**模块设计**

1. 游戏模块
   1. 模式选择
      1. 单人模式

点击后单人游玩游戏，进入选关界面

* + 1. 多人模式
       1. 随机匹配

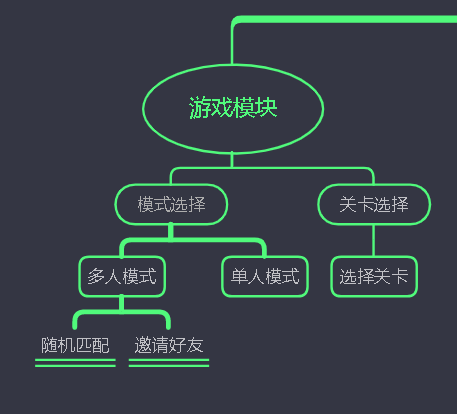
利用网络模块随机匹配队友

* + - 1. 邀请好友

从好友列表邀请在线好友进行多人游戏

* 1. 关卡选择
     1. 选关界面

从本地或者云端存档获取过关情况，然后展示成选关界面，玩家点击选关界面进入游戏。



1. 账号管理模块
   1. 云数据同步
      1. 读取存档

根据每个玩家的uid生成专属配置文件存储在云端，用于多设备的同步，同时会和本地存储的配置文件进行比对。

* 1. 登录/注册
     1. 注销登录

点击注销当前账号登录状态，返回登录界面

* + 1. 切换账号

返回登录界面，通过账号密码来切换其他账号

* 1. 好友管理
     1. 添加好友

玩家可以通过uid进行好友添加，在用户表内返回搜索结果，然后发送请求

* + 1. 邀请好友

邀请好友加入游戏



1. 行为模块
   1. 移动

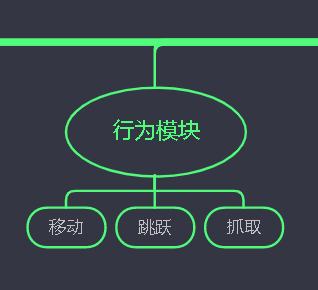
按键触发玩家移动事件，并播放相应动画和音效

* 1. 跳跃

按键触发玩家跳跃事件，并播放相应动画和音效

* 1. 抓取

按键触发玩家抓取事件，并播放相应动画和音效



1. 关卡模块
   1. 过关成功
      1. 前往下一关

点击进入下一关游戏界面

* + 1. 设置

跳出设置界面

* + 1. 退出

返回选关界面

* 1. 游戏失败
     1. 重新开始

重新加载本关卡

* + 1. 设置

跳出设置界面

* + 1. 退出

返回选关界面

* 1. 暂停
     1. 继续游戏

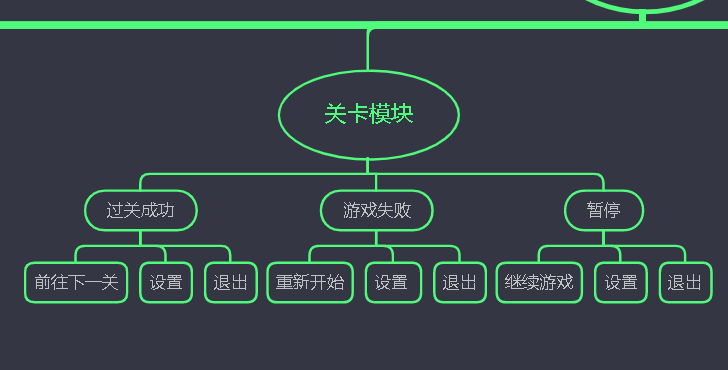
关闭暂停界面

* + 1. 设置

跳出设置界面

* + 1. 退出

返回选关界面

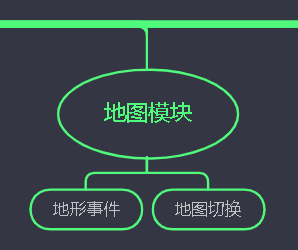


1. 地图模块
   1. 地形事件

根据不同地形的特色有特殊的效果，冰面、地刺、炸弹障碍物等等，触发特殊的事件，摩擦力减小，受伤，爆炸死亡之类。

* 1. 地图切换

加载关卡地图，以及UI界面



1. 网络模块
   1. 匹配队友

根据网络模块筛选在线的正在匹配的玩家进行随机匹配，或者连接受邀请的好友

* 1. 信息交互
     1. 位置信息

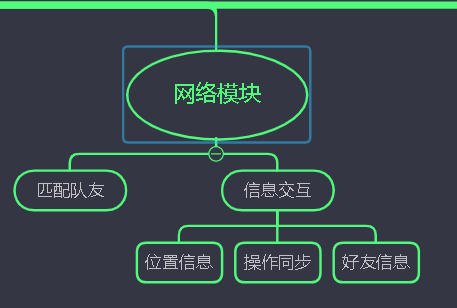
通过网络模块传输角色的位置信息，实现画面同步

* + 1. 操作同步

通过网络模块实时传输角色的操作输入，实现操作同步

* + 1. 好友信息

访问网络数据库调用好友个人信息和在线情况



1. 设置模块
   1. 音乐设置
      1. 背景音乐设置

设置背景音乐的大小

* + 1. 特效音乐设置

设置特效音乐的大小

* 1. 画面设置
