《面向对象程序设计实践（C++）》综合实验实验报告

**单词消除游戏系统设计与开发**

1. 实验要求

使用面向对象编程的思想完成一个单词消除游戏。游戏由两类参与者组成：闯关者（即游戏玩家），出题者（为游戏增加游戏中使用单词）。游戏规则为，游戏每一轮，程序会根据该关卡难度，显示一个单词，一定时间后单词消失。闯关者需要在相应地方输入刚刚显示并消失的单词，如果闯关者输入正确（即闯关者输入的单词与刚刚显示的单词完全一致，包含大小写）则为通过。一关可以由一轮或者多轮组成。

在基本游戏规则的基础上，课程设计还需学生扩展玩家注册登陆，查询，CS客户端服务端通信等功能。

1. 实验环境

相关实验环境参数如下：

* 系统：macOS Mojave 10.14.5 (18F132)
* IDE：Qt Creator 4.9.0
* 编译环境：Qt 5.12.2 (Clang 10.0 (Apple), 64 bit)
* 数据库：Sqlite3

1. 数据结构

单词消除游戏依赖现有用户和单词数据进行相关操作，本系统以 SQLite3 为数据库工具，存放了3个数据表，分别为breaker（闯关玩家）、maker（出题玩家）和word\_table（单词表）。其各表项和具体数据一例如下：

（一）、breaker和maker

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用户ID/id | 用户名/username | 用户密码/password | 单词数/mark | 经验值/xp | 闯关数/level |
| 1 | ghz | 123 | 24 | 124 | 10 |

其中id、mark、xp和level为integer数据，用户名和用户密码为text数据。

* 用户ID为用户唯一标识符，每个用户会拥有一个独一无二的ID。
* 用户名和用户密码为用户登录凭证。
* 单词数于breaker而言为已消除单词数，于maker而言为已出题单词数。
* xp为消除单词所获得的经验，level为当前最远闯关距离。

（二）、word\_table

|  |  |  |
| --- | --- | --- |
| 单词ID/id | 单词/word | 单词长度/length |
| 1 | ghz | 123 |

其中id和length为integer数据，单词为text数据。

* 单词ID为单词唯一标识符，每个单词会拥有一个独一无二的ID。
* 单词存储单词本
* 单词长度为单词的字符串长度，不包括结束符。

由于涉及到socket通信，需要在客户端和服务端之间进行UDP数据传输，故设计了数据帧的形式，用于交换信息：

#pragma pack(1)

*typedef* *struct* {

short signalType; *//信号类型**，用于判断本次传输要进行的操作*

short userType; *//用户类型*

unsigned short mark; *//单词数*

unsigned short xp; *//经验值*

unsigned short level; *//闯关数*

unsigned short length; *//单词长度*

char username[10]; *//用户名*

char password[15]; *//用户密码*

char word[20]; *//单词*

} Packet;

#pragma pack()

1. 程序结构
2. 客户端
3. client.h

*class* **Client**: *public* QObject {

Q\_OBJECT

*public*:

QUdpSocket \*socket; *//客户端* *Socket*

unsigned short clientPort; *//客户端端口*

QHostAddress clientAddress; *//客户端地址*

*explicit* **Client**(QObject \*parent = *nullptr*); *//构造函数*

void **packPacket**(QByteArray &data, Packet packet); *//打包数据帧*

void **unpackPacket**(QByteArray data, Packet &packet); *//解包数据帧*

*virtual* short ***send***(*const* QByteArray data) = 0; *//发送数据帧*

*private* slots:

*virtual* void ***processPendingDatagram***() = 0; *//接受处理数据帧（虚函数）*

};

1. user.h

*class* **User**: *public* Client {

Q\_OBJECT

*public*:

short userType; *//用户类型*

QString username; *//用户名*

QString password; *//用户密码*

unsigned short mark; *//单词数*

unsigned short xp; *//经验值*

unsigned short level; *//闯关数*

*explicit* **User**(QObject \*parent = *nullptr*); *//构造函数*

void **checkUser**(); *//验证用户*

void **insertUser**(); *//创建用户*

void **updateUser**(); *//更新用户信息*

void **searchUser**(QString name, short type); *//搜索用户*

void **signinUser**(); *//登陆用户*

void **signoutUser**(); *//注销用户*

short ***send***(*const* QByteArray data); *//发送数据包*

signals:

void **signupSignal**(Packet recPacket); *//注册信号*

void **signinSignal**(Packet recPacket); *//登陆信号*

void **getInfoSignal**(Packet recPacket); *//用户信息到达信号*

void **getResultSignal**(Packet recPacket); *//查询数据到达信号*

*private* slots:

void ***processPendingDatagram***(); *//数据帧处理槽函数*

};

