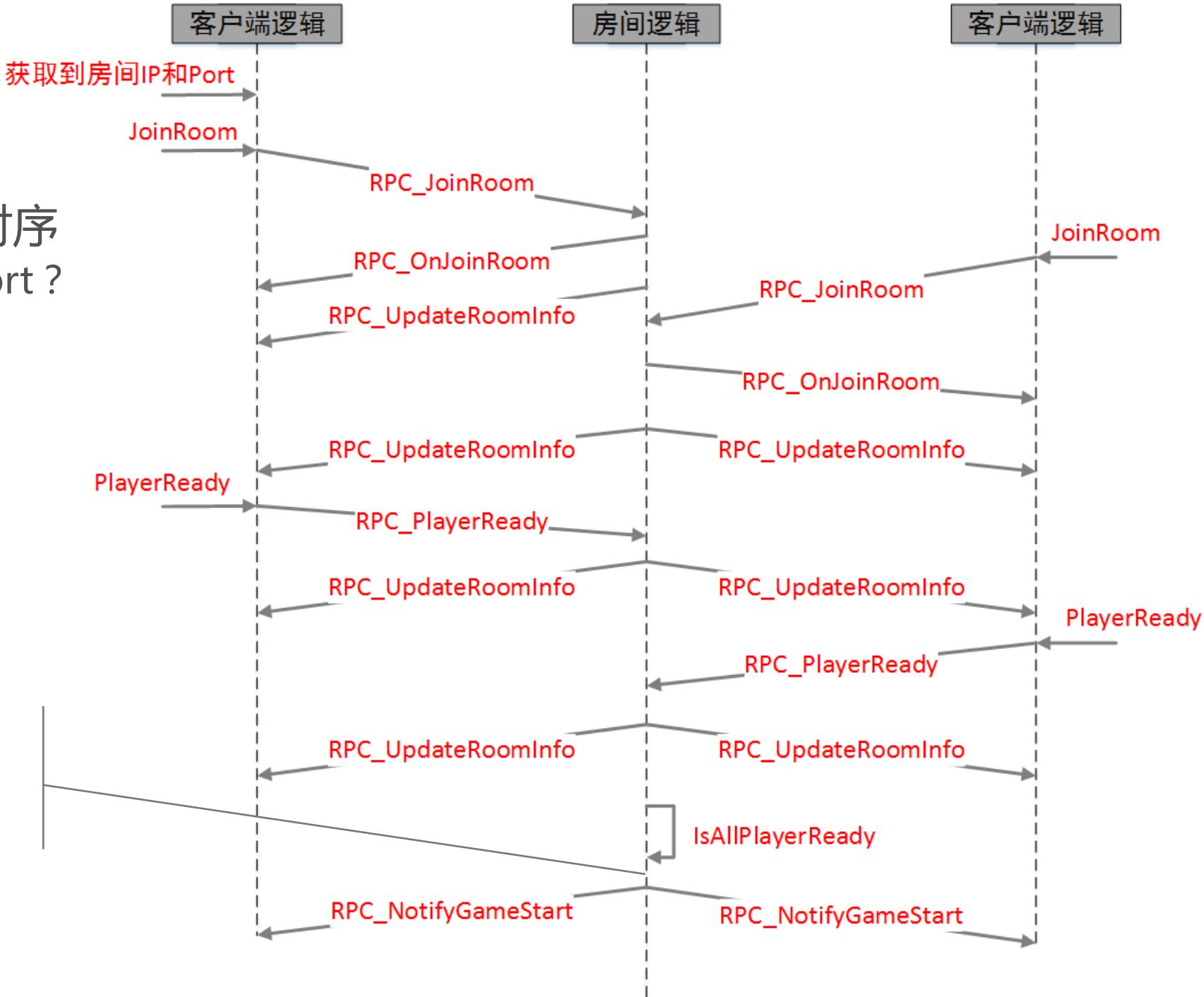
- 10.1 局域网对战原理
 - 10.1.2 通用游戏逻辑与通用房间逻辑
 - 客户端通过IP和Port与主机逻辑通讯



10、局域网对战

- 10.1 局域网对战原理
 - 10.1.3 通用房间逻辑时序
 - 怎么获取房间的IP和Port ?

启动【通用游戏逻辑】
将玩家添加到游戏逻辑中去

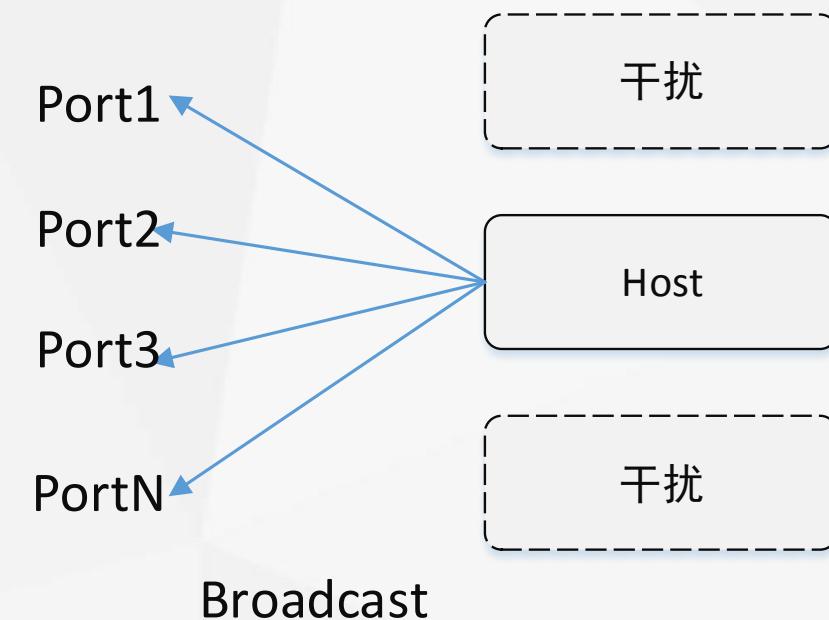
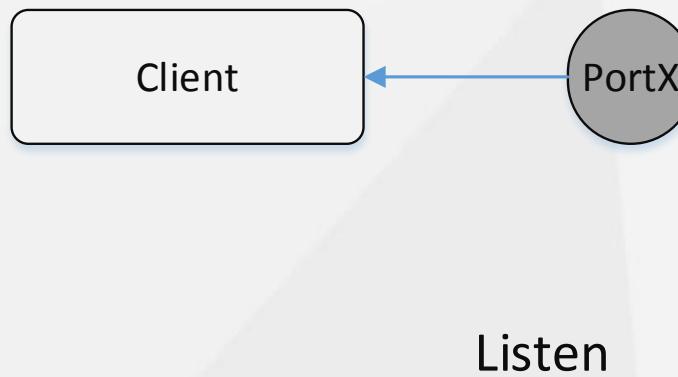


