**北京邮电大学课程设计报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **课程设计**  **名称** | **面向对象程序设计实践（C++）** | **学 院** | **计算机学院** | **指导教师** | **邹华** |
| **学生姓名** | **索政铎** | **班 级** | **2018211305** | **学 号** | **2018211279** |
| **课**  **程**  **设**  **计**  **内**  **容** | 简要介绍课程设计的主要内容，包括课程设计教学目的、基本内容、实验方法和团队分工等  **主要内容：用面向对象的设计方法来设计一款平台类对战游戏。**  **教学目的：学习使用C++、借助面向对象的设计模式进行开发。**  **基本内容：在平台类对战游戏中添加宠物小精灵、用户注册与平台登录、游戏对战等模块。**  **实验方法：在Windows平台上，借助Visual Studio进行服务端开发，借助Qt Creator进行客户端开发。** | | | | |
| **学生**  **课程设计**  **报告**  （附页） | 见程序代码及课程设计报告。 | | | | |
| **课**  **程**  **设**  **计**  **成**  **绩**  **评**  **定** | 遵照实践教学大纲并根据以下四方面综合评定成绩：  1、课程设计目的任务明确，选题符合教学要求，份量及难易程度  2、团队分工是否恰当与合理  3、综合运用所学知识，提高分析问题、解决问题及实践动手能力的效果  4、是否认真、独立完成属于自己的课程设计内容，课程设计报告是否思路清晰、文字通顺、书写规范  **评语**:  **成绩**:  指导教师签名：  年 月 日 | | | | |

1. **设计任务的描述**

用面向对象的设计方法来设计一款平台类对战游戏。

内容包括宠物小精灵的加入、用户注册与平台登录、游戏对战的设计。

1. **功能需求说明及分析**
2. **宠物小精灵的加入**

* 设计宠物小精灵的类，为简化游戏设计，精灵的属性包括种类（力量型：高攻击；肉盾型：高生命值；防御型：高防御；敏捷型：低攻击间隔，共四种）、名字、等级、经验值、攻击力、防御力、生命值、攻击间隔等（以上属性必须，其他属性可自行添加））（基本要求：本游戏 中 只有上面的 4 种类型。 进一步要求：上述 4 种类型可以进一步深入划分，比如力量型又可以再细分为：沙瓦朗、火爆猴、腕力等）。
* 每个精灵初始等级为 1 ，满级 15 级，每当精灵升级的时候，宠物对应的属性值会有少量增加（主属性增加量相对较多）。
* 每个精灵有自己独特的攻击方式，如闪电攻击、火焰攻击等等，请设计一个精灵的基类，并将精灵的攻击方法设为虚方法以方便子类重写。
* 请写一个测试程序对设计的精灵类的相关属性和方法（包括攻击函数，升级函数等）进行测试。
* 题目主要考察点：类的继承，对象数据成员设计，成员函数设计。

1. **用户注册与平台登录**

* 每个用户需要注册一个账号，用户名全局唯一，不能有任何两个用户名相同，要考虑注册失败的场景时的反馈。
* 实现注册、登录、登出功能， 均采用 C/S 模式，客户端和服务端用 socket 进行通信，服务端保存所有用户的信息。
* 每个用户拥有：用户名、拥有的精灵，两个属性。 用户注册成功时，系统自动随机分发三个 1 级精灵给用户
* 用户可以查看所有成功注册用户拥有的精灵，也可以查看所有当前在线的用户。
* 题目主要考察点： socket 通信，交互场景反馈，用户信息存储方式界面交互，其它合理的新颖设计。

