深圳大学考试答题纸

(以论文、报告等形式考核专用)

二〇 二五 ～二〇 二 六 学年度第 一 学期

课程编号	1500250001	课序号		课程名称	面向对象系统分析与设计	主讲教师	孙智达	评分	
学　　号	2023150139	姓名	杨皓翔	专业年级	软件工程 大三				

教师评语:

题目: 虚幻引擎 5 下的双模式 ARPG 系统

注:

本组分工为:

杨皓翔：ppt、参与者文档

文仁杰：EA 工程、项目背景

梁智炜：用例清单文档、数据库文档

一　选题和需求定义

1.1 选题与系统范围

1.1.1 项目选题

BM 战斗系统 是基于 UE5 Gameplay Ability System (GAS) 构建的动作角色扮演游戏战斗框架，其核心目标是提供流畅、深度和多样化的战斗体验 。系统定位为游戏核心玩法模块，负责处理所有与战斗相关的逻辑，包括能力管理、战斗类型切换、属性计算、蓄力机制以及动画同步等，旨在支持玩家根据不同场景和敌人类型灵活切换战斗策略。

1.1.2 系统范围

战斗系统由**多个核心功能模块**与**多个辅助功能模块**构成，共同形成完整的战斗系统生态。

核心功能模块

模块名称	核心职责
能力系统模块	管理角色能力库；处理能力激活与执行；应用能力效果
战斗类型切换模块	支持三种战斗类型（立棍 / 劈棍 / 戳棍）的切换与状态管理
属性管理模块	维护角色属性体系（生命 / 战斗 / 特殊属性）；处理属性计算与更新
棍势蓄力模块	实现四级棍势蓄力机制；管理棍势值的积累与衰减
战斗能力执行模块	处理战斗能力执行逻辑；命中检测；伤害计算

辅助功能模块

模块名称	核心职责
动画通知模块	同步动画与能力执行；触发关键帧游戏事件
游戏性效果模块	管理能力产生的持续效果与瞬时效果
输入处理模块	处理玩家输入指令；映射到对应战斗能力
AI 战斗行为模块	实现 AI 敌人的战斗决策与行为执行

1.2 项目背景、同类项目与特色

1.2.1 项目背景

项目背景，近年 ARPG 市场持续增长，2026 年全球规模预计达 120 亿美元，玩家对战斗系统的流畅度与深度需求日益提升，而传统战斗系统存在明显短板，难以适配现代玩家需求。

技术上，UE5 GAS 需扩展优化，动作捕捉技术保障战斗流畅性，成熟的跨平台工具也降低了开发门槛；

市场层面，玩家渴求独特有策略的战斗体验，开发者需灵活框架以控制成本，跨平台则对战斗系统的性能与兼容性提出更高要求。

1.2.2 同类项目

1) 《堡垒之夜》（GAS）：武器能力全，战斗风格单一。
2) 《无主之地 3》：武器技能足，无多元战斗切换。
3) 《黑神话：悟空》：战斗优质，但为单人模式。
4) 《战神》：手感深度佳，战斗类型固定。