     1. 清晰度设置

设置画面清晰度和分辨率，节省资源开支

* + 1. 亮度设置

设置画面亮度

* + 1. 高级设置

设置其他画面选项，抗锯齿，渲染等选项

* 1. 操作设置
     1. 操作模式

切换双键模式和三键模式

* + 1. 按键位置

设置按键的位置，贴合玩家需求

* + 1. 按键大小

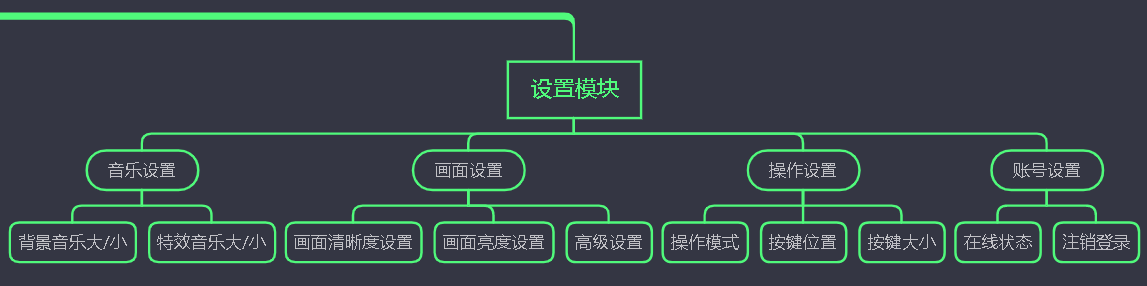
设置按键大小，贴合玩家需求

* 1. 账号设置
     1. 在线状态

可以设置在线状态来避免被邀请

* + 1. 注销登录

注销登录，退出游戏并返回登录界面



**层次设计**

1. 网络层

通过调用网络模块来进行网络链路通信，传输数据

1. 数据层

所有的输入和输出都在此层，将所有数据收集生成数据字典，其他层次通过该层获取需要的数据

1. 蓝图层

所有的脚本构成蓝图层，包括UI，逻辑，事件等等。脚本或称蓝图获取数据层传来的数据，然后做出对应的事件响应。同时也会回传数据用来输出。

1. 行为层

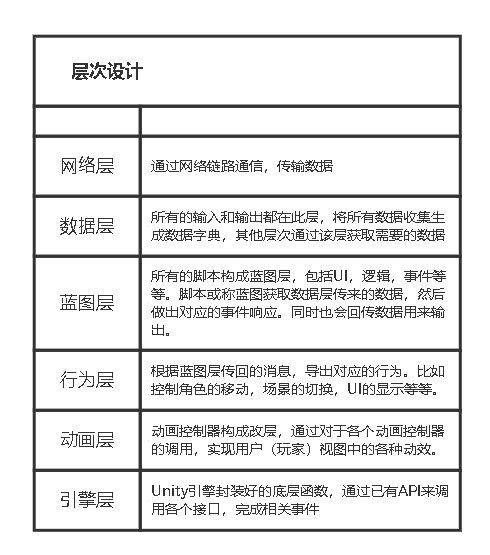
根据蓝图层传回的消息，导出对应的行为。比如控制角色的移动，场景的切换，UI的显示等等。

1. 动画层

动画控制器构成改层，通过对于各个动画控制器的调用，实现用户（玩家）视图中的各种动效。

1. 引擎层

Unity引擎封装好的底层函数，通过已有API来调用各个接口，完成相关事件



**系统安全性**

Unity 支持 Mono 和 IL2CPP 两种编译模式。使用 Mono 编译的游戏，它将 C#脚本代码编译为 IL 中间语言打包到游戏客户端，在游戏执行时候编译为汇编代码。这类中间语言存在容易被反编译为 C#源代码的风险。后来基于安全性和执行效率方面的考虑，Unity 支持了 IL2CPP 编译，大大提升了游戏安全性，但还是存在被攻击的风险。

C# 反编译相关的工具有很多，比如传统的 ILDASM、PEBroswerDbg、GrayWolf、XenoCode 等。

我们选择使用 IL2CPP 模式编译，游戏的脚本代码没有了，脚本代码被编译成了 Native 代码发布。前面提到的工具都失效了，安全性得到了一定的提升。但是 IL2CPP 编译后会生成一个 global-metadata.dat 文件，这个文件里面包含了大量的符号信息。

我们采用UPS来进行安全性保证。方案主要做了两个方面的保护：Mono 保护，IL2CPP 保护。Mono 保护的对象是游戏的主逻辑 dll：Assembly-CSharp.dll 和 Assembly-CSharp-firstpass.dll。IL2CPP 保护的对象是 metadata 文件：global-metadata.dat

**系统健壮性**

采用传统方式更新，热更仅仅用于紧急更新 / 希望战斗等性能敏感部分也能更新

使用xLua（性能在Lua系列中一般，但额外支持HotFix，可以更多使用C#代码）

使用ILRuntime解释器，用ILRuntime是比较正统的解决方案，这个语言在提供代码热更的方案的时候，会主动换成另外一个脚本语言