10、局域网对战

- 10.2 局域网配对方案

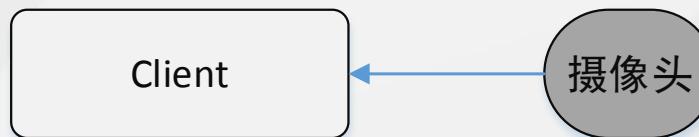
- 10.2.1 局域网广播

- 异类干扰：不同APP之间的干扰
 - 通过Token来区分
 - 同类干扰：相同APP之间的干扰
 - 显示所有主机的列表，手动选择



10、局域网对战

- 10.2 局域网配对方案
 - 10.2.2 二维码识别

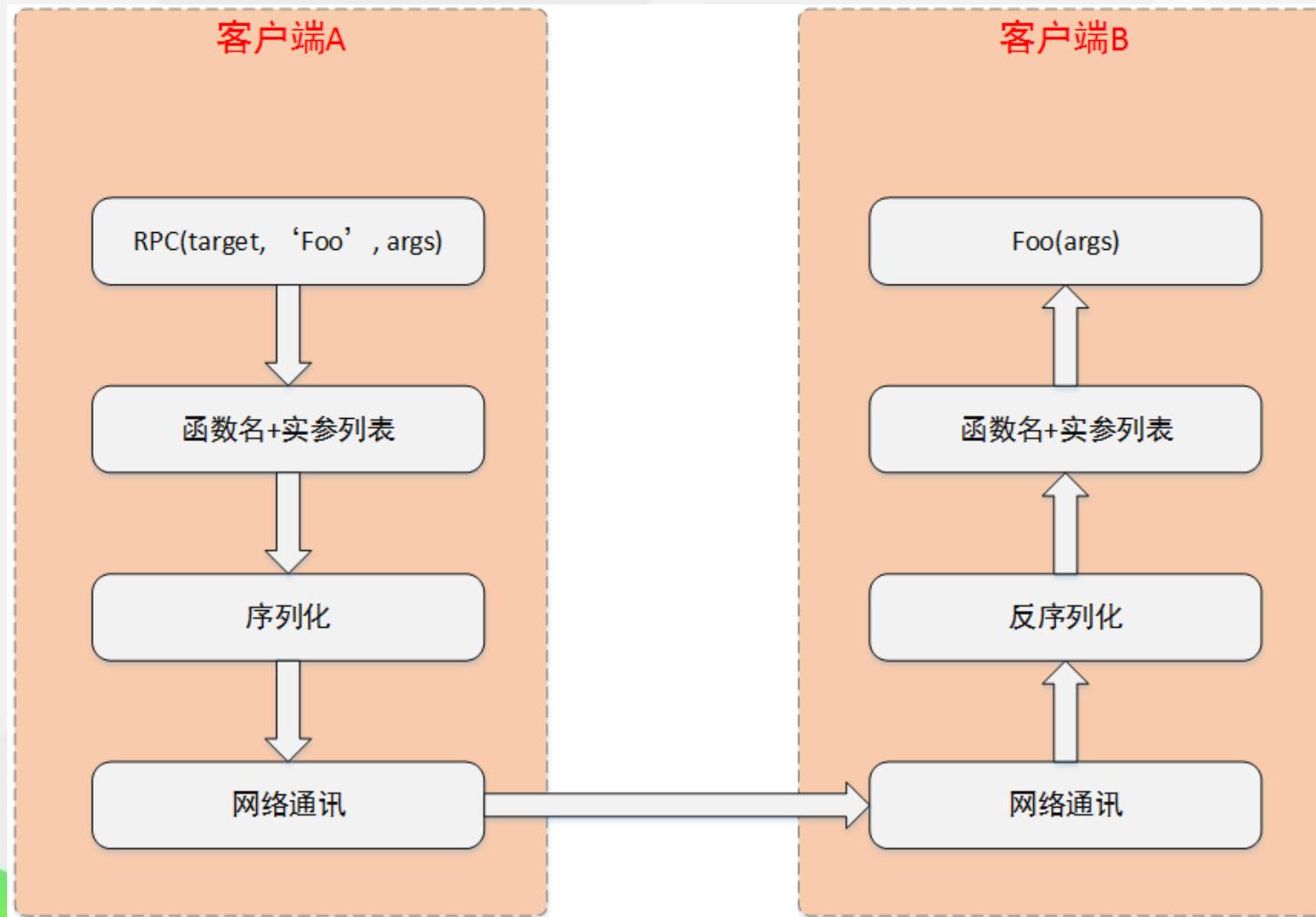


扫描



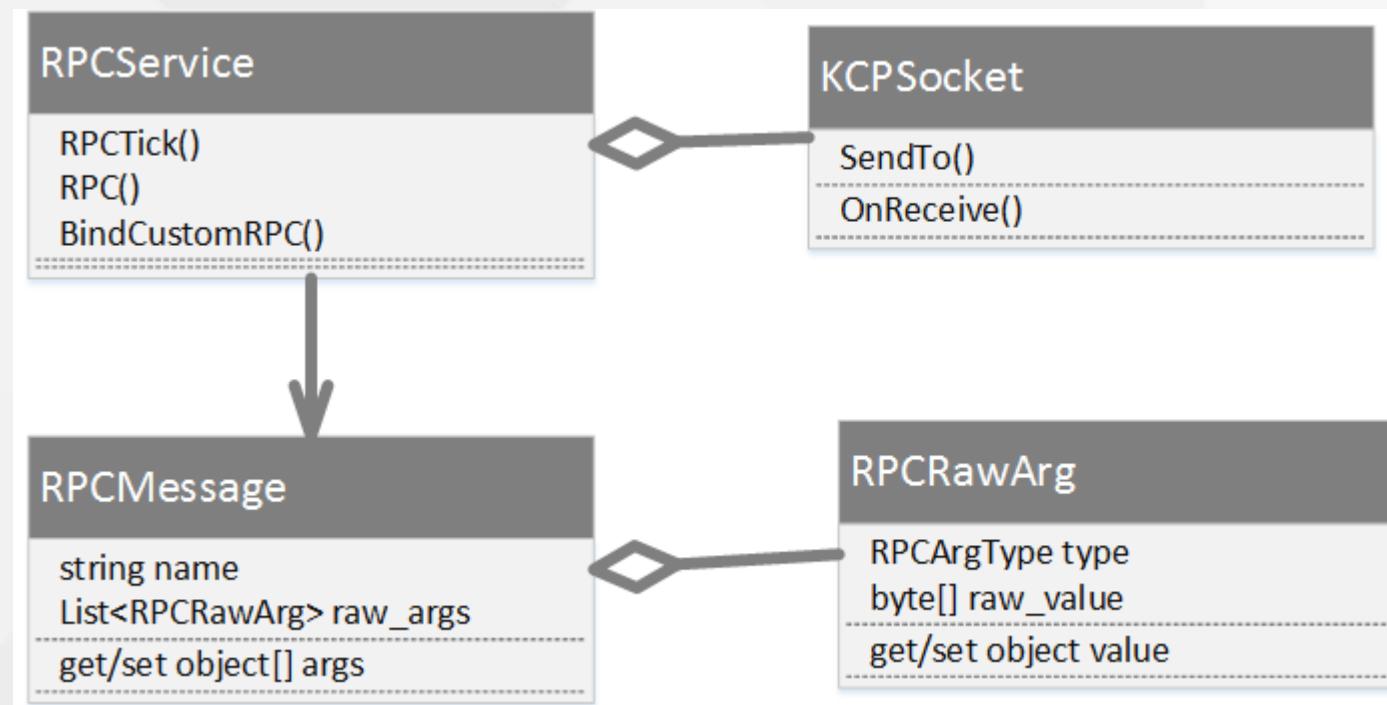
10、局域网对战

- 10.3 RPC框架 (Lite版)
 - 10.3.1 基本原理



10、局域网对战