1. **游戏对战的设计**

* 已经登录的在线用户可以和服务器进行虚拟决斗，决斗分两种：升级赛和决斗赛，两种比赛都能增长精灵宠物经验值。服务器上有一个虚拟精灵的列表，用户可以挑选其中任意一个进行比赛（升级赛或者决斗赛）。另外决斗赛中用户胜出可以直接获得该战胜的精灵，失败则系统从用户的精灵中随机选三个（不够三个精灵的情况就选择他所有的精灵），然后由用户选一个送出（不够三个精灵的情况就选择他所有的精灵）。
* 用户如果没有精灵（比如总是失败，已经全部送出去），则系统会随机放给他一个初级精灵。
* 请让你的系统自动模拟每场比赛的每次出招。另外，为了增加不确定性，可以加入概率闪避攻击和暴击伤害机制。
* 用户增加新功能，可以查看某个用户的胜率
* 用户增加新属性，为宠物个数徽章（金银铜）和高级宠物徽章（金银铜），分别根据拥有的宠物个数的多少和拥有高级宠物（15 级）个数的多少颁发。
* 题目主要考察点：客户端与服务器数据交互（可采用多进程或异步通信或其他方法均可），并发请求处理，类的方法设计，伤害计算方法设计，界面交互，其它合理的新颖设计。

1. **总体方案设计说明**

总体解决方案分为服务端与客户端两部分，采用UDP进行通信（服务器端口62100）。服务端负责所有的数据存储及处理，客户端负责发送用户操作及获取服务端信息。

服务端根据题目要求共抽象出服务器、验证、用户、精灵、战斗等类；客户端采用Qt实现图形化界面开发。

1. **数据字典**

* **服务端**

|  |  |  |
| --- | --- | --- |
| 服务器(Server) | static Server\* getInstance() | 获取单件 |
| int assign(char\*, int, SOCKET&, SOCKADDR&) | 从缓存中读取客户端发送的指定大小的内容，并分配跳转至其他函数 |
| Fairy\* getEnemybyNum(const int) | 返回指定序号的服务器端精灵 |
| list<Fairy\*> listEnemy | 保存服务器端精灵的链表 |
| int buildInfoEnemy(char\*) | 构造服务器端精灵的信息列表至指定内存中 |
| int informReport(char\*, const char\*) | 构造字符串至指定内存中 |
| static Server \_instance | 单件实例 |
| int enemyCntDefault | 服务器端精灵的初始个数 |
| 验证(Auth) | static Auth\* getInstance() | 获取单件 |
| int regist(const string , const string, const char\*) | 处理注册 |
| int login(const string, const string) | 处理登录 |
| int logout(const string, const string) | 处理注销，维护在线列表 |
| int verify(const string, const string) | 验证用户名及密码，维护在线列表 |
| list<User\*> onlineList | 用户列表 |
| int getOnlinePlayer(char\*) | 构造在线玩家信息至指定内存中 |
| struct infoOnlinePlayer | 玩家信息格式 |
| map<string, string> registMap; | 由用户名映射至密码进行验证 |
| map<string, User\*> savedataMap | 由用户名映射至玩家信息位置 |
| int saveMap() | 存储服务端数据至本地文件 |
| int loadMap(); | 从本地文件读取服务端数据 |
| 用户(User) | void giveOut(const int) | 送出指定序号的精灵 |
| Fairy\* getFairybyNum(const int) | 返回指定序号的精灵 |
| int buildInfoFairy(char\*) | 构造精灵信息至指定内存中 |
| int buildInfoBadage(char\*) | 构造徽章信息至指定内存中 |
| int buildInfoAttack(char\*, const int) | 构造指定序号精灵的攻击名称至指定内存中 |
| void checkNoFairy() | 检查是否失去所有精灵 |
| void addGiveOut() | 添加待送出精灵 |
| string getName() | 获取用户名称 |
| void healFairy(); | 治疗该玩家所有精灵 |
| void getBadage() | 获取徽章 |
| time\_t timeLastOpt | 上次用户进行操作的时间 |
| list<Fairy\*> listFairy | 精灵列表 |
| list<Fairy\*> listGiveOut | 待送出精灵列表 |
| vector<string> vecBadage | 徽章列表 |
| int timesBattleTol | 总战斗次数 |
| int timesBattleSuc | 总战斗获胜次数 |
| string nameCRC | CRC32方式转换的用户名 |
| string name | 用户名称 |
| struct fairyInfo | 精灵信息结构体 |
| const int numInitialFairy | 玩家初始精灵数 |
| 精灵(Fairy) | int getExp() | 获取经验值 |
| int getLevel() | 获取等级 |
| virtual int attackOtr(Fairy\* const) | 精灵攻击方法 |
| void addExp(int) | 增加经验值 |
| int getHurt(float) | 受到伤害 |
| void getHeal() | 治疗 |
| string name | 精灵名称 |
| string attackName | 精灵攻击名称 |
| int exp | 经验值 |
| int probMiss | 闪避率 |
| int probCritic | 暴击率 |
| float nowHP | 当前生命值 |
| float valueATK | 攻击力 |
| float valueDEF | 防御力 |
| float valueHP | 生命力 |
| float valueAGL | 敏捷力 |
| advantage adv; | 精灵属性 |
| const float timesSuperior | 属性克制倍率 |
| const float timesAttack | 攻击与攻击力倍率 |
| const float timesDefend | 防御与防御力倍率 |
| const float timesLevelUp | 升级加成倍率 |
| const float timesHP | 生命力与生命值倍数 |
| const float timesCritical | 暴击伤害倍率 |
| const string enemyName[] | 怪物名称 |
| const int enemyCnt | 怪物总种类 |
| enum advantage | 属性，枚举类型 |
| 战斗(Battle) | static Battle\* getInstance() | 获取单件 |
| static Battle \_instance | 单件实例 |
| int pk(char\*, Fairy\* const, Fairy\* const, User\* const, const int) | 在两个指定精灵间发起战斗 |