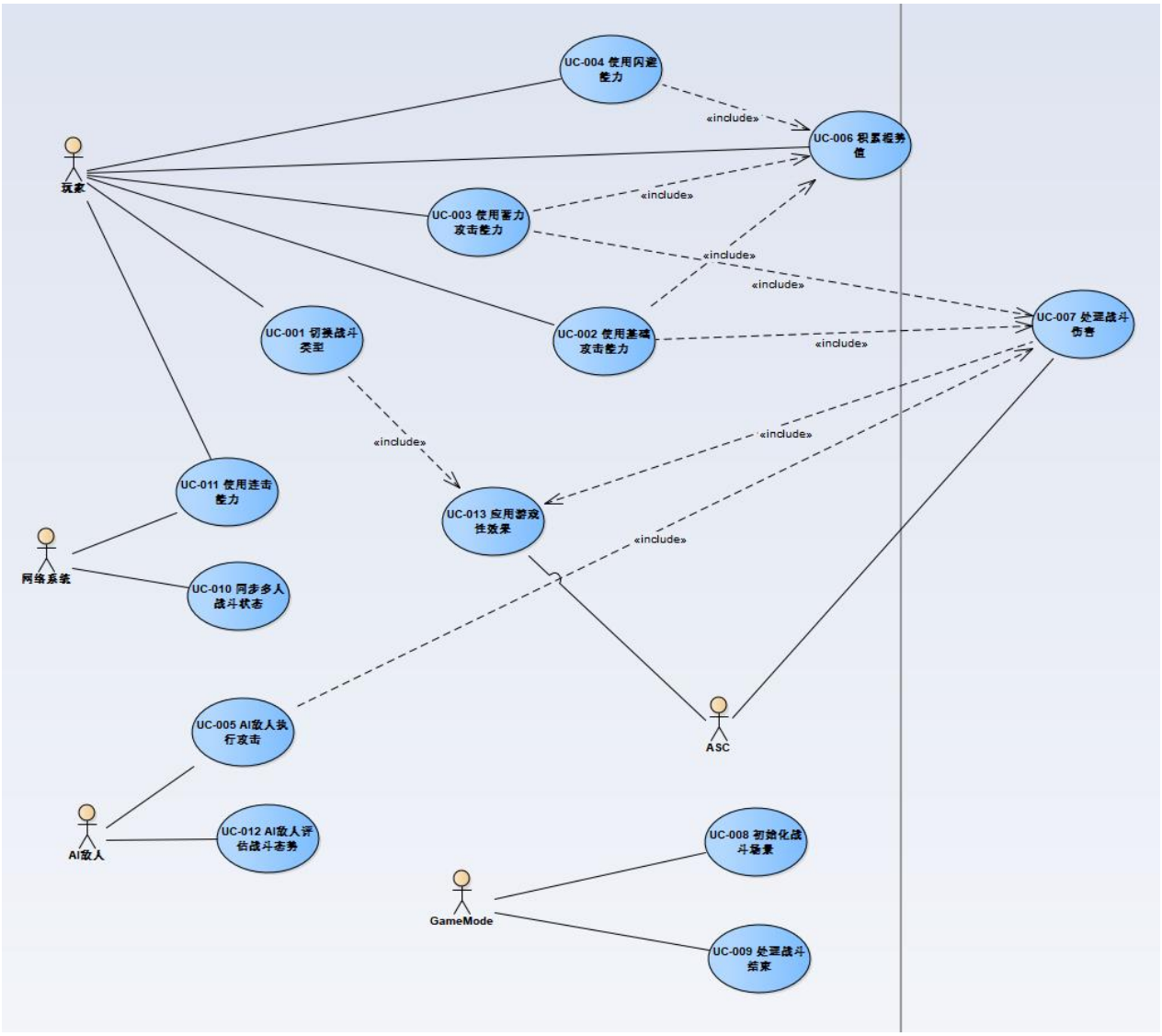
1.2.3 特色

本项目的特色与创新集中在黑神话战斗系统上加入多人对战支持。允许进行远程联机。

二　需求分析

2.1 迭代 1 用例模型

2.1.1 用例图



2.1.2 用例说明

用例 1：切换战斗类型

- **用例 ID**：UC-001
- **用例名称**：切换战斗类型
- **参与者**：玩家
- **前置条件**：
  1. 角色处于可战斗状态
  2. 角色未被控制（如眩晕、冰冻等状态）
  3. 角色已装备武器
- **后置条件**：
  1. 角色的战斗类型切换为目标类型
  2. 角色的能力库更新为对应战斗类型的能力
  3. 角色的战斗属性根据战斗类型调整
- **基本流程**：

1. 玩家输入战斗类型切换指令（按键 Z）
2. 系统检查角色是否满足切换条件
3. 系统切换角色的战斗类型（立棍 → 劈棍 → 戳棍 → 立棍...循环切换）
4. 系统更新角色的能力库
5. 系统应用对应战斗类型的 GameplayEffect，调整角色属性
6. 系统向玩家反馈切换成功（UI 视觉反馈）
7. 用例结束