1. game.h

*class* **Game**: *public* Client {

Q\_OBJECT

*public*:

QStringList wordList; *//本局游戏单词表*

unsigned short difficulty; *//当前出题难度*

unsigned short currentLevel = 1; *//当前关卡数*

unsigned short currentTimeLimted = 5; *//当前时间限制*

*explicit* **Game**(QObject \*parent = *nullptr*); *//构造函数*

void **getWord**(unsigned short wordLength); *//拉取单词*

void **updateWord**(string \_word); *//更新单词*

void **endGame**(); *//结束游戏*

short ***send***(*const* QByteArray data); *//发送数据帧*

signals:

void **getWordSignal**(Packet recPacket); *//单词到达信号*

void **updateWordSignal**(Packet recPacket); *//单词更新信号*

*private* slots:

void ***processPendingDatagram***(); *//数据帧处理槽函数*

};

1. rank.h

*class* **Rank**: *public* Client {

Q\_OBJECT

*public*:

*explicit* **Rank**(QObject \*parent = *nullptr*); *//构造函数*

void **getRank**(short type); *//拉取排行榜*

short ***send***(*const* QByteArray data); *//发送数据帧*

signals:

void **reciveUserSignal**(Packet recPacket); *//用户数据到达信号*

*private* slots:

void ***processPendingDatagram***(); *//接受处理数据帧（虚函数）*

};

1. 服务端
2. database.h

*class* **Database** {

QSqlDatabase database; *//数据库对象*

*public*:

**Database**(); *//构造函数*

~**Database**(); *//析构函数*

void **connect**(); *//连接数据库*

void **close**(); *//关闭数据库*

QSqlQuery **execute**(QString sqlCommand); *//执行* *sql* *语句*

void **initDatabase**(); *//创建数据库*

};

1. server.h

*class* **Server**: *public* QObject {

Q\_OBJECT

*public*:

Database database; *//数据库*

QUdpSocket \*socket; *//Socket*

*explicit* **Server**(QObject \*parent = *nullptr*);

void **packPacket**(QByteArray &data, Packet packet); *//打包数据帧*

void **unpackPacket**(QByteArray data, Packet &packet); *//解包数据帧*

short **send**(*const* QByteArray data, QHostAddress remote, unsigned short port); *//发送数据帧*

*private* slots:

*virtual* void ***processPendingDatagram***() = 0; *//数据帧处虚函数*

};

1. user.h

*class* **User**: *public* Server {

Q\_OBJECT

*private*:

QStringList makerOnline; *//在线* *maker*

QStringList breakerOnline; *//在线* *breaker*

*public*:

*explicit* **User**(QObject \*parent = *nullptr*);

signals:

void **updateSignal**(Packet recPacket); *//更新用户信息请求信号*

void **signupSignal**(QHostAddress remote, unsigned short port, Packet recPacket); *//注册请求信号*

void **signinSignal**(QHostAddress remote, unsigned short port, Packet recPacket); *//登陆请求信号*

void **logoutSignal**(Packet recPacket); *//登出请求信号*

void **userInfoSignal**(QHostAddress remote, unsigned short port, Packet recPacket); *//拉取用户信息请求信号*

void **searchUserSignal**(QHostAddress remote, unsigned short port, Packet recPacket); *//搜索用户请求信号*

*private* slots:

void ***processPendingDatagram***(); *//数据帧处理槽函数*

void **updateUser**(Packet recPacket); *//更新用户槽函数*

void **logoutUser**(Packet recPacket); *//登出用户槽函数*

void **creatUser**(QHostAddress remote, unsigned short port, Packet recPacket); *//创建用户槽函数*

void **validateUser**(QHostAddress remote, unsigned short port, Packet recPacket); *//验证用户槽函数*

void **getUserInfo**(QHostAddress remote, unsigned short port, Packet recPacket); *//获取用户信息槽函数*

void **searchUserInfo**(QHostAddress remote, unsigned short port, Packet recPacket); *//搜索用户信息槽函数*

