项目文档

项目小组 10小组

小组成员 范宇扬 向成 张震

联系方式 18306012126

重庆师范大学软件工程系

摘要

请以简练的文字，介绍项目的背景、目标、开发过程、结果等内容。请以简练的文字，介绍项目的背景、目标、开发过程、结果等内容。请以简练的文字，介绍项目的背景、目标、开发过程、结果等内容。

请以简练的文字，介绍项目的背景、目标、开发过程、结果等内容。请以简练的文字，介绍项目的背景、目标、开发过程、结果等内容。请以简练的文字，介绍项目的背景、目标、开发过程、结果等内容。请以简练的文字，介绍项目的背景、目标、开发过程、结果等内容。请以简练的文字，介绍项目的背景、目标、开发过程、结果等内容。

请以简练的文字，介绍项目的背景、目标、开发过程、结果等内容。请以简练的文字，介绍项目的背景、目标、开发过程、结果等内容。

关键词：关键字 3～5 个，逗号分割

日期	修改	描述	作者
2021 年 11 月 25 日	0.1.0	初始版本	
2021 年 12 月 8 日	0.2.0	大改：“黑盒”，无界面描述，描述细节	
2021 年 12 月 19 日	0.2.1	更新参与者名称及用况图	
2021 年 12 月 26 日	0.3.0	对系统所要做的事情进行详细描述	
2022 年 1 月 1 日	0.3.1	增加系统要做的事情，同时避免抢占设计	
2022 年 1 月 10 日	0.3.2	增加互动交流处关于点赞，收藏的细节	
2022 年 3 月 21 日	0.3.3	修改获取素材的细节	
2022 年 3 月 28 日	0.4.0	用况变为交替进行	
2022 年 4 月 5 日	0.4.1	添加业务流程切换的有界备选流描述	
2022 年 4 月 9 日	0.4.2	添加社交细节，修改版本号	
2022 年 4 月 26 日	0.4.3	添加关于上传的素材的细节	
2022 年 6 月 18 日	0.4.4	笔记交互细节	

目录

第1章 立项

重要说明：本文档的结构不是标准的样例，请根据你们项目的情况自行编写。

1.1. 项目起源与提案

当前游戏市场中，大多数主流多人对战游戏都面临着外挂泛滥、网络延迟高、配置要求苛刻等问题，影响了玩家的核心竞技体验。同时，现有游戏往往过于复杂，加入了大量商业化内容，忽视了纯粹的游戏乐趣。

本项目旨在开发一款专注于核心竞技体验的第三人称动作 PVP 游戏。我们将基于 Unreal Engine 5 引擎，打造一个公平、流畅、易上手的多人对战平台。游戏将采用权威服务器架构确保公平性，优化网络同步机制降低延迟，并通过简洁的设计让玩家能够快速享受对战的乐趣。

1.2. Business Case

本项目的核心价值在于为玩家提供一个纯粹、公平的竞技环境。通过采用权威服务器架构和严格的防作弊机制，我们将从根本上杜绝外挂问题，确保每位玩家都能在公平的环境下进行游戏。同时，游戏将专注于核心对战体验，避免复杂的养成系统和付费门槛，让玩家能够快速投入并享受对战的乐趣。

从市场角度来看，虽然大型商业游戏占据主导地位，但仍存在对精品化、专注型竞技游戏的明确需求。我们的目标用户是那些追求纯粹竞技体验、厌倦了外挂和复杂系统的核心玩家。通过精准定位这一细分市场，项目有望获得稳定的用户群体，并通过持续的内容更新和赛事运营，建立长期的竞争优势。

第2章愿景

重要说明：本文档的结构不是标准的样例，请根据你们项目的情况自行编写。

2.1. 问题陈述

1. 问题一

要素	描述
问题	许多小型团队或独立开发者创作的多人对战游戏，常因网络同步不佳、核心玩法缺乏深度或优化不足，导致玩家体验挫败，难以形成稳定社区
影响	热爱 PVP 对战但设备配置不高、或寻求轻松竞技体验的玩家
结果	使他们难以找到一款既能满足竞技需求，又不会带来过大学习与设备负担的游戏
优点	一个成功的解决方案将提供一个网络稳定、玩法直观且有深度、性能友好的多人动作游戏平台

2.2. 涉众与用户

1. 涉众

涉众	涉众类型	简要描述
核心开发成员	开发者、愿景制定者	负责编程、美术、设计的核心成员，决定游戏的核心玩法和技术方向。
游戏测试者（内测组）	顾问用户	由熟悉同类游戏的玩家组成，在开发早期进入，负责反馈手感和平衡性。
同好游戏社群	社区、间接涉众	项目在开发期间希望吸引和服务的核心玩家群体
开源技术社区	支持者、协作者	为项目所依赖的 UE5 引擎、插件等提供技术支持和问题解答的线上社区。

2. 用户

竞技型玩家	享受与人斗的乐趣，喜欢研究战术。	获得公平的竞技体验，通过技巧和策略战胜对手，追求更高的天梯排名。
娱乐型玩家	主要与朋友组队，享受合作乐趣。	能快速开始一局游戏，操作顺手，与朋友有良好的配合空间，获得轻松愉快的体验。

2.3. 关键涉众和用户的需要

编号	用户需求	说明
UR-01	轻松加入同一局域网下的游戏	玩家无需复杂配置，即可快速发现并加入好友创建的房间。
UR-02	角色移动流畅、响应及时	操作指令应立即转化为游戏内的角色动作，无明显延迟或卡顿感。
UR-03	进行基本的攻击并看到清晰的反馈	攻击动作应顺畅，命中敌人时有明确的视觉或音效提示。
UR-04	清晰地了解游戏状态	能够随时便捷地查看自身血量、技能状态、胜负条件等关键信息。

2.4. 产品概述

1. 产品定位陈述

for	热爱多人对抗、追求纯粹竞技乐趣的玩家
who	对现有游戏的商业化、高门槛或糟糕优化感到失望
the	是一款由爱好者开发的第三人称 PVP 游戏
That	专注于提供流畅的操作、清晰的胜负反馈和富有战术深度的团队对抗

2. 完整的产品概述

本产品是一款专注于核心对战体验的第三人称动作游戏。玩家可以选择不同特性的英雄，在专门设计的紧凑地图中进行自由对抗。游戏将确保在主流硬件上流畅运行，并通过服务器权威架构保障对战的公平性。我们的目标是打造一个能让玩家专注于竞技本身、并乐于与朋友分享的优质游戏体验。

2.5. 产品特性

特性 ID	特性名称	描述
FEAT-01	无障碍局域网联机	提供简洁的房间创建与发现功能，使玩家能轻松与局域网内的朋友开始游戏。
FEAT-02	流畅的角色控制	实现符合玩家预期的第三人称角色移动、镜头控制和跳跃机制。
FEAT-03	直观的基础战斗	提供近战和/或远程攻击方式，并配有清晰的攻击动画与受击反馈。

FEAT-04	明确的胜负体系	通过生命值管理与死亡机制，以及简单的胜利条件（如团队击杀数），清晰地界定每局游戏的进程与结果。
FEAT-05	基础信息呈现	通过游戏内 HUD 和菜单界面，向玩家展示必要的状态信息和操作入口。

2.6. 其他产品需求

类型	需求	描述
性能	高帧率运行	在局域网环境下，游戏客户端应保持帧率稳定在 60 FPS 以上，以确保操作响应的流畅性。
性能	低操作延迟	核心游戏操作（如移动、攻击）的输入响应延迟应足够低，使玩家无法感知到明显的操作滞后。
可靠性	主机稳定性	作为游戏主机（服务器）的客户端不应发生非预期的崩溃
可靠性	网络容错	当有客户端异常断开连接时，不应导致主机或其他客户端崩溃。游戏应能妥善处理网络异常，如提供重连机制或友好的错误提示。
可用性	操作符合惯例	游戏的核心操作逻辑（移动、攻击、技能释放等）应符合游戏的惯例，降低玩家的学习成本。
可用性	界面直观	用户界面（UI）应简洁直观，关键信息（如血量、技能冷却）一目了然。
兼容性	Windows 平台支持	游戏客户端必须能够在 Windows 10/11 操作系统上正常运行。