- **扩展流程：**

若角色处于不可切换状态（如被控制）

1. 系统拒绝切换请求
2. 系统向玩家反馈切换失败（视觉/音效反馈）
3. 用例结束

用例 2：使用基础攻击能力

- **用例 ID：**UC-002
- **用例名称：**使用基础攻击能力
- **参与者：**玩家
- **前置条件：**
  1. 角色处于可战斗状态
  2. 角色已装备武器
  3. 角色拥有基础攻击能力
  4. 角色的体力值足够
- **后置条件：**
  1. 基础攻击能力被执行
  2. 若命中目标，目标受到伤害
  3. 角色的体力值减少
  4. 角色的棍势值增加
- **基本流程：**
  1. 玩家输入基础攻击指令（鼠标左键）
  2. 系统检查角色是否满足攻击条件
  3. 系统激活基础攻击能力
  4. 系统消耗相应的体力值
  5. 系统播放基础攻击动画
  6. 系统在动画关键帧触发命中检测
  7. 若命中目标，系统计算伤害值
  8. 系统应用伤害到目标角色
  9. 系统增加角色的棍势值
  10. 用例结束
- **扩展流程：**

若角色不满足攻击条件：

1. 系统拒绝攻击请求
2. 系统向玩家反馈攻击失败（视觉反馈）
3. 用例结束

若攻击未命中目标：

1. 跳过步骤 8
2. 系统增加少量棍势值（命中时的 50%）
3. 用例结束

用例 3：使用蓄力攻击能力

- **用例 ID**：UC-003
- **用例名称**：使用蓄力攻击能力
- **参与者**：玩家
- **前置条件**：
  1. 角色处于可战斗状态
  2. 角色已装备武器
  3. 角色拥有蓄力攻击能力
  4. 角色的棍势值达到最低要求
- **后置条件**：
  1. 蓄力攻击能力被执行
  2. 若命中目标，目标受到伤害（考虑蓄力等级加成）
  3. 角色的棍势值消耗
- **基本流程**：
  1. 玩家输入蓄力攻击指令（长按鼠标 Y）
  2. 系统检查角色是否满足蓄力条件
  3. 系统激活蓄力攻击能力
  4. 系统进入蓄力状态，播放蓄力动画
  5. 系统根据蓄力时间增加棍势等级（最多 4 级）
  6. 玩家释放按键，系统执行蓄力攻击
  7. 系统播放蓄力攻击动画
  8. 系统在动画关键帧触发命中检测
  9. 若命中目标，系统计算伤害值
  10. 系统应用伤害到目标角色
  11. 系统消耗角色的棍势值
  12. 用例结束
- **扩展流程**：

若角色不满足蓄力条件（如棍势值不足）：

1. 系统拒绝蓄力请求
2. 系统向玩家反馈蓄力失败（视觉反馈）
3. 用例结束

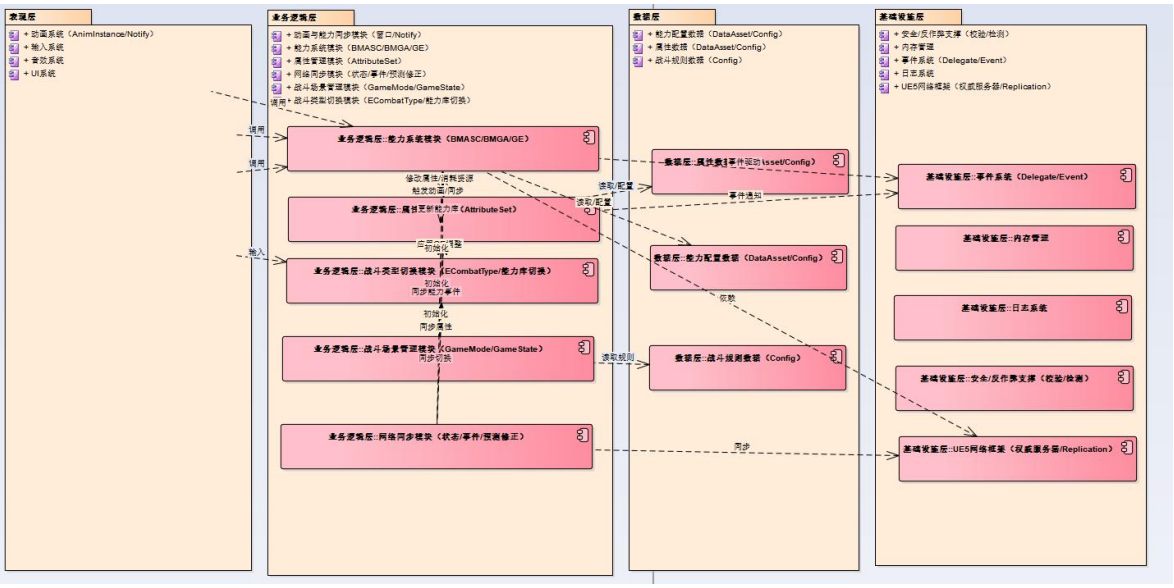
若攻击未命中目标：

1. 跳过步骤 10
2. 系统消耗少量棍势值（命中时的 30%）
3. 用例结束

更多用例见“用例清单文档”