- 10.3 RPC框架 (Lite版)
 - 10.3.2 UML

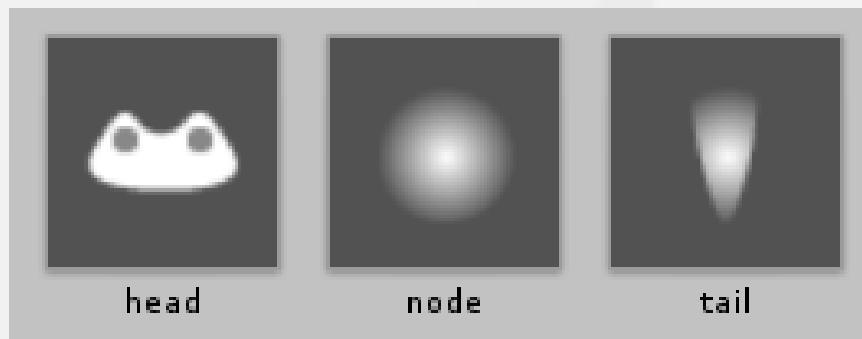
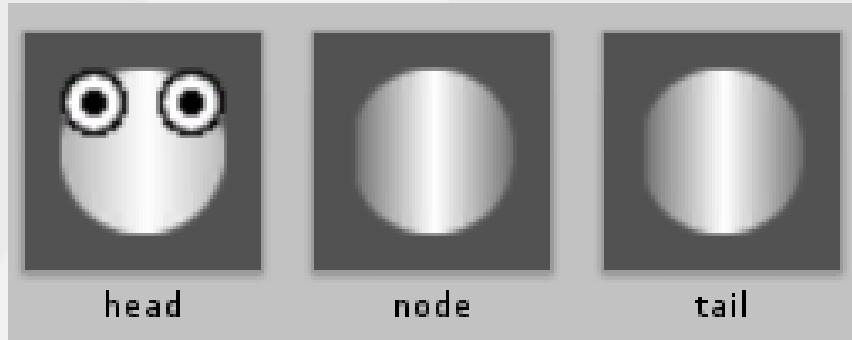


11、皮肤及特效

- 11.1 皮肤制作方式
- 11.2 纹理制作规范
- 11.3 特效实现示例

11、皮肤及特效

- 11.1 皮肤制作方式
 - 11.1.1 直接改变纹理



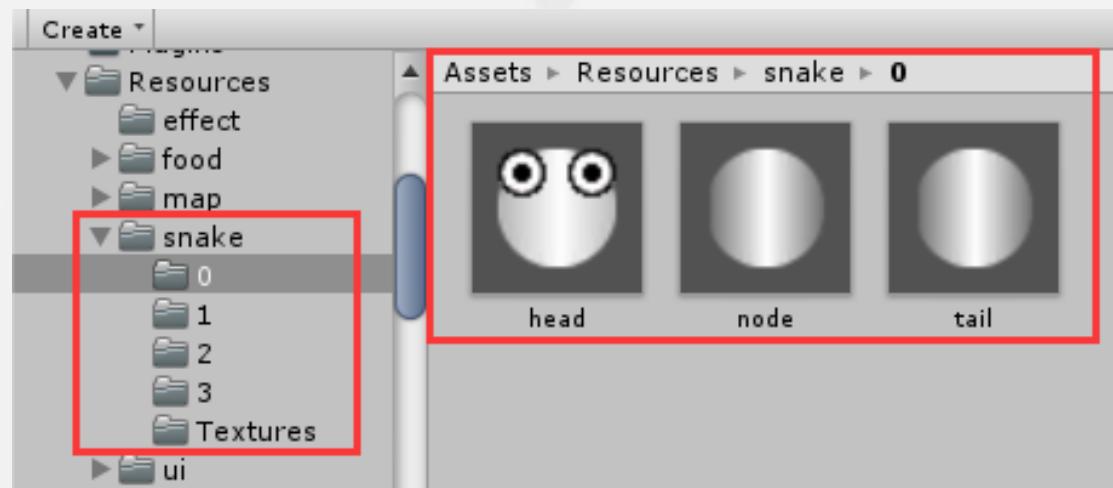
11、皮肤及特效

- 11.1 皮肤制作方式
 - 11.1.2 使用特效
 - 相同的纹理，附加不同的特效
 - 比如闪光皮肤
 - 或者直接使用特效作为皮肤
 - 比如水墨皮肤