第3章 用况建模

重要说明：本文档的结构不是标准的样例，请根据你们项目的情况自行编写。

请确定系统所有的用况，写出它们的简要描述。然后，选择某个用况，进行列表式提纲～完整描述的用况开发。

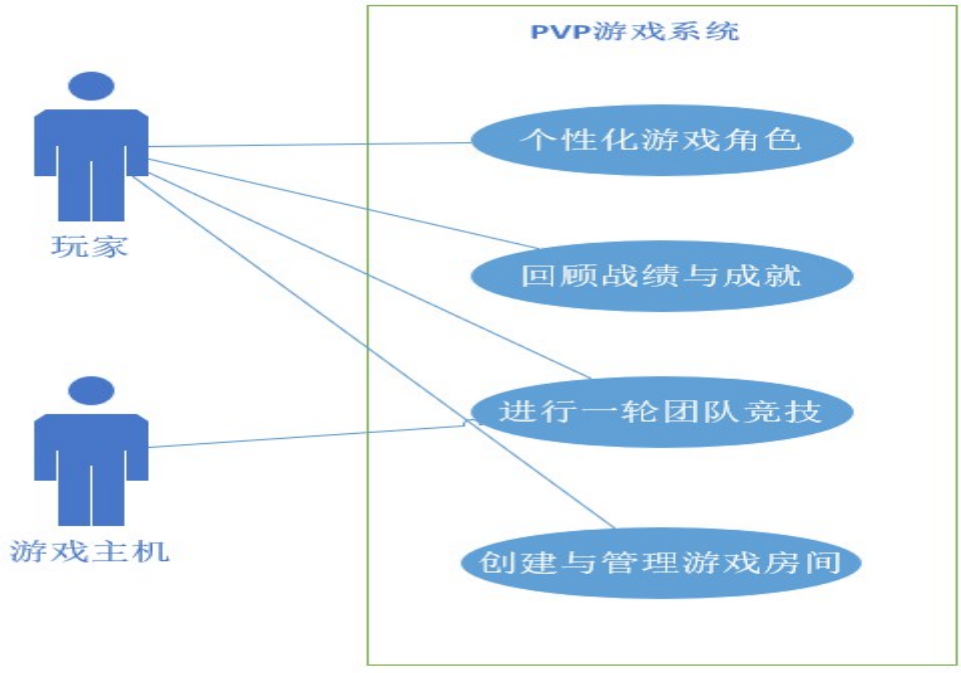
3.1. 术语表

名称	定义
玩家	控制游戏角色进行对战的用户。一个玩家实例对应一个游戏客户端。
游戏会话 (Game Session)	一次从创建房间到所有玩家退出、游戏结束的完整对战过程。包含房间阶段和对战阶段
主机 (Host)	运行游戏逻辑权威版本的客户端。负责游戏规则执行、状态同步和最终裁决。
客户端 (Client)	连接至主机，接收同步信息并处理玩家输入的玩家机器。
网络复制	UE5 中用于将 Actor 的属性和函数调用从服务器（主机）同步到客户端的技术机制。
角色	玩家在游戏中控制的虚拟实体
重生	玩家角色死亡后，经过一段等待时间，在指定重生点重新生成的过程
局域网对战	在本地网络范围内进行的联机对战，以简化网络配置并保证低延迟
战绩	对玩家单场或长期游戏表现的数据化记录，如击杀数、胜利场次等。
成就	玩家在达成特定游戏目标后获得的荣誉性奖励与标识。

3.2 主要用况

编号	名称	描述
UC-01	进行一轮战局竞技	玩家参与并完成一轮完整的团队对抗，从加入战局到获得最终胜负结果。这是游戏核心价值的体现。
UC-02	创建与管理游戏房间	玩家为自己和他人创建一场对战，并通过设置规则、管理玩家来准备一场游戏。
UC-03	个性化游戏角色	玩家在战斗之余，配置不同的英雄、技能或外观定义自己的战斗风格和形象。
UC-04	追踪与展示成长历程	玩家查看自己的战绩统计、成就与历史记录，获得持续的游戏反馈与荣誉感

3.2. 用况的描述（完整描述）#说明，不同阶段，用况应该有不同版本。



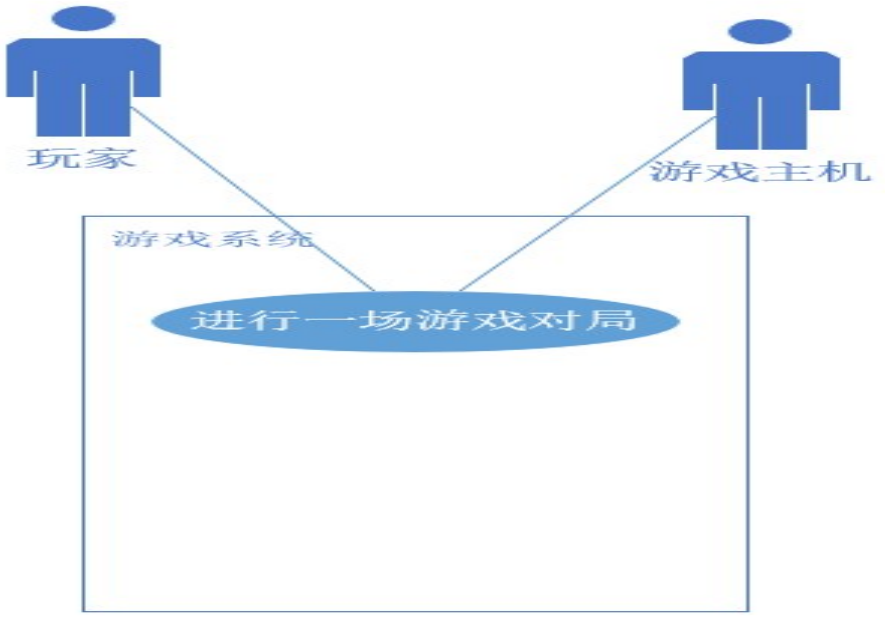
用况 **UC-01**：进行一轮团队竞技

1. 简要描述

本用况描述了玩家参与并完成一轮完整 PVP 对战的完整体验。玩家通过与队友协作、与敌

人交战，最终达成胜利条件，获得竞技乐趣与反馈。

2. 用况图



3. 前置条件：

玩家已启动游戏并进入主菜单。
存在一个已创建且未开始的游戏房间，或玩家有能力自己创建房间。

4. 基本流：

1. 玩家通过用况“创建与管理游戏房间”成功进入一个准备就绪的房间。
2. 房间主机开始游戏，系统为所有玩家加载游戏地图、分配队伍并生成角色。{游戏开始}
3. 玩家控制角色在战场中移动，寻找战术位置和敌人。{对战进行中}
4. 玩家与敌方角色相遇，通过普通攻击和技能进行交战。
5. 玩家通过精准的操作和战术配合，击败敌方角色，或根据模式要求达成目标。
6. 系统持续监控游戏状态（如团队得分、剩余时间）。
7. 当一方达成胜利条件（如达到目标分数），系统判定游戏结束。{游戏结束}
8. 系统显示本局游戏的详细结算界面（包括胜负、战绩数据、奖励结算）。
9. 所有玩家返回游戏大厅，用况结束。

子流

（此用况通过系统内部机制处理移动、攻击等底层交互，这些是实现核心价值的步骤，而非独立子流。）

1.5. **备选流：**

A1. **玩家中途退出：** 在基本流步骤 3-7 期间，如果有玩家断开连接，系统记录其为“逃跑”，其余玩家继续游戏。

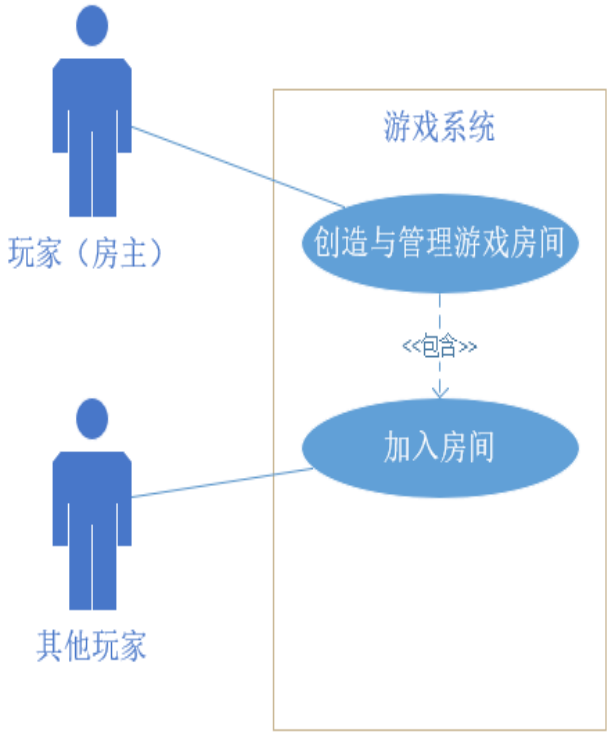
A2. **战斗失败与重生：** 在基本流步骤 5，如果玩家被敌方击败，角色进入死亡状态，系统在短暂等待后于重生点使其**重生**，玩家继续参与战斗。