* 客户端

|  |  |  |
| --- | --- | --- |
| 主窗口(MainWindow) | char ADDR\_SRV[32] | 存储IPv4格式的服务器地址 |
|  | QUdpSocket\* sock | UDP发送套接字 |
|  | void sockListen() | 接收服务端的报文并分配至其他函数中 |
|  | void getAttackName(const int) | 获取指定序号精灵的攻击名称 |
|  | void requestArena() | 请求决斗场精灵信息 |
|  | void requestSelfFairy() | 请求自己的精灵信息 |
|  | void requestOthersFairy() | 请求他人的精灵信息 |
|  | void report(const QString&) | 将内容反馈至信息框 |
|  | QString time\_t2QString(const time\_t) | 将时间类型转换为QString类型 |
|  | Ui::MainWindow \*ui | ui设计文件 |
|  | struct sockHeder | 报文头结构 |
|  | struct fairyInfo | 精灵信息结构 |
|  | struct infoOnlinePlayer | 玩家列表信息结构 |
|  |  |  |

1. **各模块设计说明**

* **服务端**

|  |  |
| --- | --- |
| 服务器(Server) | 采用单件设计 |
| 负责处理客户端发送的信息，分配任务至其余模块中 |
| 维护服务端的精灵列表 |
| 验证(Auth) | 采用单件设计 |
| 实现注册、登录、登出功能 |
| 维护注册、在线玩家列表 |
| 引导至用户精灵信息、用户名及密码等所有信息的存储位置 |
| 备份及还原所有信息 |
| 用户(User) | 存储用户名、精灵列表等基本信息 |
| 实现对精灵的送出、治疗等操作 |
| 维护徽章信息 |
| 维护战斗次数、胜率等信息 |
| 精灵(Fairy) | 存储精灵名称、属性、经验、攻击类型等基本信息 |
| 负责造成伤害、接受伤害、治疗等实现 |
| 战斗(Battle) | 采用单件设计 |
| 负责战斗详细判定的具体实现 |
| 维护战斗详细信息的记录和构造 |