11、皮肤及特效

• 11.2 纹理制作规范

- 基准尺寸：直径32像素
- 部位命名：head、node、tail
- 默认朝向：上
- UnitPerPixel：1
- 颜色：黑白灰
- 目录结构：



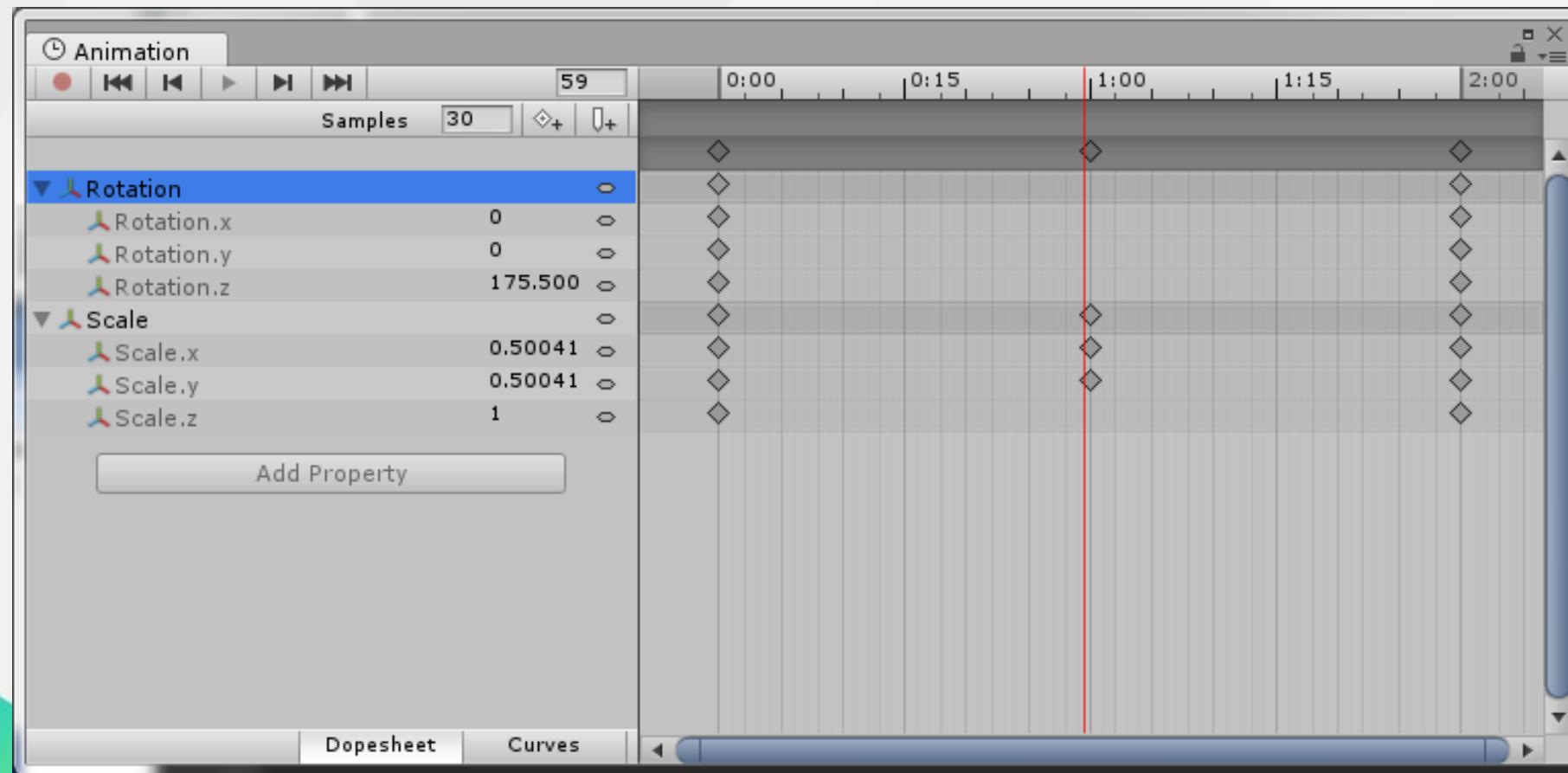
11、皮肤及特效

- 11.3 特效实现示例

- AnimationClip
- HardCode
- ParticleSystem

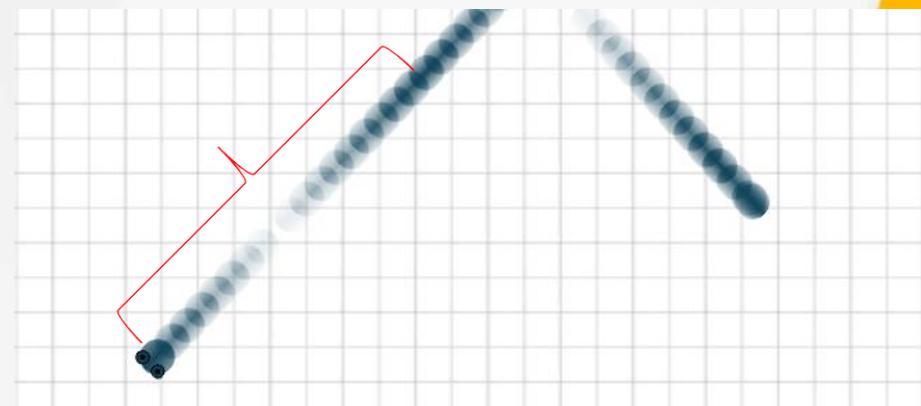
11、皮肤及特效

- 11.3 特效实现示例
 - 11.3.1 AnimationClip
 - 制作示例



11、皮肤及特效

- 11.3 特效实现方式
 - 11.3.2 HardCode
 - 闪光效果
 - 修改View的Alpha值
 - $[1,0,1] \leq |[1, 0, -1]| \leq |10 - [0, 20]| / 10$
 - $[0, 20] \leq (\text{View.index}) \% 20$
 - $\text{View.index} \leq \text{Entity.index}/\text{KeyStep}$



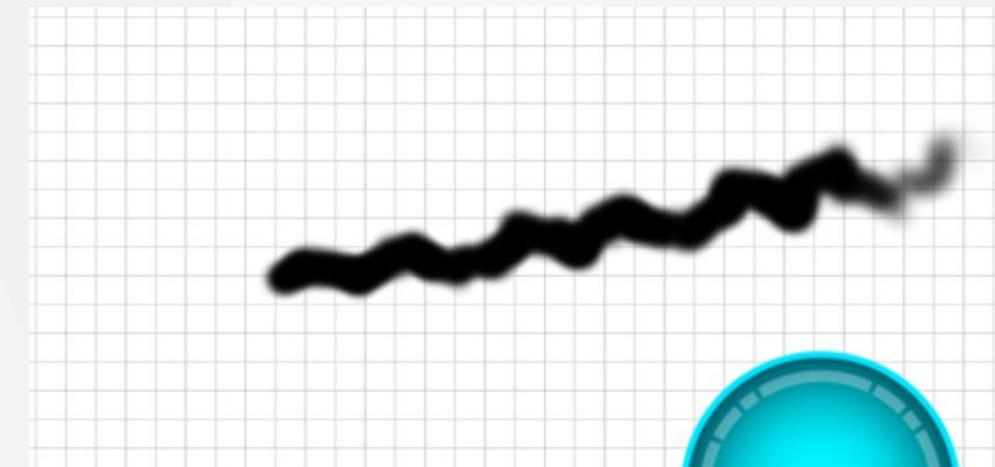
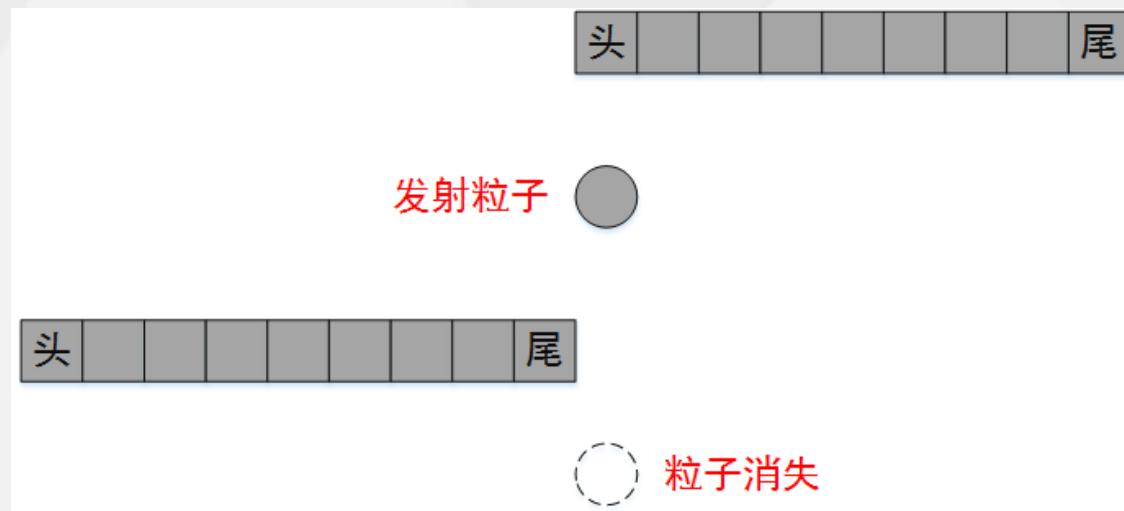
11、皮肤及特效

- 11.3 特效实现方式

- 11.3.3 ParticleSystem

- 水墨特效

- 在Head结点附加一个粒子系统



- 粒子的LifeTime = NodeNum*0.033f;