（**2**）创建与管理游戏房间

1.1. **简要描述**

玩家为自己和他人创建一场对战，并通过设置规则、管理玩家来准备一场游戏。此用况为“进行一轮团队竞技”准备了必要的系统状态。

1.2. 用况图



1.3. **前置条件：**

玩家已启动游戏并进入主菜单。

1.4基本流：

1. 玩家在主菜单中选择"创建房间"。{创建房间}

2.系统创建一个默认设置的房间，并赋予玩家"房主"权限。

3.玩家可以自定义房间参数，如地图、游戏模式、胜利条件、人数限制等。{自定义设置}

4.其他玩家通过局域网发现并加入该房间。

5.房主可以管理房间内的玩家（如：移除玩家）。

6.当房间人数满足条件且房主选择"开始游戏"时，用况成功完成。系统状态转变为"游戏准备就绪"，并触发"进行一轮团队竞技"用况。

1.5子流

S1. 浏览与加入房间：

1. 玩家在主菜单中选择“浏览房间”。
2. 系统显示局域网内所有可加入的房间列表。
3. 玩家选择一个房间并申请加入。{加入房间}
4. 房主接受申请后，玩家加入房间并等待游戏开始。

1.6. 备选流：

A1. 加入房间失败： 在子流S1步骤3，如果房间已满或房主拒绝申请，系统提示玩家并返回到房间列表。

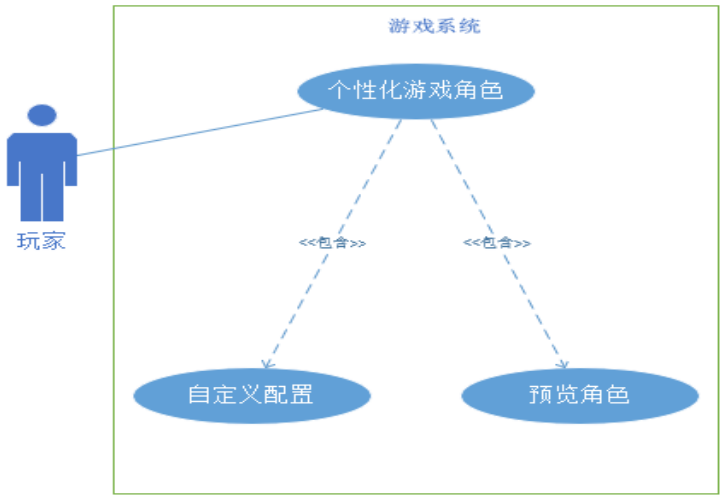
A2. 房主解散房间： 在基本流任何步骤，如果房主选择解散房间，则房间被关闭，所有玩家返回大厅，用况结束。

（3）个性化游戏角色

1.1 简要描述

玩家在战斗之余，通过解锁和配置不同的英雄、技能或外观，来定义自己的战斗风格和形象。此用况满足了玩家的个性化需求和长期成长感。

1.2 用况图



1.3. 前置条件：

玩家已成功登录系统。

1.4. 基本流：

1. 玩家在主菜单或大厅中选择“英雄”或“仓库”选项。
2. 系统显示玩家已拥有的所有角色、技能、武器和外观。
3. 玩家浏览可用的个性化选项。{浏览配置}
4. 玩家选择一个角色作为默认出战英雄。
5. 玩家为该角色搭配不同的技能组合、武器或外观。{自定义配置}
6. 系统保存此配置，并将在下一局游戏中生效。
7. 用况结束。

1.5 子流

S1. 预览角色：

1. 在基本流步骤 3，玩家选择预览一个角色或外观。
2. 系统在展示界面中渲染该角色的 3D 模型和特殊动作。
3. 子流结束。

1.6. 备选流：

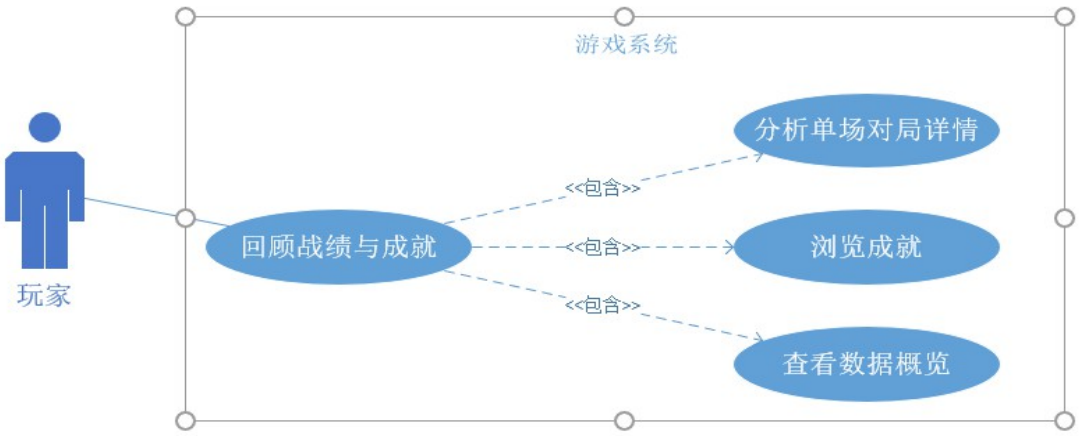
A1. 资源不足： 在基本流步骤 4 或 5，如果玩家尝试使用一个尚未解锁的项目，系统提示“需要先解锁该内容”。

（4）回顾战绩与成就

1.1. 简要描述

玩家在游戏结束后或在大厅中，查看自己及他人的详细战绩数据、已解锁的成就与排名信息。这为玩家提供了持续的反馈和追求目标，是维持长期游玩动力的关键

1.2. 用况图



1.3. 前置条件：

玩家已成功登录系统。

1.4. 基本流：

1.玩家在主菜单或结算界面选择“战绩”、“统计”或“个人资料”选项。{访问数据入口}

2. 系统显示玩家的核心数据概览，如总场次、胜率、常用英雄、总击杀/死亡/助攻（KDA）等。{查看数据概览}

3. 玩家可以切换到“成就”页面，查看已解锁和未解锁的成就及其完成进度。{浏览成就}

4. 玩家可以切换到“战斗记录”页面，逐条查看近期的详细对局记录，包括每局的得分、英雄使用情况等。

5. 玩家可以选择查看好友或排行榜上其他玩家的公开资料以进行对比。

6. 用况结束。

1.5 子流

◦ S1. 分析单场对局详情：

1. 在基本流步骤4，玩家从战斗记录列表中选择一场特定的对局。

2. 系统显示该场对局的详细报告，包括：双方队伍组成、每个玩家的详细数据（伤害量、承受伤害、关键操作）、时间线事件（如重要击杀、目标占领）等。

3. 玩家可以浏览这些深度数据以复盘自己的表现。

4. 子流结束。

1.6. 备选流：

A1. 无历史记录： 如果玩家是新手，尚无任何对战数据，系统在相应页面显示“暂无数据，快去进行第一场对战吧！”等引导性提示。

第4章 需求分析

4.1. 健壮性分析

4.1.1 用况UC-01：进行一轮团队竞技

类名	类型	职责说明
BattleUI	边界类	接收：玩家操作输入（移动、攻击、技能） 显示：游戏HUD、角色状态、战斗反馈 提供：游戏结算界面展示
BattleController	控制类	协调：整个战斗流程和状态同步 处理：游戏规则执行、胜负条件判定 知晓：当前游戏阶段、玩家操作队列
Player	实体类	知晓：生命值、坐标位置、装备状态、技能冷却 能做：执行移动、进行攻击、释放技能、更新状态
GameSession	实体类	知晓：队伍得分、游戏状态、剩余时间、玩家列表 能做：更新比分、管理玩家状态、判定游戏结束
Map	实体类	知晓：地形数据、重生点位置、碰撞信息 能做：验证位置合法性、提供重生坐标

4.1.2 用况 **UC-02**：创建与管理游戏房间

类名	类型	职责说明
LobbyUI	边界类	提供：房间列表展示、创建界面、设置面板 接收：玩家创建/加入房间请求

		显示：房间内玩家列表和状态
RoomController	控制类	管理：房间完整生命周期（创建→配置→开始→解散） 处理：玩家加入/退出请求、房间状态更新 协调：游戏开始前的准备工作
GameRoom	实体类	知晓：房间配置、玩家列表、房间状态 能做：添加/移除玩家、验证设置合法性
NetworkManager	控制类	处理：局域网广播和发现 管理：客户端连接状态 知晓：网络拓扑和可用连接