2.2 系统架构

2.2.1 系统架构图



2.2.2 系统架构解释

分层结构（4 层）：

1. 表现层：动画系统、UI 系统、音效系统、输入系统——负责输入采集与画面/音频反馈。
2. 业务逻辑层：能力系统、属性管理、战斗类型切换、动画与能力同步、战斗场景管理、网络同步——负责战斗规则与核心运行逻辑。
3. 数据层：属性数据、能力配置数据、战斗规则数据——以 DataAsset/Config 为主，提供数据驱动的配置来源。
4. 基础设施层：UE5 网络框架、事件系统、内存管理、日志、安全/反作弊支撑——提供网络、事件、运行支撑与安全能力。

关键依赖链路：

1. 表现层通过输入/调用触发业务逻辑（例如输入系统 → 战斗类型切换；动画/UI → 能力系统）。
2. 战斗类型切换模块会更新能力库（依赖能力系统）并通过 GameplayEffect 调整属性（依赖属性模块）。
3. 能力系统在执行能力时会消耗/修改属性（依赖属性模块），并触发动画与同步（依赖动画与能力同步模块）；同时动画通知会回调能力逻辑（形成一条“同步闭环”）。
4. 业务逻辑层从数据层读取配置（能力/属性/规则），并依赖基础设施层（网络框架、事件等）。
5. 网络同步模块把战斗类型、能力事件、属性变化同步到多人环境（依赖 UE5 网络框架，并关联到切换/能力/属性模块）。
6. 战斗场景管理模块（GameMode/GameState）负责初始化战斗所需模块与规则（初始化切换、能力、属性，并读取战斗规则数据）。