12、游戏打包

- 12.1 Android平台
- 12.2 IOS平台

12、游戏打包

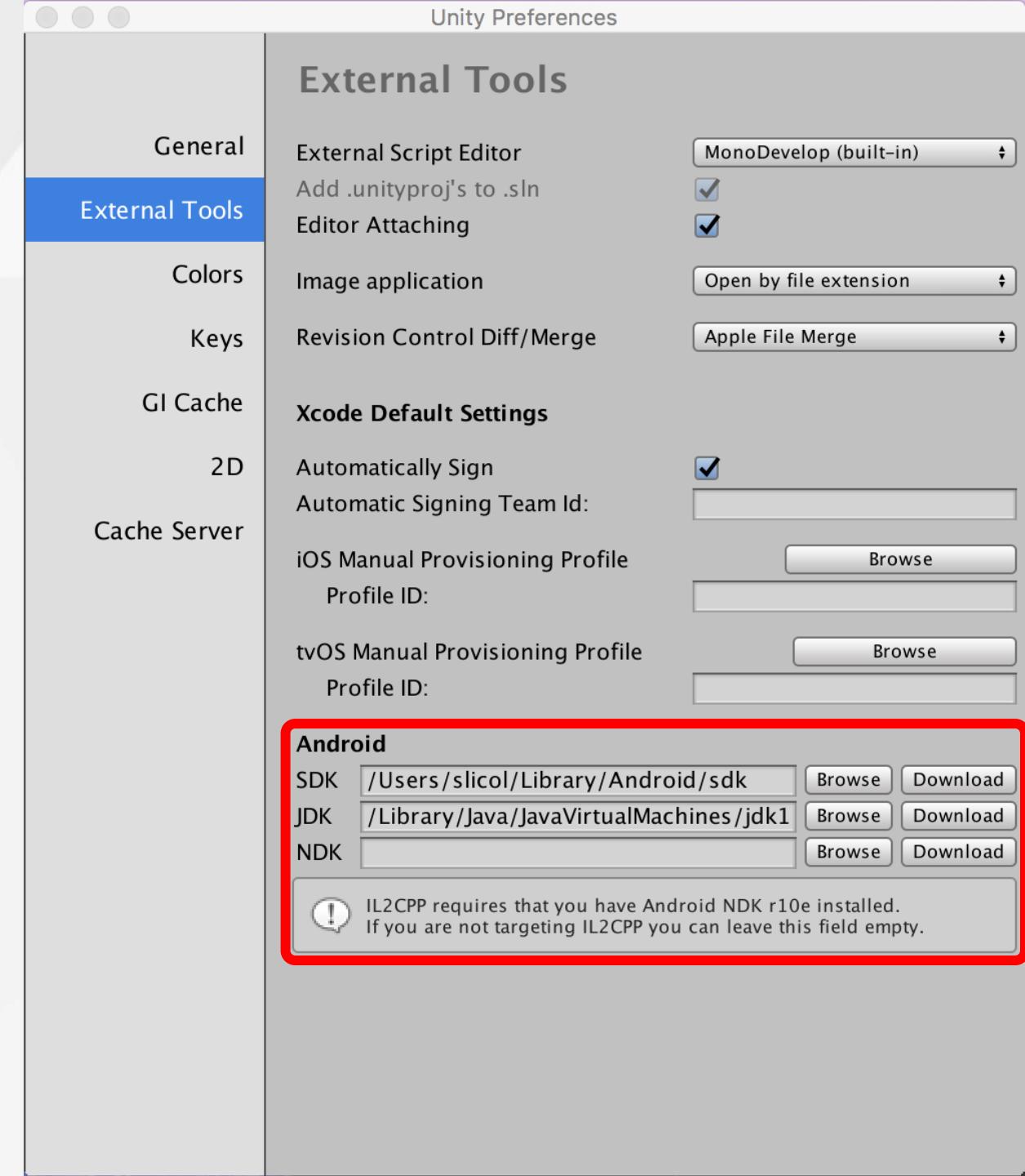
- 12.1 Android平台
 - 12.1.1 配置SDK和JDK

- SDK下载地址 :

<https://developer.android.com/studio/index.html#Other>

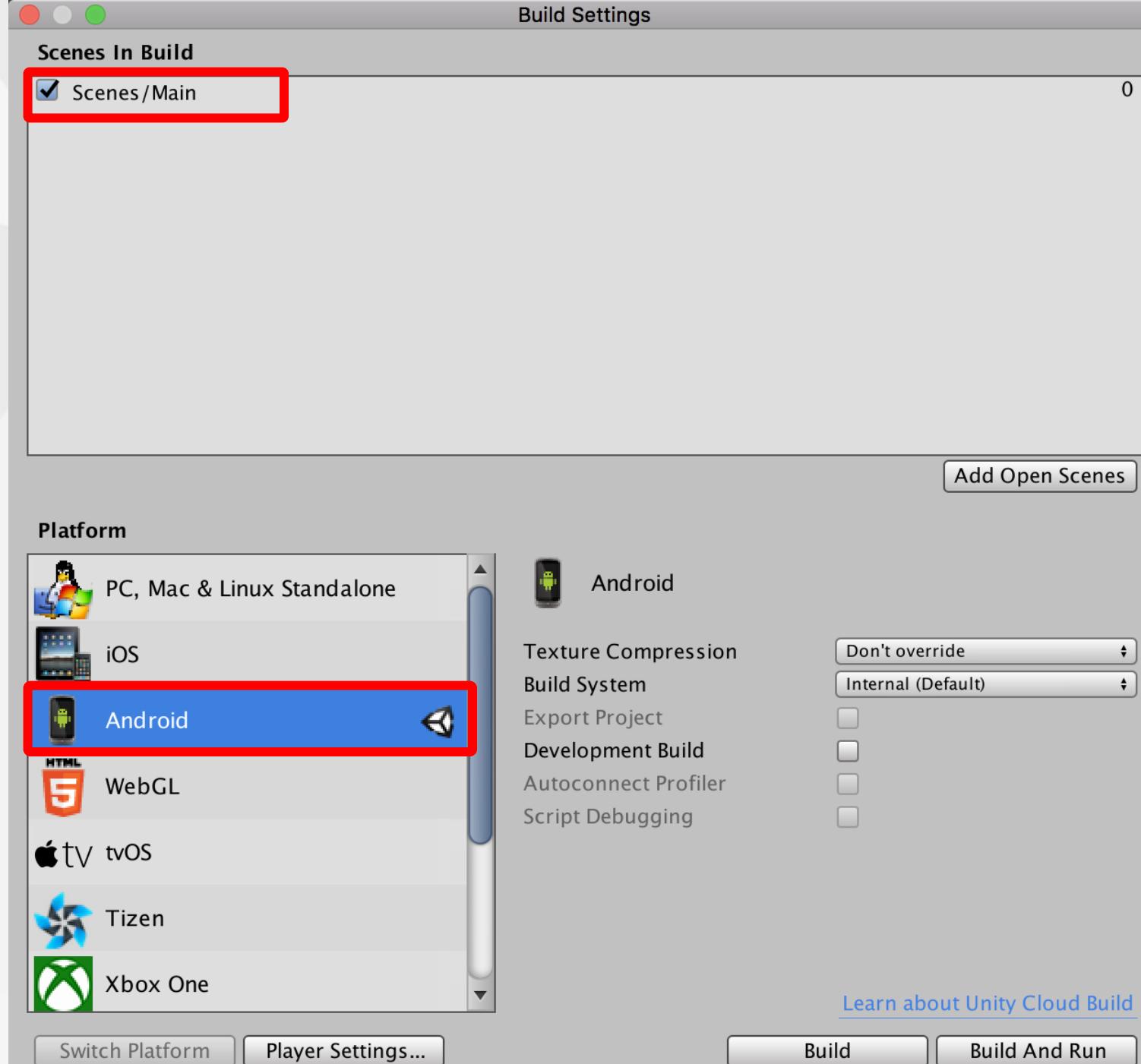
- JDK下载地址 :