4.1.3 用况 **UC-03**：个性化游戏角色

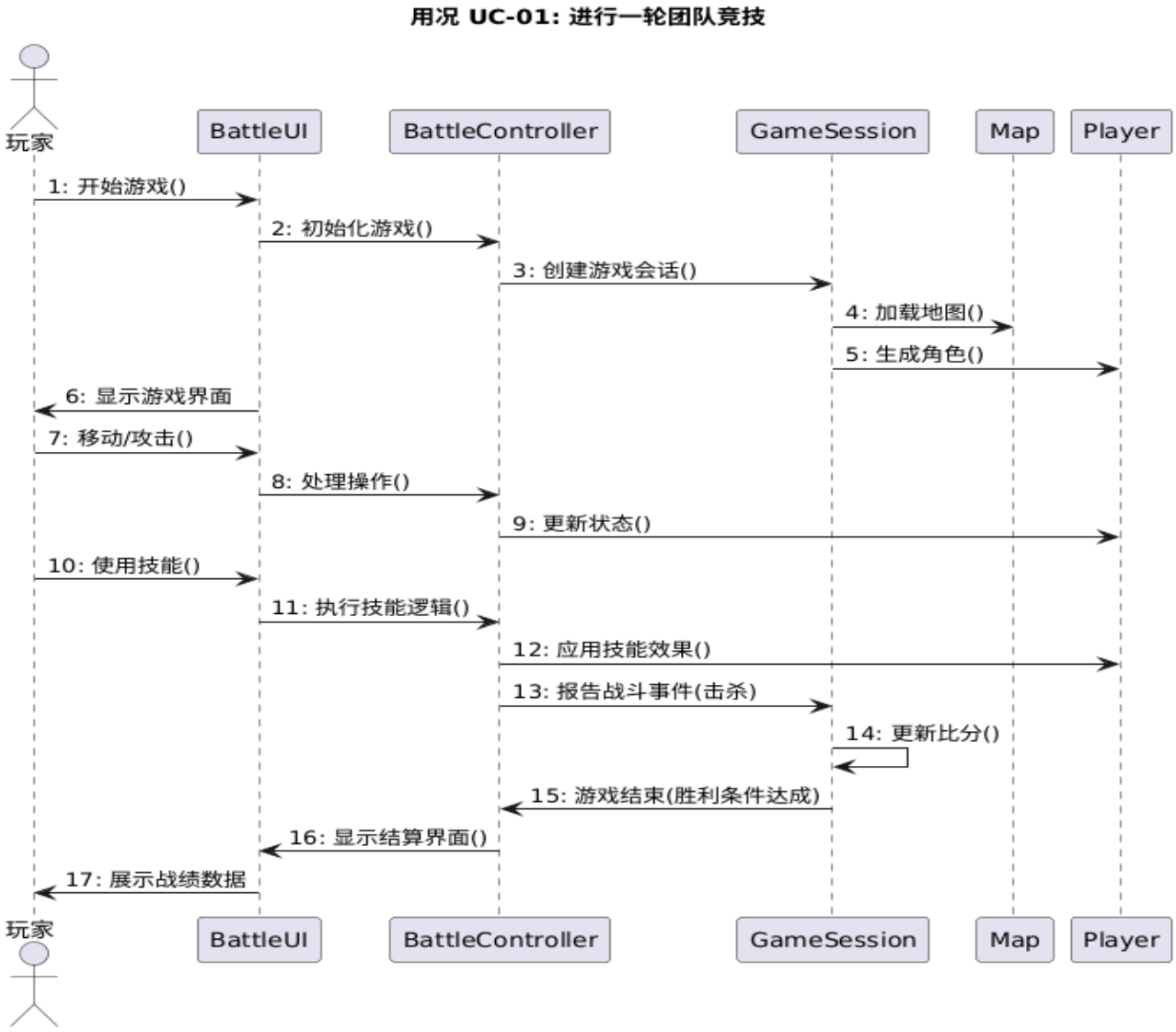
类名	类型	职责说明
CustomizationUI	边界类	提供：英雄选择界面、技能配置面板、外观预览 接收：玩家个性化配置选择 显示：3D 模型渲染和特效展示
HeroManager	控制类	验证：配置组合的合法性 管理：英雄数据加载和保存 处理：个性化设置的持久化
Hero	实体类	知晓：基础属性、可用技能、外观选项 能做：提供技能信息、验证配置兼容性
PlayerProfile	实体类	存储：默认出战英雄、技能搭配、解锁外观 管理：玩家个性化配置档案

4.1.4 用况UC-04：回顾战绩与成就

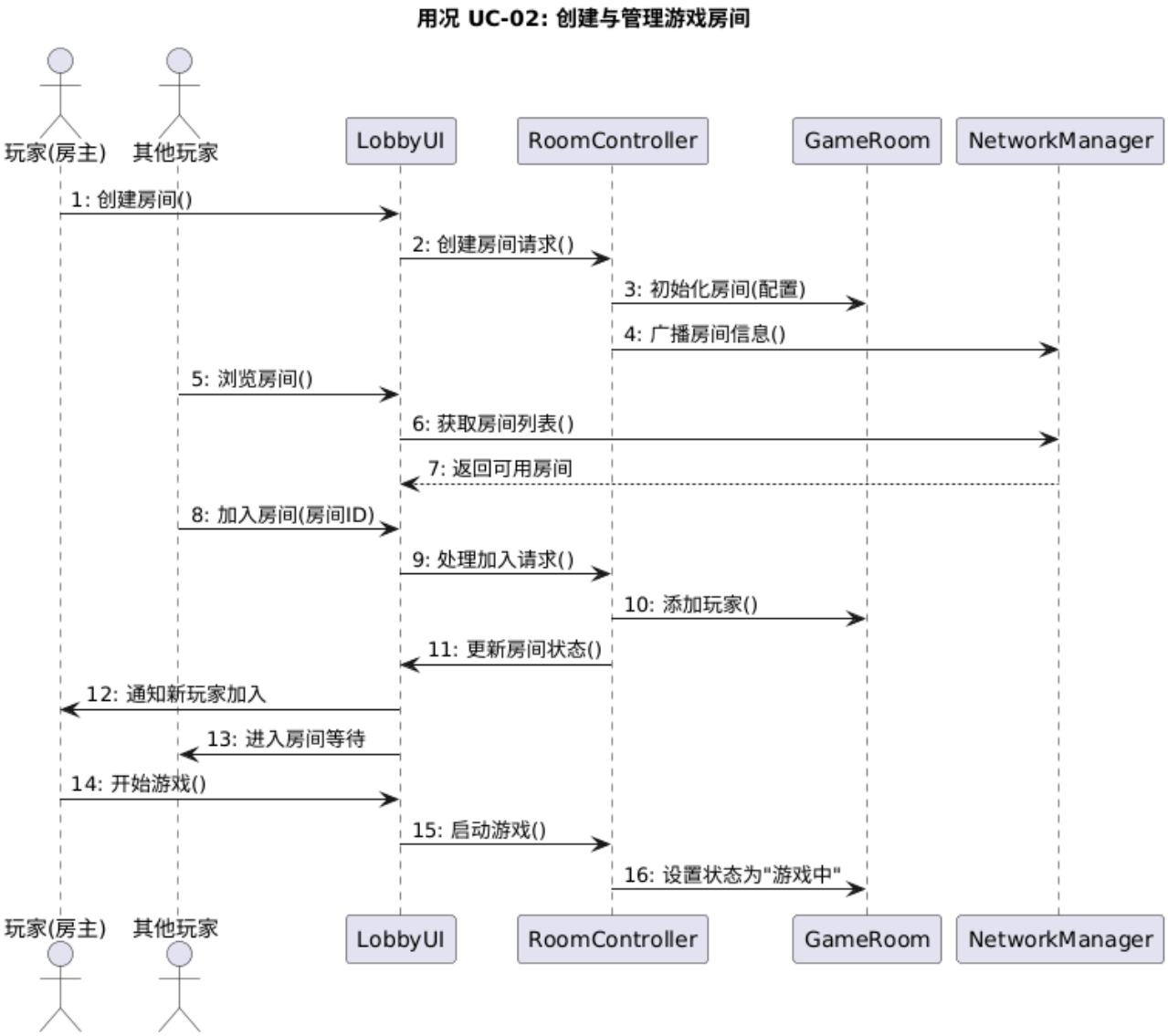
类名	类型	职责说明
StatsUI	边界类	显示：数据概览、战斗记录、成就列表 提供：详细战报查看界面

		支持：数据筛选和对比功能
StatsController	控制类	处理：数据查询和聚合计算 分析：玩家表现统计数据 追踪：成就进度和完成状态
MatchHistory	实体类	存储：对局时间、参与者、详细数据 提供：单场对局的完整复盘信息
Achievement	实体类	知晓：达成条件、当前进度、奖励信息 能做：检查进度状态、触发成就完成