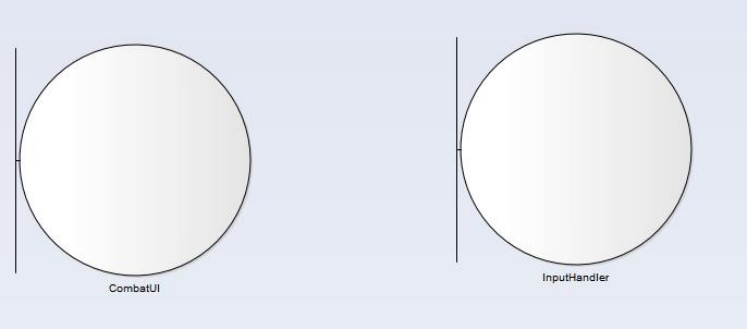
2.3 完善用例文档

见用例清单文档

2.4 识别分析类

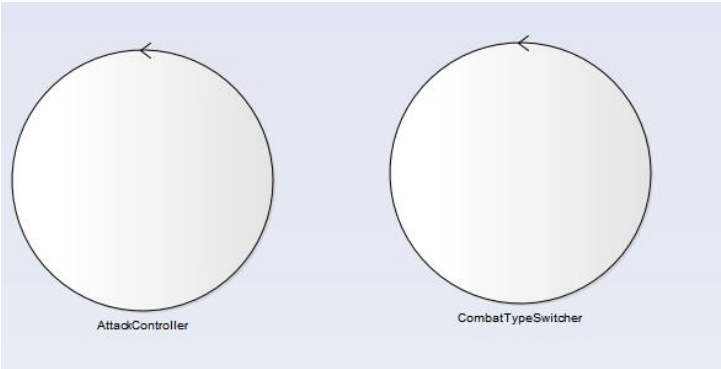
2.4.1 Boundary（边界类：与外部交互/界面/输入）

1. InputHandler：输入采集与输入状态管理；向控制层发出操作请求
2. CombatUI：战斗界面显示（血条/体力/棍势/战斗类型指示等）



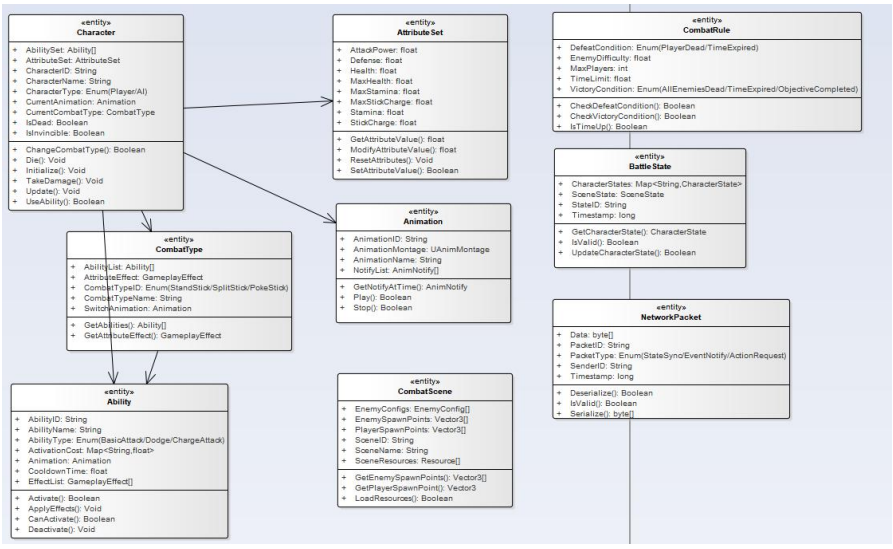
2.4.2 Control（控制类：用例流程编排/业务控制）

1. CombatTypeSwitcher：切换战斗类型流程控制（校验条件、应用效果、驱动角色变化）
2. AttackController：攻击流程控制（处理攻击输入、触发攻击、命中回调、驱动更新）

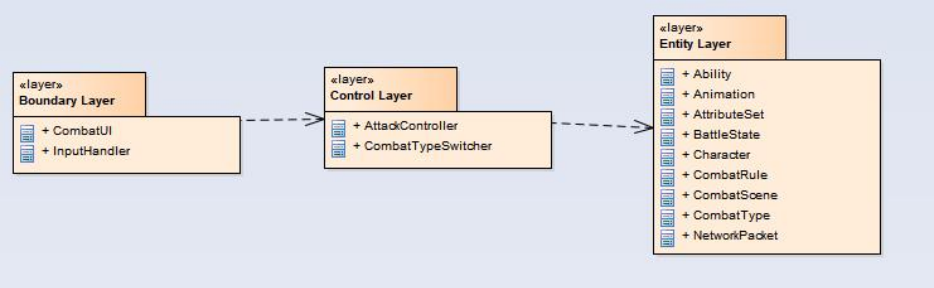


2.4.3 Entity（实体类：业务核心信息与规则承载）

1. Character：角色实体（属性、能力、动画、状态）
2. CombatType：战斗类型（可用能力、切换效果、切换动画）
3. Ability：能力/技能（消耗、冷却、动画、效果列表）
4. AttributeSet：属性集（血量/体力/棍势/攻防等）
5. Animation：动画实体（蒙太奇资源、通知）
6. CombatScene：战斗场景（资源、出生点、敌人配置）
7. CombatRule：战斗规则（胜负条件、时间限制等）
8. BattleState：战斗状态快照（角色状态映射、场景状态、时间戳）
9. NetworkPacket：网络包（同步/事件/请求的数据承载）



2.4.4 layer 之间关系



2.5 用例文档完善

见“用例清单文档”

三　系统设计

3.1 数据库设计

3.1.1 类（属性）与表（字段）之间的对应关系

分析模型实体类	对应数据表	主要存储内容
Character	characters	角色基础信息、状态字段、当前战斗类型/动画等引用
AttributeSet	attribute_sets	角色属性集合（生命、体力、棍势、攻防等）
CombatType	combat_types	战斗类型定义（名称、属性效果、切换动画等）
Ability	abilities	能力定义（类型、冷却、成本、效果、动画引用）
Animation	animations	动画资源信息
AnimationNotify	animation_notifies	动画通知定义（时间点、类型、附加数据）
CombatScene	combat_scenes	场景资源与出生点、敌人配置等
CombatRule	combat_rules	胜负条件、时间限制、最大玩家数、难度系数等
BattleState	battle_states	战斗状态快照（角色状态映射、场景状态、时间戳）
NetworkPacket	network_packets	网络包历史（类型、发送者、时间戳、内容）