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>



12、游戏打包

- 12.1 Android平台
 - 12.1.2 Build Settings
 - Scene In Build
 - Platform

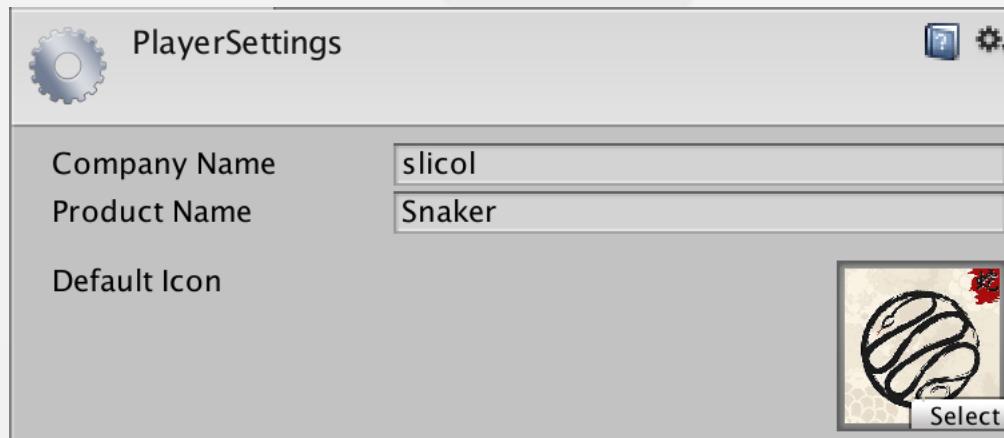


12、游戏打包

- 12.1 Android平台

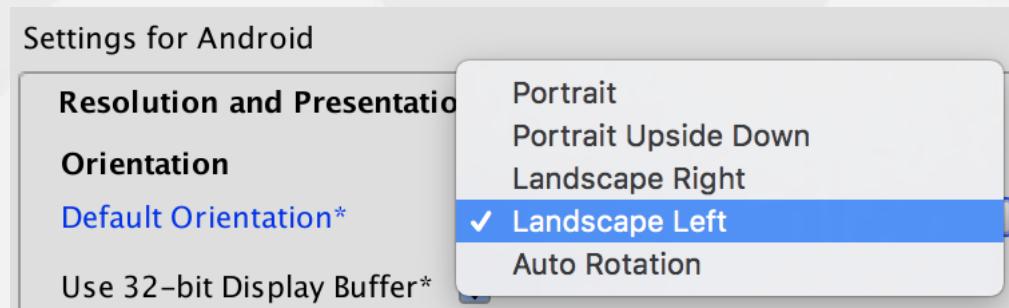
- 12.1.3 Player Settings

- Company Name
 - Product Name
 - Default Icon



12、游戏打包

- 12.1 Android平台
 - 12.1.4 Player Settings
 - Default Orientation



12、游戏打包

- 12.1 Android平台

- 12.1.5 Player Settings

- Package Name

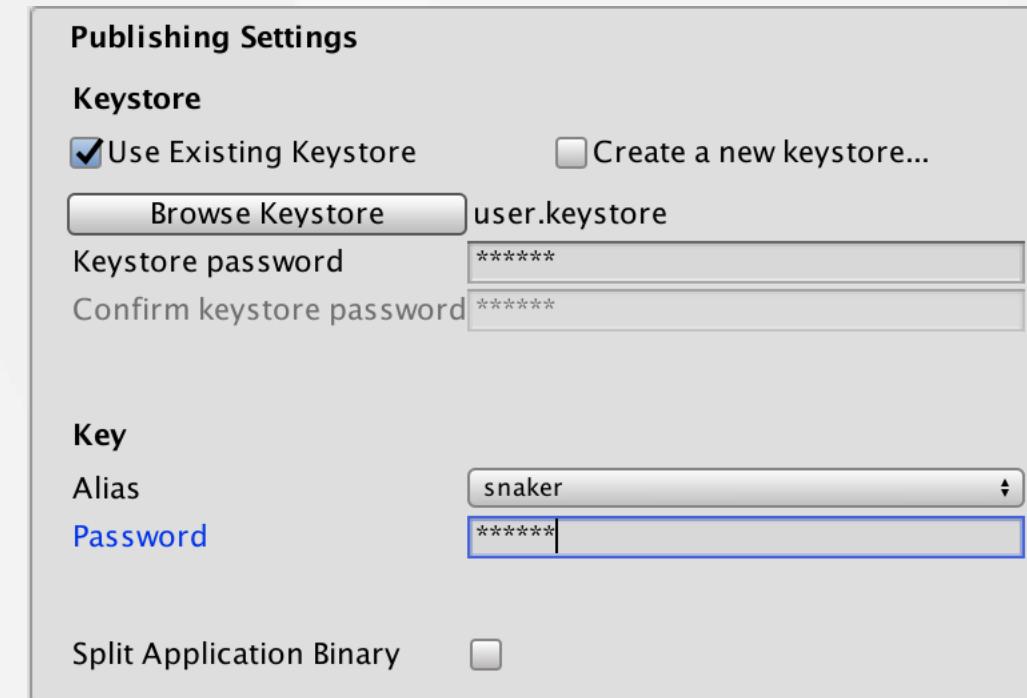
Package Name	com.slicol.snaker
Version*	1.0

- Scripting Define Symbols

Scripting Define Symbols*
EnableLog

12、游戏打包

- 12.1 Android平台
 - 12.1.6 Player Settings
 - 安装包签名
 - Keystore
 - Key



12、游戏打包

- 12.1 Android平台

- 12.1.7 Build

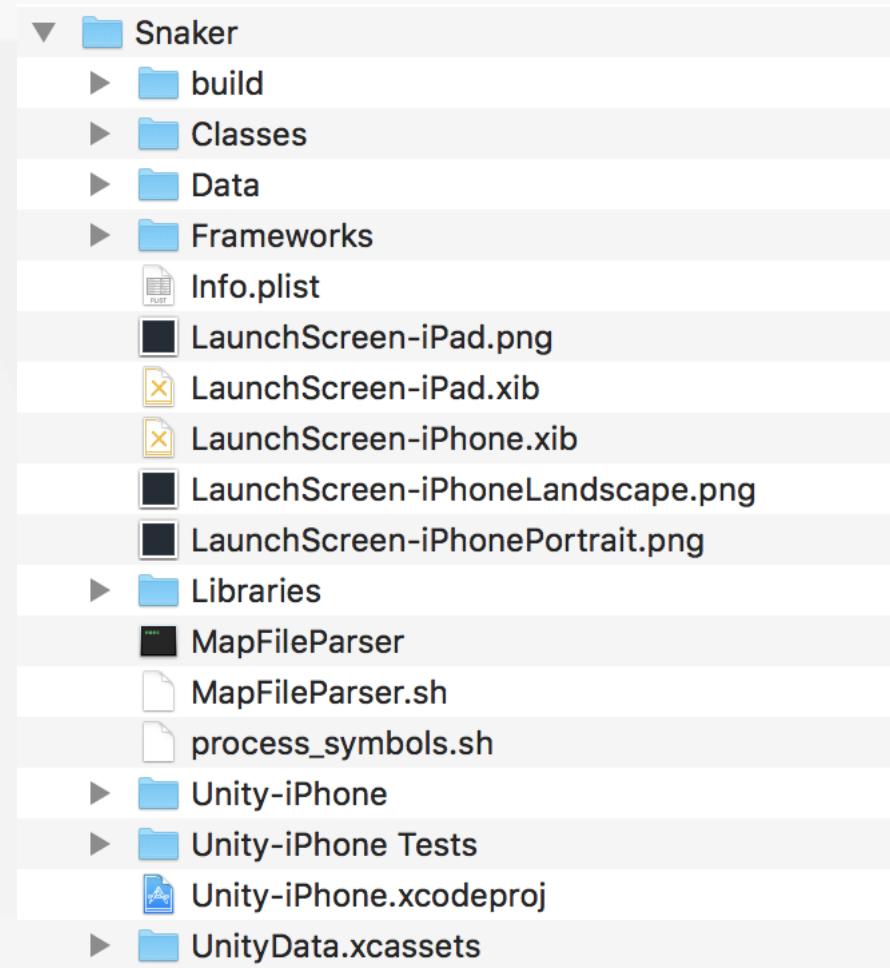
- 如果没有指定Key，则使用默认的Debug签名，在安装时会提醒安全风险。
 - 如果指定了Key，则生成带发布签名的APK，在安装时不会提醒安全风险。