4.2. 交互建模

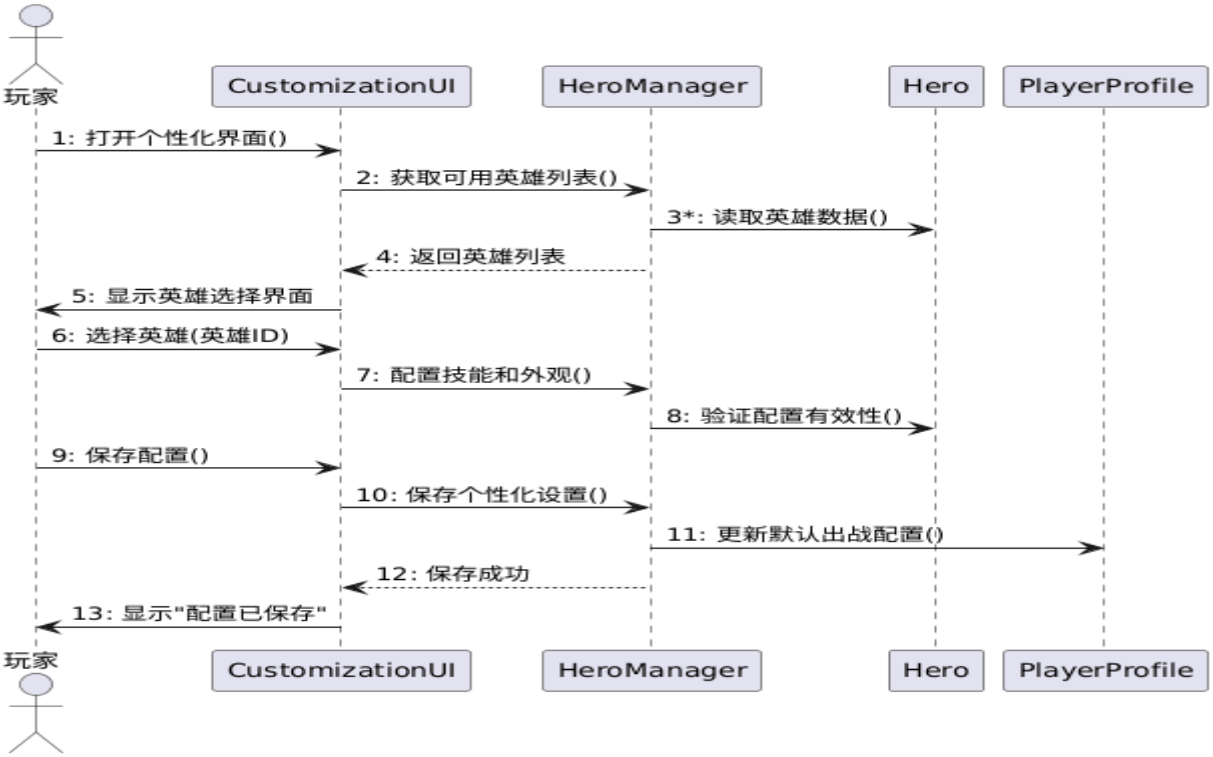


4.1.2 用况 **UC-02**：创建与管理游戏房间

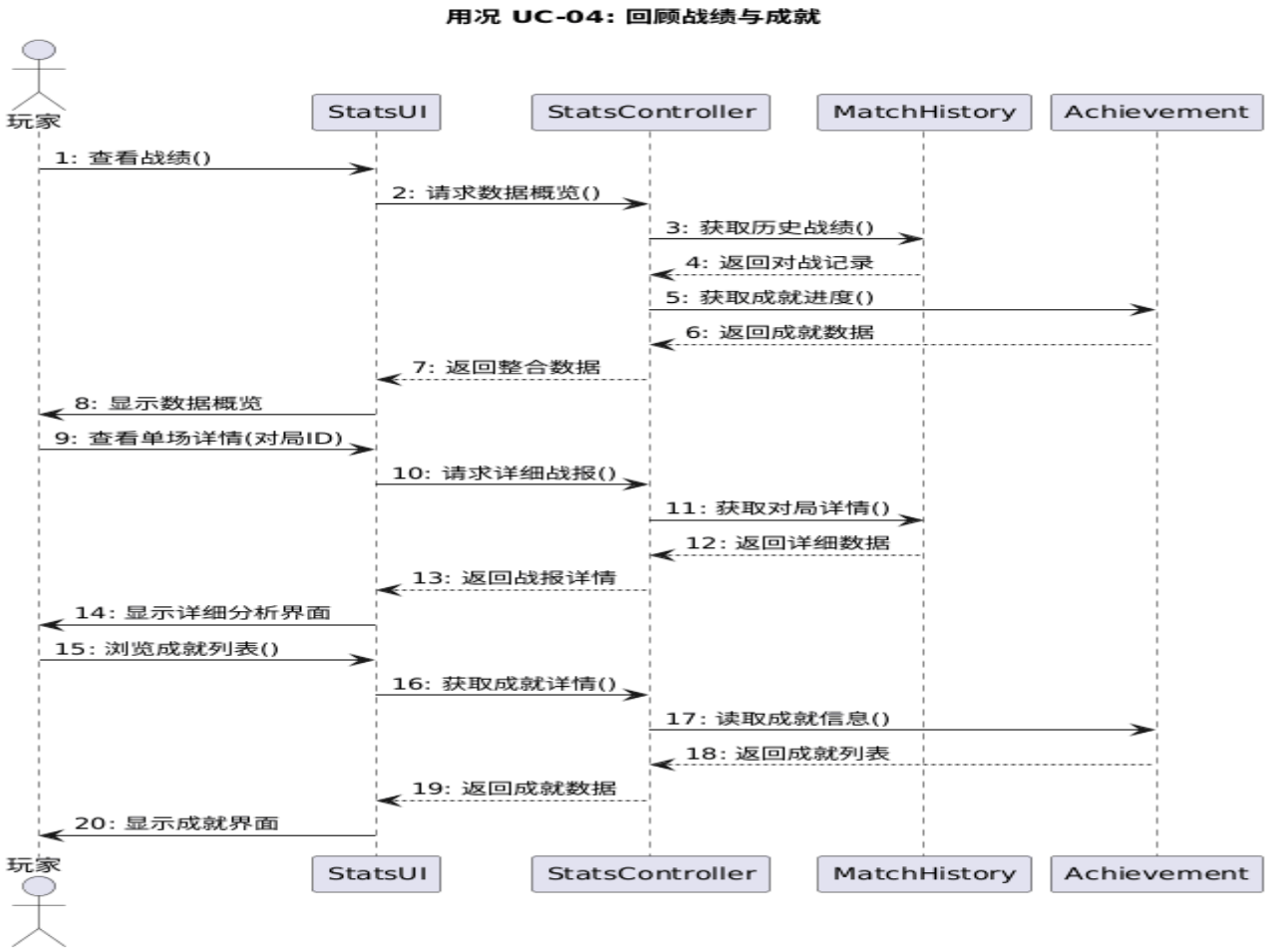


4.1.3 用况 **UC-03**：个性化游戏角色

用况 UC-03: 个性化游戏角色



4.1.4 用况UC-04：回顾战绩与成就



第5章 架构设计

第 **5** 章 架构设计

5.1 设计哲学与概述

本章旨在定义“PVP 对战游戏”的高层组织结构。我们融合了 **责任驱动设计（RDD）** 的思想，将系统视为一个由承担特定职责、并通过契约协作的**角色**所组成的社区。同时，我们

采用 **“4+1 视图”模型** 对这一角色社区进行多角度描述。此架构的核心目标是承载“公平竞技”、“流畅操作”等核心质量属性，并直接支持“团队竞技”（UC-01）、“房间管理”（UC-02）等关键用况。

5.2 架构的 4+1 视图

我们通过以下五个互补的视图来描述系统架构，每个视图聚焦于一组特定的关注点。

视图	系统描述与关键要素	对质量属性的贡献
用例视图	驱动架构设计的核心用况是：**UC-01（进行一轮团队竞技）**和**UC-02（创建与管理游戏房间）**。它们决定了系统必须支持低延迟同步和动态房间管理的关键行为。	识别出需**高性能（流畅战斗）**和**高可靠性（稳定房间）**的质量要求，直接驱动架构设计关注点。
逻辑视图	系统被组织为一系列子系统包。关键逻辑包包括：**GamePlayCore（游戏规则）**、**NetworkSession（会话管理）**、**UI（用户界面）**和**Data（本地数据）**。清晰的包结构实现了高内聚、低耦合。	提高了**可维护性**与**可扩展性**，核心业务逻辑独立于 UI 和网络代码，便于模块的独立更新与替换。
进程视图	系统运行时有**客户端进程**与**权威服务器进程**。在局域网模式下，主机同时运行这两个进程。进程内部分离渲染、游戏逻辑与网络线程，通过 UDP 套接字进行进程间通信。	将**权威逻辑集中于服务器进程**，保障了游戏的**公平性**；进程与线程的分离提高了系统的**可靠性**与**性能**（响应能力）。
开发视图	在 Unreal Engine 5 中体现为：**PVPGame（主游戏模块）**、**PVPGame.Target（客户端目标）**、**PVPGameServer.Target（服务器目标）**。资源按子系统组织在 `Content/` 目录下。	模块化划分支持**并行开发**，客户端与服务器目标分离确保了**部署的灵活性**，便于独立构建与测试。
物理视图	在局域网典型部署中：一台玩家主机同时运行**客户端和服务器进程**；其他玩家机器仅运行**客户端进程**。所有节点通过局域网交换机连接。	客户端-服务器的物理分配是实现**权威服务器架构**、保证**状态一致性**与**低延迟**的基础。

