Tango v3.2.0 文档

Myriad-Dreamin 2017211279 2017211301

所有版本的 github 托管网站：https://github.com/Myriad-Dreamin/Tango

1 Tango 架构

1.1 服务器端 -客户端架构

Tango 采用服务器端 -客户端架构。两者使用公共的 TangoLib(TangoCommon) 库。在 TangoCommon 中目前一共包含六大内容。

automator

automator 中包含两大类，一是 GameAutomation, 二是 GameConfig.

GameAutomation 共同的基类是 AbstractGameAutomation，GameAutomation 真正实现了游戏的逻辑，而 GameAutomationRelayer 仅仅拥有一个信号转发机制。

GameAutomation 依赖的一个类是 GameConfig.GameConfig 中包含了各种游戏的设置，例如经验更新，例如回答时间的计算，例如当前游戏的单词池。

client

client 中只有一种类，即 Client 类。Client 类都有一个共同的抽象基类 AbstractClient，其下有三个类。

LocalClient 负责完成与本地数据库交互的逻辑，RemoteClient 负责完成与远程服务器交互的逻辑，Client 负责完成 LocalClient 或者 RemoteClient 与客户端图形化界面的配接。

component

component 中包含一些常用的组件。Logger 是分级的日志管理，MessageBox 是适用于客户端的特殊的消息框，PairTableItem 用于一些组合型的 Widget 场景，TimerWidget 是一个用于显示可倒计时的特殊 Wdiget。

network

network 中包含服务器端和客户端共享的一套网络协议。SocketX 为 QTcpSocket 的子类，在 QTckSocket 的基础上约定了包发送和接受的处理。client_rpc 是客户端和服务器端收发消息遵守的一套 json_rpc 协议。

players

players 中包含一大类，Player 是 Consumer 和 Author 的共同基类。Player 中包含与数据库交互的底层逻辑和自身所带的属性信息。

types

typers 是杂类，包含一系列有意义的枚举常量 (enumeration) 或者命名元组 (named tuple)。

1.2 Tango 库

1 Class automator/AbstractGameAutomation 抽象自动机类

2 Class automator/AbstractGameAutomation.GameAutomation 自动机类

3 Class automator/AbstractGameAutomation.GameAutomationRelayer 转接自动机类

4 Class automator/GameConfig 游戏设定类

5 Class client/AbstractClient 抽象客户端类

6 Class client/AbstractClient.LocalClient 本地客户端类

7 Class client/AbstractClient.RemoteClient 远程客户端类

8 Class client/AbstractClient.Client 客户端类

9 Class component/Logger 日志类

10 Class component/MessageBox 消息框类

11 Class component/PairTableItem 表对类

12 Class component/TimerWidget 计时器类

13 Library network/client_rpc 消息 rpc 库

14 Class network/SocketX TangoSocket 类

15 Class players/Player 玩家类

16 Class players/Author 作者类

17 Class players/Consumer 读者类

18 Enum types/RetriveMode 数据库 Fetch 模式枚举类型

19 Class types/TangoPair 单词类

20 Class types/UserBriefInfo 简略用户信息类

21 Class types/UserFullInfo 详细用户信息类

22 Enum types/UserStatus 用户状态枚举类型

1.3 客户端

1 Function main 主程序入口

2 Class MainWindow 窗口类

3 Class Scene 场景类

4 Class Scene.CreationScene

5 Class Scene.MainScene

6 Class Scene.MultiplayScene

7 Class Scene.PlayingScene

8 Class Scene.PlaySettleScene

9 Class Scene.PlaySubScene

10 Class Scene.QueryUsersScene

11 Class Scene.RankingAuthorsScene

12 Class Scene.RankingConsumersScene

13 Class Scene.RegisterScene

14 Class Scene.SelectingScene

1.4 服务端

1 Function main 主程序入口

2 Class MainWindow 窗口类

3 Class engine/TcpServer 服务器类

4 Class engine/TangoThread 服务器线程类

2 核心逻辑

2.1 Client 转接逻辑

LocalClient 和 RemoteClient 都是 AbstractClient 的子类。MainWindow(客户端) 不关心 LocalClient 和 RemoteClient 具体实现，仅仅是利用了 Client 类的 `setup_remote_connection` 或是 `setup_local_connection` 申请 touch Model。

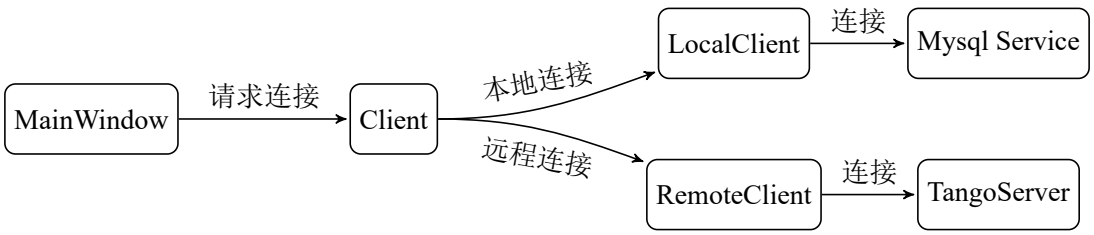


图 1: Client 转接逻辑 (连接)