3.1.2 类关系与表关系之间的对应

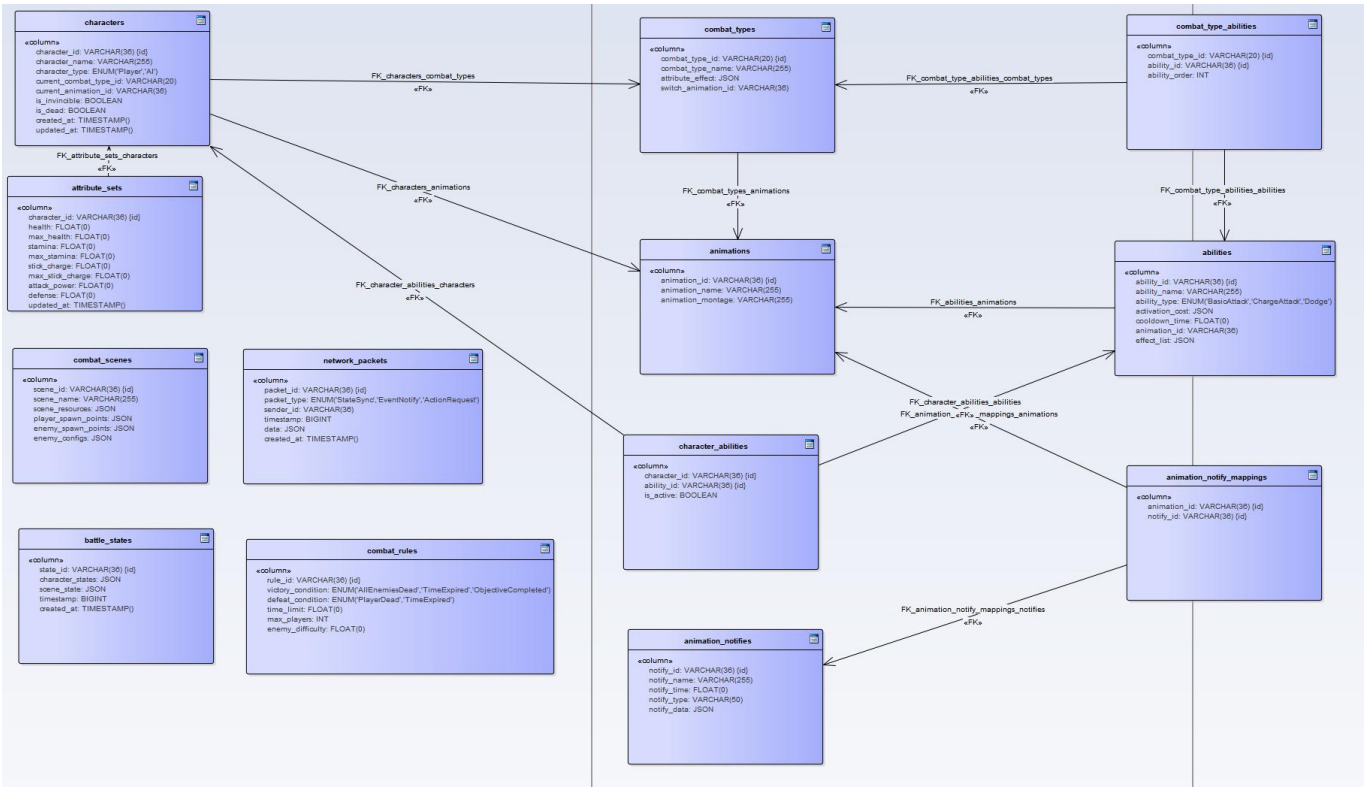
类关系类型	表关系实现方式	示例（本系统）
一对一(1:1)	共享主键或外键 + UNIQUE	Characters <-> attribute_sets（共享主键 character_id）
一对多(1:N)	N 端持有外键	Animations -> animation_notify_mappings
多对多(M:N)	关联表（中间表）	Characters <-> abilities（character_abilities）
引用关系	外键引用	characters.current_combat_type_id -> combat_types
记录时间序列	业务字段关联或外键扩展	battle_states 与 network_packets 按需要增加关联字段