5.3 关键架构图表

1. 逻辑视图 **-** 子系统包图

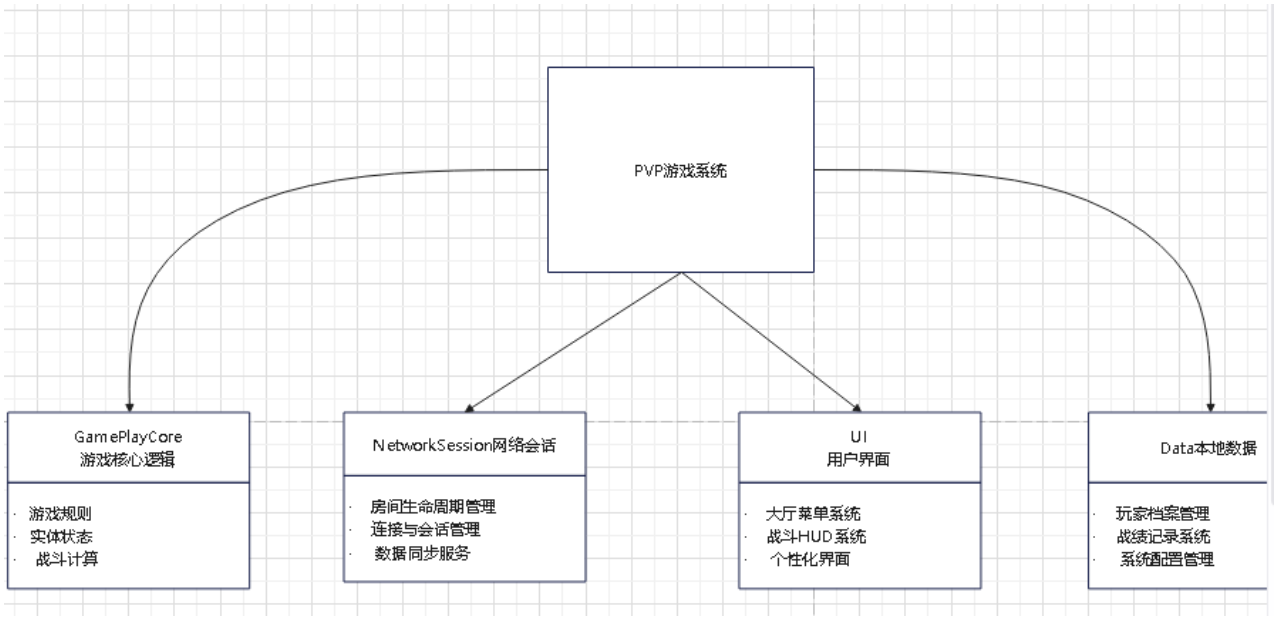


图 5.1 系统逻辑视图 - 子系统包图

2. 进程视图 **-** 运行时进程与线程图

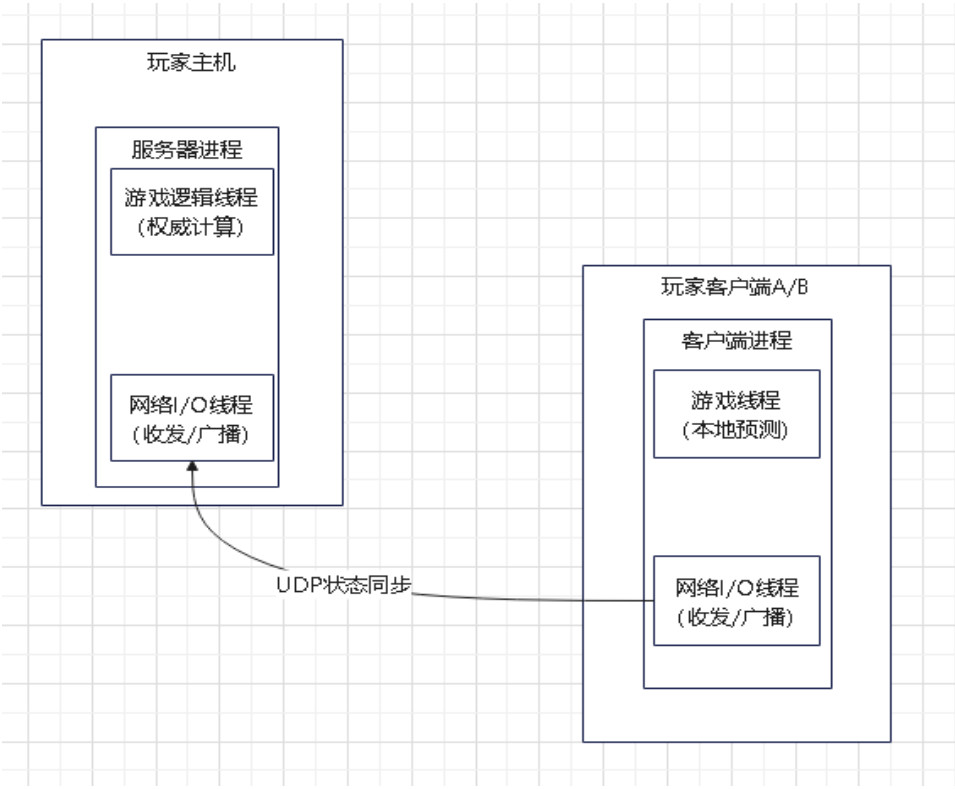


图 5.2 系统进程视图 - 运行时进程与线程图

3. 物理视图 **-** 系统部署图

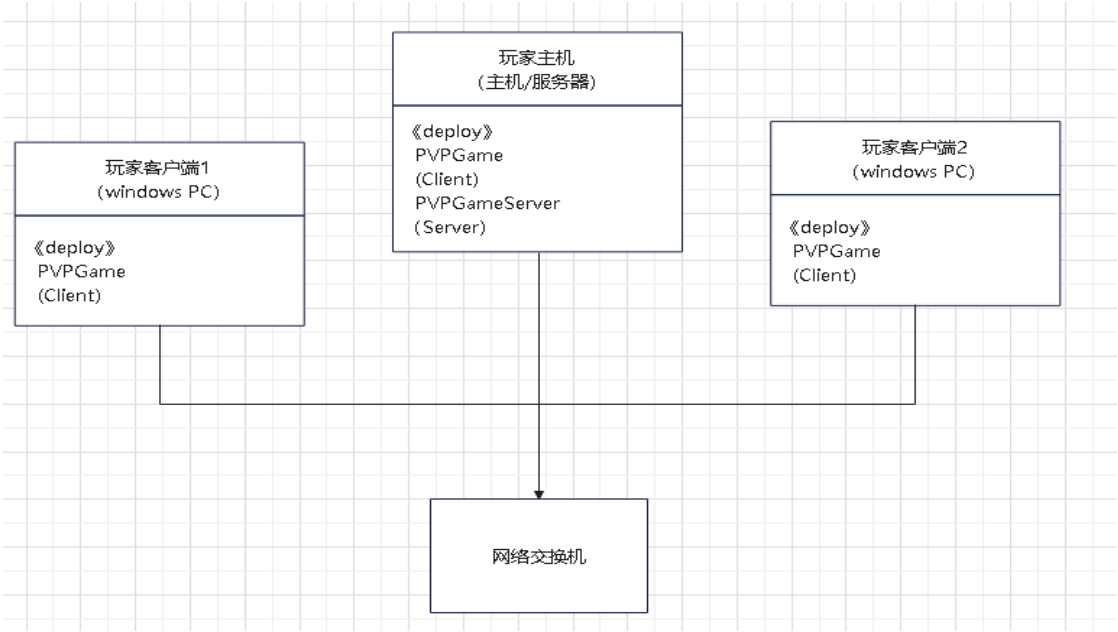


图 5.3 系统物理部署视图

5.4 架构风格、子系统与关键角色

架构风格

分层架构：采用**表示层-业务逻辑层-数据层**三层逻辑架构，确保职责分离。

客户端-服务器风格：服务器是游戏状态的“唯一真相源”，承担所有权威逻辑；客户端主要负责表现与输入。

1.子系统划分

网络通信子系统（`NetworkSession`）：处理局域网发现、连接管理、数据包收发。

游戏逻辑子系统（`GamePlayCore`）：实现核心战斗规则、实体行为与胜负判定。

用户界面子系统（`UI`）：管理所有屏幕的流程与交互。

数据管理子系统（`Data`）：负责玩家数据的本地持久化与缓存。

2.关键架构角色（**RDD** 视角）

权威仲裁者：由服务器进程实现，承担维护游戏唯一真理、执行规则的核心责任。

房间管家：由 `NetworkSession` 子系统实现，负责房间的生命周期管理。

玩家代言人：由客户端进程及 `PlayerController` 等实现，负责输入、预测与表现。

5.5 并发与处理器分配

固有并发性：多玩家同步（数据并行）与**客户端内部多线程**（渲染、逻辑、网络线程分离）。

处理器分配：在局域网主机上，**服务器逻辑进程**与**本地客户端进程**共享物理处理器资源，需通过操作系统调度确保服务器逻辑获得优先计算资源，以最小化网络延迟。