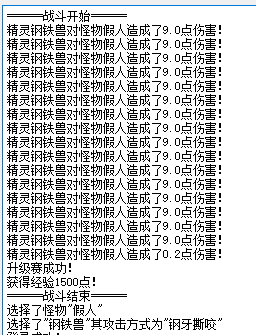
* **客户端**

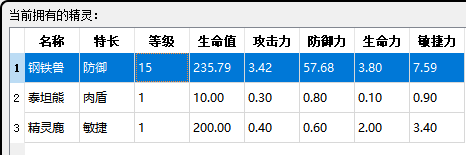
|  |  |
| --- | --- |
| 登录框 | 输入用户名及密码 |
| 进行注册、登录、登出操作 |
| 玩家列表 | 查看当前在线玩家 |
| 获取玩家胜率、精灵等信息 |
| 精灵框 | 显示自己的精灵信息 |
| 进行送出、治疗操作 |
| 显示其他玩家的精灵信息 |
| 决斗场 | 显示服务器上精灵的信息 |
| 信息框 | 显示当前的操作信息 |
| 传递服务器发送的战斗详细信息、提示等内容 |
| 徽章框 | 获取徽章信息 |
| 服务器IP地址框 | 进行修改服务器地址的操作 |

1. **测试情况**

* 题目一



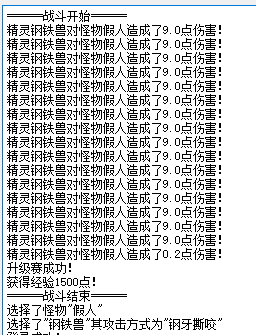


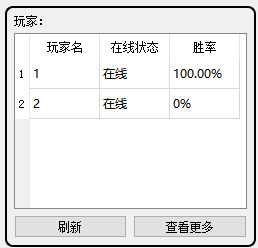


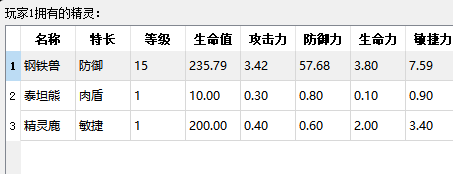
* 题目二



* 题目三







1. **评价和改进意见**

界面只有文字，暂时不够美观。

数值设计待改进，游戏性不够强。

1. **用户使用说明**

* 服务器选择



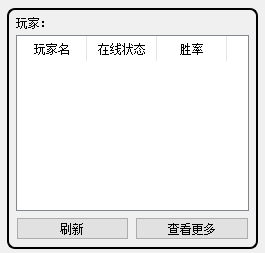
在界面中服务器IP地址窗口中填写要登录的服务器IP地址。

* 注册、登录与登出



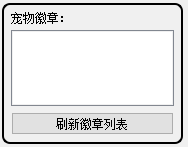
在界面中输入用户名、密码后，即可选择【注册】、【登录】，在登录后可选择【注销】。

* 玩家列表



点击该界面下的【刷新】查看所有服务器上的玩家。选择某一玩家后点击【查看更多】，即可查询到该玩家的精灵信息。

* 徽章



点击该界面下的【刷新徽章列表】，即可查询到当前的徽章信息。

* 战斗

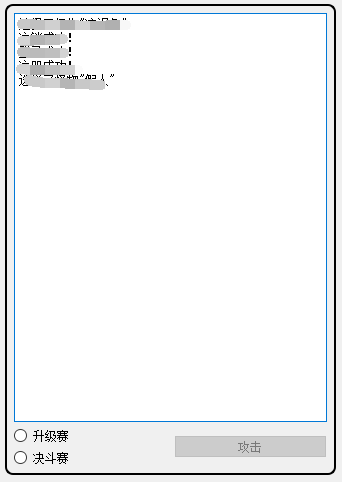


在【决斗场】及【当前拥有的精灵】界面中，请单击选择要出战的精灵及要对战的怪物。

1. **生成方法**

服务端：服务端使用Visual Studio 2019进行开发，控制台应用。

客户端：客户端使用Qt Creator 4.11.1 (Community)进行开发，构建套件为Desktop Qt 5.14.2 MinGW 32-bit。



随后在此选择升级赛、决斗赛类型，并单击右下角【攻击】即可开始战斗。