};

1. game.h

*class* **Game**: *public* Server {

Q\_OBJECT

*public*:

*explicit* **Game**(QObject \*parent = *nullptr*);

signals:

void **getWordSignal**(QHostAddress remote, unsigned short port, Packet recPacket); *//拉取单词请求信号*

void **updateWordSignal**(QHostAddress remote, unsigned short port, Packet recPacket); *//更新单词请求信号*

*private* slots:

void ***processPendingDatagram***(); *//数据帧处理槽函数*

void **getWord**(QHostAddress remote, unsigned short port, Packet recPacket); *//拉取单词槽函数*

void **updateWord**(QHostAddress remote, unsigned short port, Packet recPacket); *//更新单词槽函数*

};

1. rank.h

*class* **Rank**: *public* Server {

Q\_OBJECT

*public*:

*explicit* **Rank**(QObject \*parent = *nullptr*);

signals:

void **getUserSignal**(QHostAddress remote, unsigned short port, Packet recPacket); *//排行榜数据请求信号*

*private* slots:

void ***processPendingDatagram***(); *//数据帧处理槽函数*

void **getUserByRank**(QHostAddress remote, unsigned short port, Packet recPacket); *//排行榜数据拉取槽函数*

};

1. 界面设计
2. 登陆注册界面

图片包含 屏幕截图, 监视器

描述已自动生成图片包含 屏幕截图, 监视器

描述已自动生成

1. 主界面

图片包含 屏幕截图

描述已自动生成

1. 游戏界面

图片包含 监视器, 电子产品, 屏幕截图, 室内

描述已自动生成

图片包含 屏幕截图, 监视器, 电子产品, 室内

描述已自动生成图片包含 监视器, 室内, 屏幕, 电子产品

描述已自动生成

1. 排行榜界面

图片包含 监视器, 黑色, 屏幕

描述已自动生成

1. 搜索用户界面

图片包含 监视器, 屏幕截图, 黑色, 室内

描述已自动生成

1. 服务端请求命令行

图片包含 屏幕截图

描述已自动生成

1. 基本程序逻辑

整个单词消除游戏分为Client客户端和Server服务端两部分，客户端负责交互和数据显示，服务端负责数据的处理和呈递。

用户首次进行注册后，方可登陆进行游戏，对于闯关者即breaker而言，游戏每3个单词为一关，并随着已消除单词的个数不断增加难度，难度体现在单词长度不断增加，显示时间和输入时间不断变短。并随着玩家输入正确的单词后，增加值为单词长度个数的经验值，倘若玩家直到超时都未输入成功，便会被淘汰退出游戏，可重新进行闯关；对于maker出题者而言，登陆后随时可以开始出题，出题界面会显示本次登陆所有出过的单词，如果输入了数据库中已有的单词，用户便会得到相应提示。

用户每时每刻只可在一处登陆，如果重复登陆会得到登陆重复的提示并把另一处登陆踢出下线。

1. 心得体会

开发过程中遇到的第一个大问题就是对类与对象的理解。由于之前从未详细了解过类与对象的详细运作方式，所以对其背后的机制和原理都很不了解，所以要想开始这次实验，必须经历一个理论知识完全从无到有的过程。好在老师经验丰富，通过本学期的课堂学习，就大致掌握了类与对象的基本理论知识。

遇到的第二个问题是对类与对象底层知识的理解。在学习了类与对象的表层知识后，尚还匮乏的便是对其底层——虚函数，构造函数，Qt槽函数这类高级特性的理解。但是考虑到C++这门语言有类似Qt等成熟的第三方库可以调用，使得开发者可以在掌握些许并非全部详细知识的情况下方便的调用相关功能，这个问题也便迎刃而解了。

实验遇到的第三个问题是对Socket以及数据库的了解。数据库功能全面但复杂，Socket原始但强大，所以搞清楚每一个部分的意义十分重要，在写代码进行处理时也较为繁琐。

通过本次实验，我经历了一次“从不了解到写出完整成品”的开发过程。过程中学习到了很多知识，实践了很多之前只是浅尝辄止的尝试，是对自我学习能力和开发能力一次很有意义的提升。