5.6 系统设计标准

数据管理标准：使用Unreal Engine `USaveGame` 系统进行本地数据持久化。

用户界面标准：遵循UE5 **UMG** 设计规范；战斗HUD需清晰呈现关键信息；操作符合PC游戏惯例。

构建指南：采用UE5标准的 `PascalCase` 类命名和 `camelCase` 变量命名；代码组织遵循引擎模块化结构。

5.7 架构决策总结

决策项	决策内容	依据与理由
核心架构风格	分层架构 + 权威服务器	满足**职责分离**（架构原则），保障**游戏公平性**（项目核心目标）。
网络模型	局域网，主机兼任服务器	契合目标用户场景（线下多人），简化初期部署，直接支持 **UC-02（创建与管理游戏房间）**。
数据持久化	纯本地存储(USaveGame)	技术方案简单可靠，与当前项目复杂度匹配。
架构描述方法	4+1视图模型	全面、多角度描述系统，符合软件架构描述的行业与课程规范。

第6章 详细设计

6.1 详细设计概述

详细设计在系统架构（第 5 章）的基础上，定义各个子系统内部的具体实现方案。本章将遵循以下设计原则：

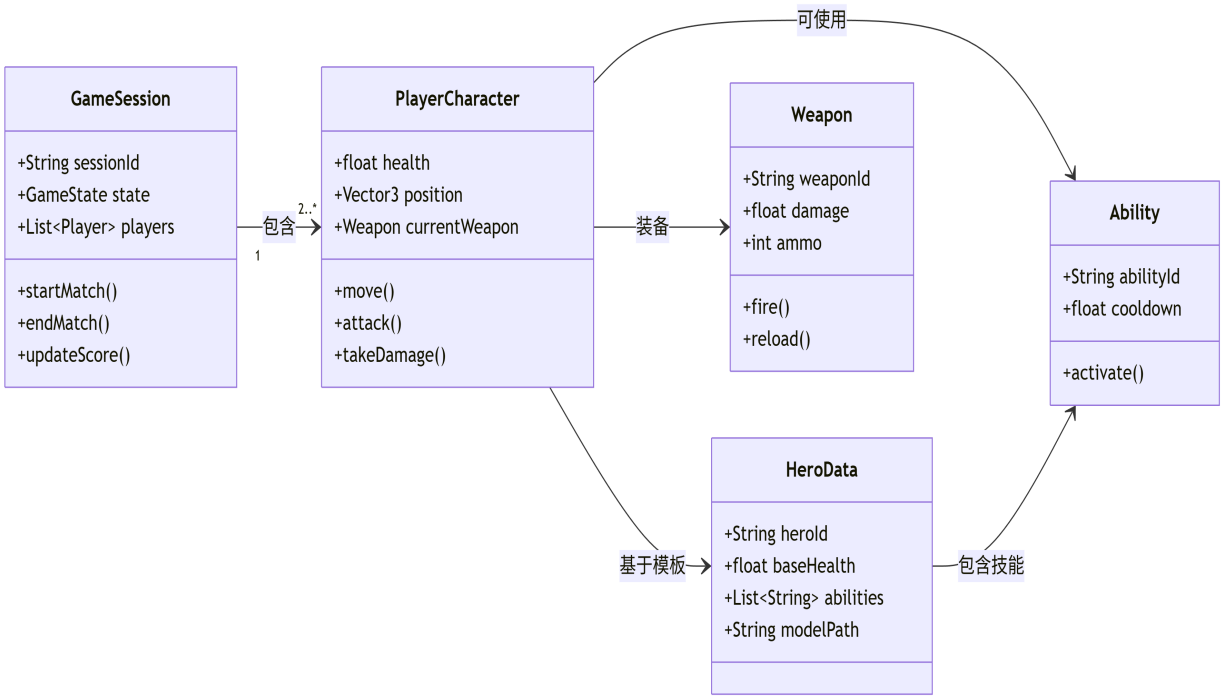
1. **职责驱动设计**：明确每个类的职责及其协作方式
2. **高内聚、低耦合**：确保类职责单一，减少不必要的依赖
3. **面向扩展设计**：为未来功能扩展预留空间
4. **实现独立性**：设计独立于特定实现细节，聚焦于逻辑设计

设计模型将使用类图、简要状态机和设计说明进行描述，确保与架构视图的一致性。

6.2 核心类设计

6.2.1 核心类关系图

以下类图展示了系统中最关键的类及其关系：



6.2.2 关键类说明

类名	类型	核心职责	关键属性
GameSession	实体类	管理单局游戏生命周期	sessionId, state, players
PlayerCharacter	实体类	玩家控制的游戏角色	health, position, currentWeapon
HeroData	数据类	英雄模板定义	heroId, baseHealth, abilities
Weapon	实体类	武器逻辑和状态	weaponId, damage, ammo
Ability	实体类	技能行为和冷却	abilityId, cooldown

6.3.1 网络同步机制设计

设计原则：

服务器权威原则：所有游戏关键逻辑由服务器计算和验证。

客户端预测原则：为提供流畅体验，客户端可预测并立即响应用户操作。

状态同步原则：服务器定期广播权威状态，客户端进行插值处理。

预测校正原则：当客户端预测与服务器结果不一致时，客户端需进行状态校正。

攻击命中流程设计：

阶段	客户端行为	服务器行为	设计目的与要点

1. 输入与预测	1. 立即播放攻击动画。 2. 进行本地的命中判断（如射线检测）。 3. 显示预测性的受击视觉效果。	-	**目的**：实现零延迟的输入反馈，保障操作手感。 **要点**：此阶段所有效果均为"预测"，非最终结果。
2. 请求验证	将此次攻击事件（包含攻击者、目标、时间戳等信息）发送给服务器进行权威验证。	接收客户端发来的攻击事件。	**目的**：将所有可能改变游戏状态的操作提交给唯一权威源（服务器）裁决。
3. 权威验证与计算	-	1. **验证**：检查此次攻击是否合法（如攻击者是否冷却、目标是否在射程和视野内）。 2. **计算**：若合法，根据公式计算伤害（考虑防御、暴击等）。 3. **应用**：更新目标的权威生命值，并判断是否死亡。	**目的**：确保游戏规则的严格执行与绝对公平，杜绝作弊可能。 **要点**：伤害计算等核心逻辑仅在服务器运行。
4. 结果广播	接收服务器发来的此次攻击的最终结果通知。	将攻击的最终结果（命中与否、伤害值、击杀事件）广播给所有相关的客户端。	**目的**：让所有玩家观看到一致的游戏世界状态。
5. 预测校正	对比服务器的结果与本地预测： - **若一致**：确认预测，无需额外操作。 - **若不一致（如未命中）**：回滚预测效果（如取消受击特效、调整角色状态）。	当验证失败时，需明确通知发起攻击的客户端。	**目的**：解决因网络延迟或预测错误导致的客户端与服务器状态不一致，保证最终状态的统一。