3.1.3 关联类（Association Class）的处理

当类之间存在多对多关系，并且关系本身需要携带属性（例如排序、激活状态），采用**关联表**建模：

1. character_abilities
   a) 解决：角色可拥有多个能力，能力也可被多个角色使用
   b) 关系属性示例：is_active（是否激活）
2. combat_type_abilities
   a) 解决：战斗类型与能力的多对多
   b) 关系属性示例：ability_order（该战斗类型下能力顺序）
3. animation_notify_mappings
   a) 解决：动画与通知之间的映射关系
   b) 若 notify 可复用，则更适合用映射表；否则也可改为 notify 表直接引用 animation_id（1:N）

3.2 数据库图



详情见数据库文档。

四　总结与展望

1. 总结

本次课程设计围绕基于 UE5 Gameplay Ability System（GAS）的 BM 战斗系统完成了从需求到设计的完整建模过程。系统以“能力管理—战斗类型切换—属性计算—棍势蓄力—动画同步—多人联机同步”为核心主线，明确了战斗系统在游戏中的定位与系统边界，并形成了较为完整的模块化方案。

在**需求分析**阶段，采用“用例驱动”方法建立迭代 1 用例模型，围绕“切换战斗类型、基础攻击、蓄力攻击、闪避、AI 敌人攻击”等核心用例，对前置条件、基本流程、扩展流程进行了规约描述，保证需求可追踪、可验证，并为后续设计提供依据。

在**架构与设计**阶段，建立了“四层分层架构”，并梳理关键依赖链路：输入驱动业务逻辑、能力执行与属性消耗联动、动画通知触发关键帧逻辑回调形成同步闭环、网络同步模块将能力事件与属性变化广播到多人环境，从而实现关注点分离、可维护与可扩展。

在**面向对象分析与建模**方面，基于 Boundary-Control-Entity 思路识别分析类：以 InputHandler、CombatUI 作为边界类；以 CombatTypeSwitcher、AttackController 作为控制类；以 Character、Ability、CombatType、AttributeSet 等为实体类，形成清晰的领域模型分工，为系统实现提供结构化蓝图。

在**数据建模**方面，将核心实体类映射到关系型表结构，明确 1:1、1:N、M:N 的实现方式，并使用关联表处理“角色-能力”“战斗类型-能力”等多对多关系，保证数据层在后续扩展（新能力、新战斗类型、更多动画通知）时具备良好的可演进性。

2. 反思

(1) 战斗逻辑复杂、模块耦合风险高

问题：战斗系统涉及输入、动画、命中、伤害、效果、属性、网络同步等多个子域，若直接“事件堆叠”容易形成强耦合。

解决思路：采用分层架构与控制类编排流程，将“表现/动画”“能力执行”“属性与效果结算”分离，并通过动画通知机制将关键帧事件回调到能力逻辑，避免把时间控制写死在业务代码里。

(2) 蓄力与棍势机制需要可扩展建模

问题：蓄力等级（最多四级）与棍势积累/衰减属于典型的“状态随时间变化”的业务，容易在实现时出现状态不一致。

解决思路：在需求侧先以用例规约明确“长按—蓄力—释放—结算—消耗”的阶段性流程，并在设计侧将棍势与蓄力等级统一收敛到 AttributeSet/属性模块管理，保证逻辑一致性。

(3) 多人联机同步带来一致性挑战

问题：多人环境下需要同步战斗类型、能力触发、属性变化等关键事件，若缺乏统一的数据承载会导致状态漂移。

解决思路：将同步事件抽象为网络包/状态快照等数据承载（NetworkPacket、BattleState），并在架构层明确网络同步模块依赖 UE5 网络框架，围绕“关键事件同步”的思路设计扩展路径。

3. 展望

(1) 补全动态行为模型与一致性验证

后续可进一步完善顺序图/状态机图，覆盖“连击能力、完美闪避、AI 行为评估”等更复杂流程，并对关键用例做一致性检查（用例规约 ↔ 动态模型 ↔ 类职责）。

(2) 增强 GAS 能力复用与数据驱动

将能力、效果、属性曲线进一步 DataAsset/Config 化，实现“新增能力无需改动核心代码”的数据驱动扩展；并补充受击硬直等高级战斗机制。

(3) 完善多人网络策略

围绕“服务端权威结算 + 客户端预测/回滚”的方向设计同步策略，明确伤害结算、状态变更的权威侧，并结合基础设施层扩展日志、安全与反作弊策略。