12、游戏打包

- 12.2 IOS平台
 - 12.2.1 Unity相关设置（完全参考Android平台）
 - Build Settings
 - Company Name
 - Product Name（对应Xcode的Display Name）
 - Default Icon
 - Default Orientation
 - Package Name（对于Xcode的Bundle ID）
 - Scripting Define Symbols

12、游戏打包

- 12.2 IOS平台
 - 12.2.2 Untiy Build
 - 将Unity工程转换为Xcode工程



12、游戏打包

- 12.2 IOS平台

- 12.2.3 申请证书（用于安装包签名）

- (1) 购买Apple开发者帐号
 - (2) 打开【钥匙串访问】
 - (3) 打开【证书助理】



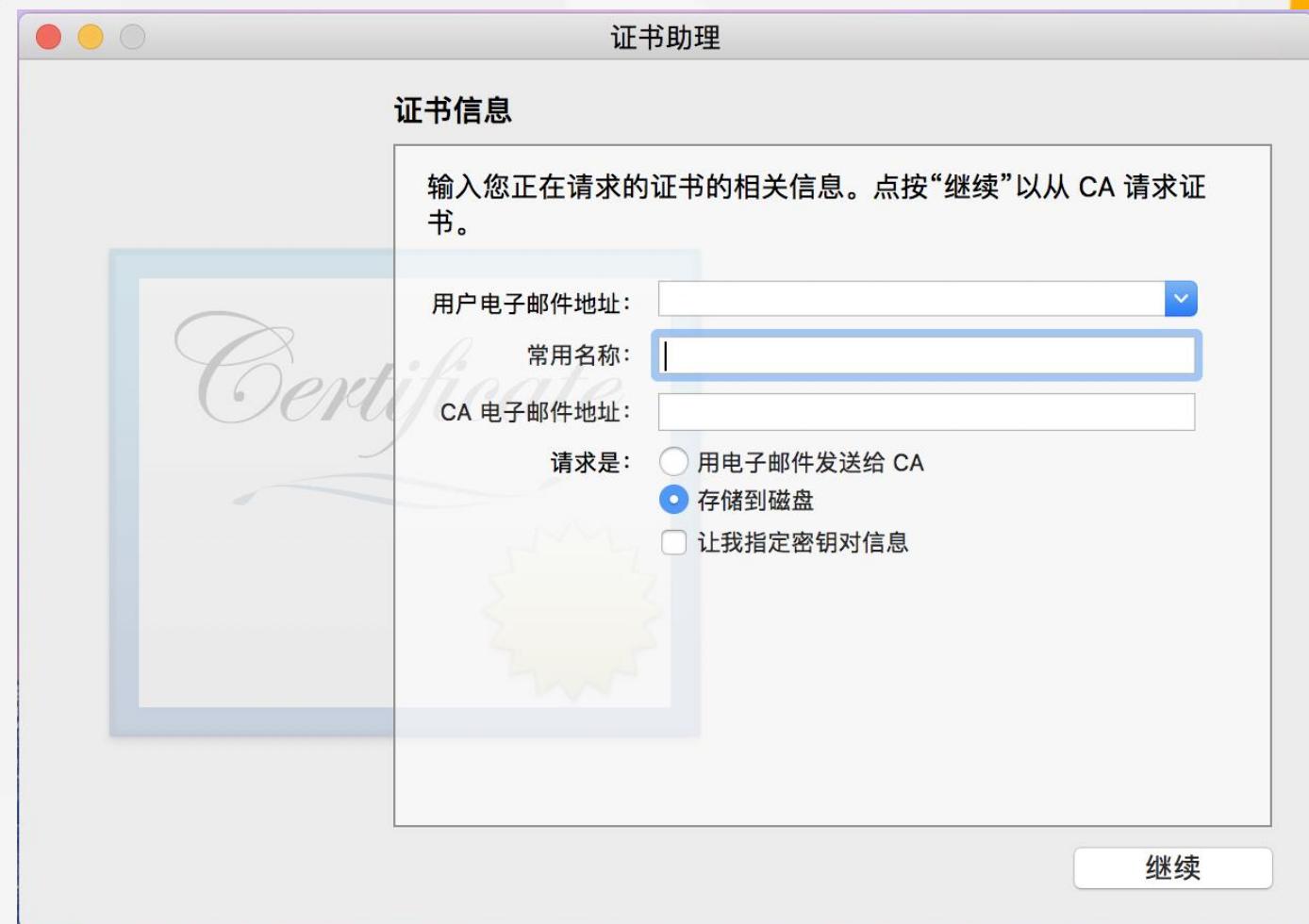
12、游戏打包

- 12.2 IOS平台

- 12.2.3 申请证书

- (4) 生成证书请求文件 : CertificateSigningRequest.certSigningRequest

- 电子邮件 (必填)
 - 常用名称 (必填)
 - CA电子邮件地址 (可不填)
 - 一般选择保存到磁盘
 - 手动上传给Apple开发者网站



12、游戏打包

- 12.2 IOS平台
 - 12.2.3 申请证书
 - (5) 生成证书

The screenshot shows the Apple Developer portal interface. The top navigation bar includes links for Developer, Discover, Design, Develop, Distribute, Support, Account, and a search bar. The main section is titled "Certificates, Identifiers & Profiles". On the left, there's a sidebar with a dropdown for "iOS, tvOS, watchOS", a "Certificates" section with a radio button for "All", and a "CSR Requests" section. The main content area is titled "iOS Certificates" and shows "2 Certificates Total". It has columns for Name, Type, and Expires. At the bottom, there's a form for uploading a CSR file, with a "Choose File..." button and a preview area showing "CertificateSigningRequest.certSigningRequest". A red box surrounds the CSR upload area, and a red square highlights the "+" button in the top right corner of the certificate list.

Developer Discover Design Develop Distribute Support Account Q

Certificates, Identifiers & Profiles ShengFu Tang ▾

iOS, tvOS, watchOS Certificates All CSR Requests

iOS Certificates

2 Certificates Total

Name	Type	Expires
------	------	---------

Upload CSR file.
Select .certSigningRequest file saved on your Mac.

Choose File... CertificateSigningRequest.certSigningRequest

12、游戏打包

- 12.2 IOS平台

- 12.2.4 创建AppID

- (1) App ID Description

- 这个AppID在AppID列表中显示的名字，它具有唯一性

- (2) 严格限定的AppID

- Explicit App ID**

- If you plan to incorporate app services such as Game Center, In-App Purchase, Data Protection, and iCloud, or want a provisioning profile unique to a single app, you must register an explicit App ID for your app.

- To create an explicit App ID, enter a unique string in the Bundle ID field. This string should match the Bundle ID of your app.

Bundle ID:

We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (*).

12、游戏打包

- 12.2 IOS平台

- 12.2.4 创建AppID

- (3) 为了方便各种小游戏开发，可以创建带通配符的AppID
 - 但是其功能会有些限制

- **Wildcard App ID**

This allows you to use a single App ID to match multiple apps. To create a wildcard App ID, enter an asterisk (*) as the last digit in the Bundle ID field.