关键设计决策总结：

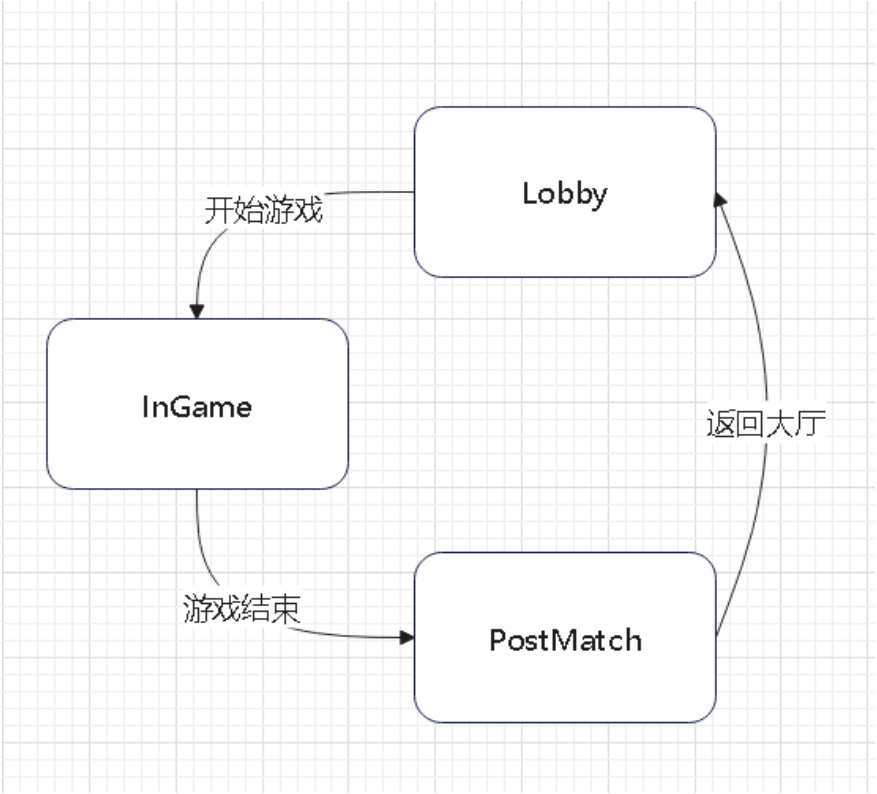
1. **逻辑位置**：伤害计算、胜负判定等核心逻辑**仅存在于服务器端**。
2. **通信模式**：客户端向服务器发送**操作意图**，服务器验证后广播**确定结果**。
3. **预测与校正**：客户端通过"预测-验证-校正"循环来兼顾响应速度与状态一致性。

6.3.2 状态同步策略

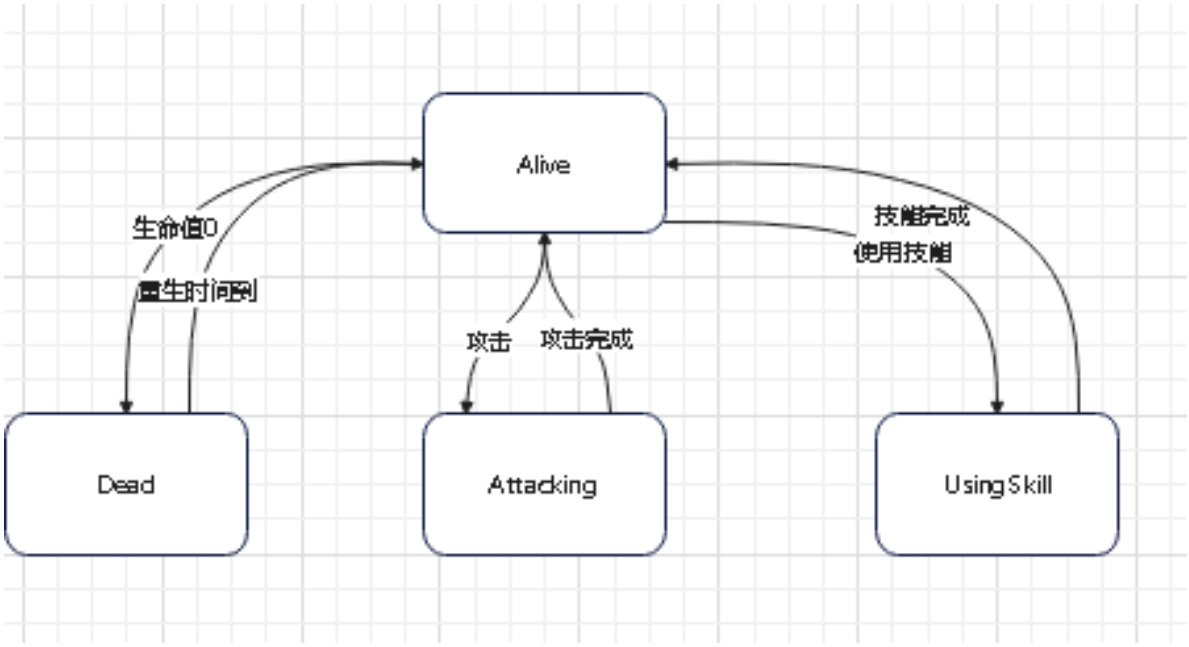
同步类型	同步内容	频率	设计策略
高频状态同步	角色位置、旋转、速度等。	高（如每秒 15-30 次）	采用不可靠协议（如 UDP）进行增量更新，客户端对收到的最新状态包进行插值平滑，以呈现流畅移动。
可靠事件通信	攻击、使用技能、获得道具、玩家死亡等离散事件。	事件发生时	采用可靠协议（如 TCP 或可靠 UDP）确保事件必达且有序，触发关键的游戏逻辑与表现。
低频数据同步	游戏模式分数、剩余时间、全局状态。	低（如每秒 1-2 次）	在状态同步包中附带或单独广播，供客户端更新 UI 显示。

6.4 状态机建模

6.4.1 GameSession 状态机（核心状态）



6.4.2 PlayerCharacter 状态机（战斗相关）



6.5 用户界面设计要点

6.5.1 界面布局原则

1. **信息分层**：重要信息（生命值、弹药）置于显眼位置；次要信息（小地图、技能图标）置于边缘
2. **视觉反馈**：关键事件必须有明显的视觉或听觉反馈
3. **一致性**：同类操作在不同界面保持一致的交互方式

6.5.2 关键界面设计

1. **战斗 HUD**：显示生命值、弹药、技能冷却、小地图和计分板
2. **角色选择界面**：提供英雄预览、技能说明和属性对比
3. **结算界面**：清晰展示本局数据、经验获取和奖励信息

6.6 数据存储设计

6.6.1 存储内容

1. **玩家配置**：按键设置、视频选项、音频偏好

2. **游戏进度**：解锁的英雄、皮肤、成就进度
3. **本地统计**：近期战绩、常用英雄数据

6.6.2 存储策略

1. **本地文件存储**：使用平台通用的文件格式（如 JSON）
2. **自动备份**：重要数据保存时自动创建备份
3. **版本兼容**：数据格式包含版本号，支持升级迁移

6.7 关键设计决策

决策点	选择方案	理由	风险缓解
网络模型	权威服务器	确保公平性，防止作弊	增加服务器负担，需优化性能
伤害计算	服务器计算	杜绝客户端篡改	网络延迟影响体验，需客户端预测
数据存储	本地存储	简化架构，无需服务器	数据无法跨设备，考虑未来扩展
角色系统	数据驱动	易于扩展和平衡	需要数据编辑工具支持

6.8 待验证假设

1. **网络延迟容忍度**：假设局域网延迟<50ms 可接受。
2. **同步频率**：假设 20Hz 同步频率能满足需求。
3. **预测准确性**：假设 80%以上的操作预测正确。
4. **客户端性能**：假设主流配置能稳定 60FPS 运行。

后记

正文内容，方正仿宋，小四，首行缩进。正文内容，方正仿宋，小四，首行缩进。正文内容，方正仿宋，小四，首行缩进。

参考文献

[1] YOUNG.RSS 是什么？[EB/OL]. http://jingpin.org/what-is-rss/.

[2] 杨博，彭博.RSS 提要分析与阅读器设计[R].成都：四川大学计算机学院，2007：42-43.

[3] 逸出络然.RSS 技术的原理[EB/OL].http://yclran.blog.163.com/blog/static/97945496200911103411155 8/.

[4] 佚名.Qt 是什么[EB/OL]. http://qt.nokia.com/title-cn.

[5] 佚名.Model/View Programming[EB/OL]. http://doc.trolltech.com/4.6/model-view-programming.html.

[6] [加拿大]Jasmin Blanchette[英]Mark Summerfield 著 闫锋欣,曾泉人,张志强译.

[7] C++ GUI Qt4 编程（第二版）[M].电子工业出版社：2008:182-206,291-305.

[8] 佚名.XML Processing[EB/OL]. http://doc.trolltech.com/4.6/xml-processing.html.

[9] Michael Blala James Rumbangh 著.UML 面向对象建模与设计（第 2 版）[M].北京：人民邮电出 版社,2006:136-235.

[10] 胡海静,王育平,等. XML 技术精粹[M]. 北京：机械工业出版社，2001:17-19.