Bundle ID:

Example: com.domainname.*

12、游戏打包

- 12.2 IOS平台
 - 12.2.4 创建AppID
 - 已经创建的AppID

iOS App IDs	
Name	ID
8 App IDs total.	
AppX	com.slicol.*
BubbleBreak	slicol.doodle.bubblebreak
BunnyRulerPro	com.slicol.BunnyRulerPro
com-slicol-BunnyRuler	com.slicol.BunnyRuler
Snaker	com.slicol.snaker

12、游戏打包

- 12.2 IOS平台
 - 12.2.5 添加Devices
 - 如果是开发版本（不需要在AppStore上架也能安装的版本）
 - 需要将安装该版本的设备添加进来

All Devices

+ **编辑** **搜索**

Name	Identifier
Slicol iPhone6	00000000-0000-0000-0000-000000000000
Slicol IPad Mini2	00000000-0000-0000-0000-000000000000
Slicol iPhone5s	00000000-0000-0000-0000-000000000000

12、游戏打包

- 12.2 IOS平台
 - 12.2.6 创建Provisioning Profiles
 - 该文件关联了以下信息：
 - 证书（开发证书OR发布证书）
 - AppID
 - Devices（如果是开发证书的话）

iOS Provisioning Profiles		
Name	Type	Status
4 profiles total.		
AppX_Development	iOS Development	● Active
AppX_Distribution	iOS Distribution	● Active
Snaker_Development	iOS Development	● Active
Snaker_Distribution	iOS Distribution	● Active

12、游戏打包

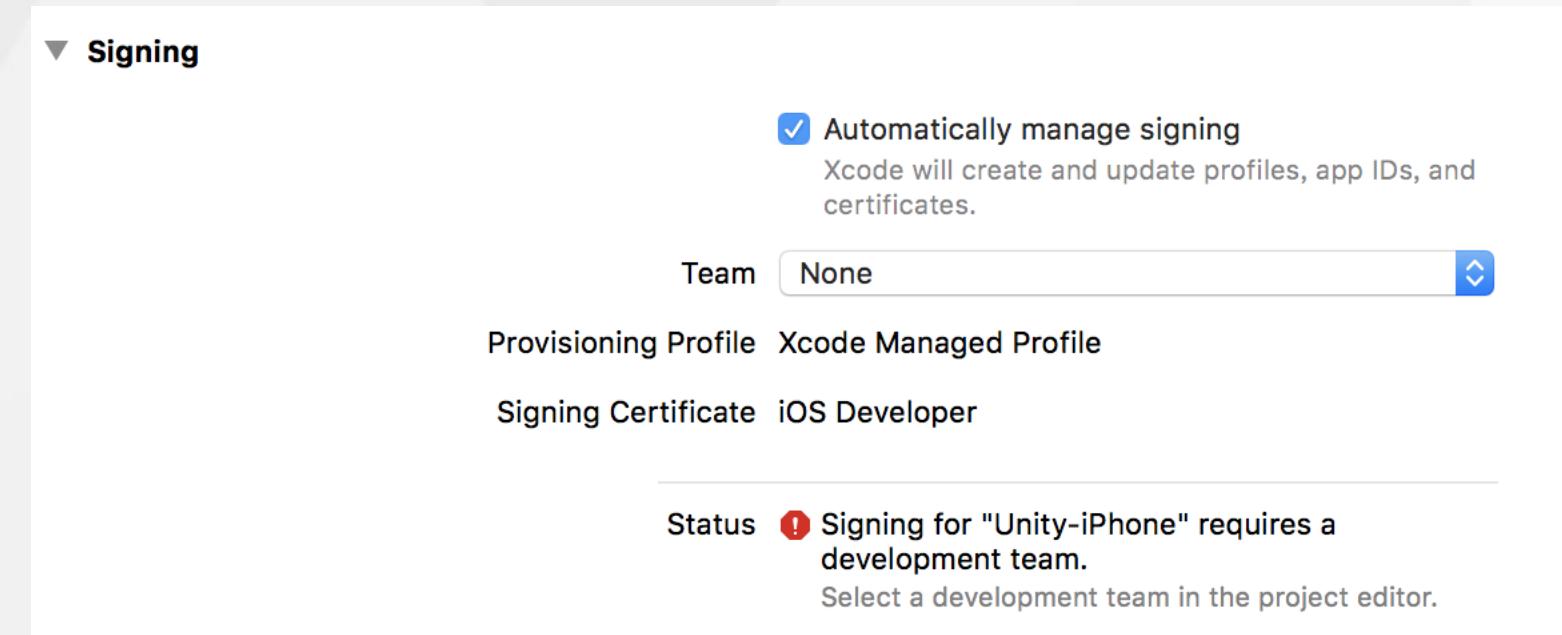
- 12.2 IOS平台
 - 12.2.7 安装证书
 - 在AppleDeveloper网站上下载证书到Mac上，双击即可安装



12、游戏打包

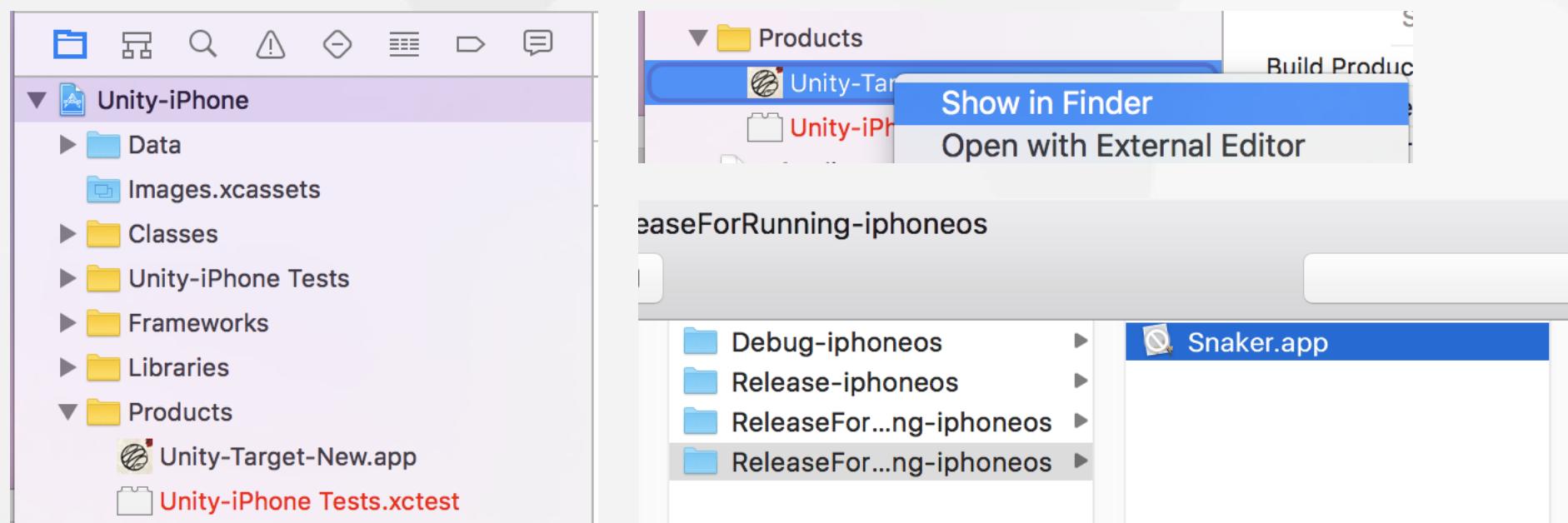
- 12.2 IOS平台

- 12.2.8 在Xcode里设置证书



12、游戏打包

- 12.2 IOS平台
 - 12.2.9 Build
 - 如果没有Link错误，最终会生成一个.app文件
 - 这个文件是在Xcode的目录下，比较深，可以直接右键ShowInFinder找出来



12、游戏打包

- 12.2 IOS平台
 - 12.2.10 打包成IPA
 - 在Unity工程的Build目录下创建一个新目录：Payload
 - 将上一步生成的Snaker.app文件Copy到Payload
 - 将Payload目录压缩成一个Zip文件：Payload.zip
 - 将这个Zip文件重命名为：Snaker.ipa

THANKS