申请服务时，TangoServer 实际上也使用 LocalClient 与服务器上的 Mysql 服务交互。

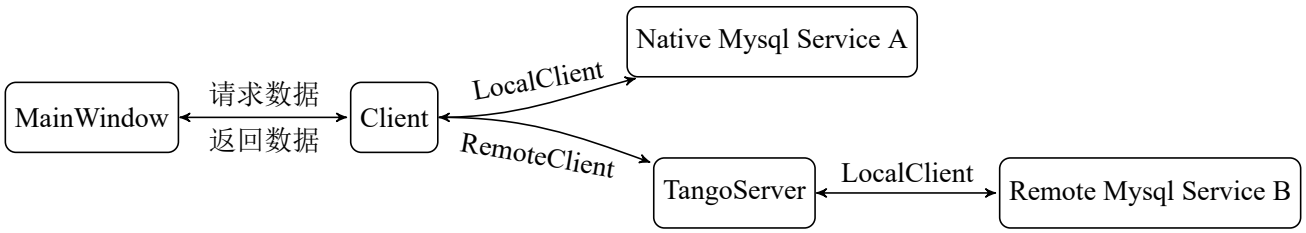


图 2: Client 转接逻辑 (请求数据)

事实上，由于 Client 的设计，TangoServer 可以是一个子服务器，继续转接母服务器。

2.2 游戏逻辑

2.2.1 MainWindow 向 Client 请求游戏服务

GameAutomation(简称 GA) 和 GameAutomationRelayer(简称 GAR) 都是 AbstractGameAutomationRelayer(简称 AGA) 的子类。

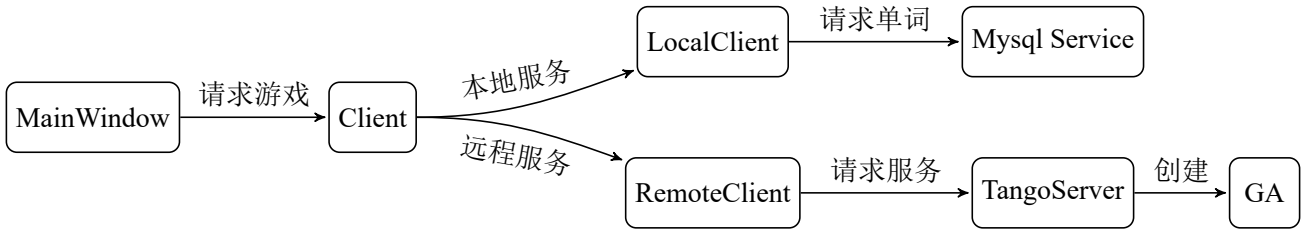


图 3: MainWindow 请求游戏 (请求)

MainWindow 从 Client 请求到的是 AGA 的 handler，根据 AGA 发出的信号做出动作。

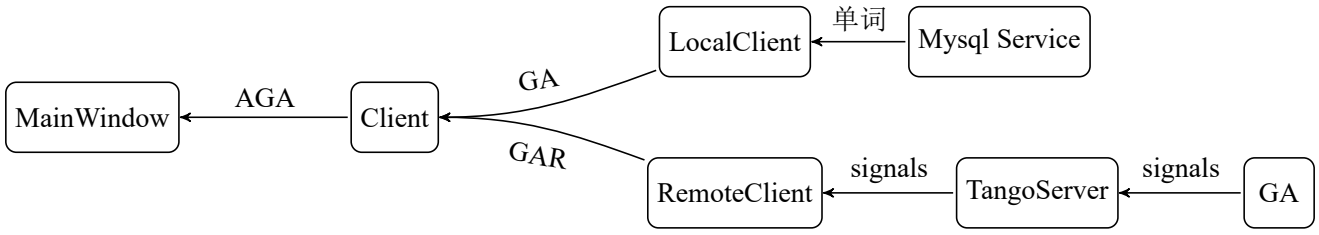
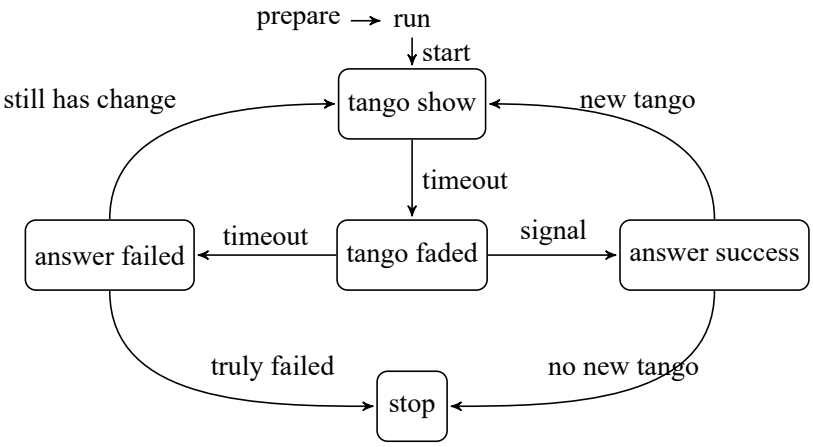


图 4: MainWindow 请求游戏 (返回 handler)



$g(\text{tango show}, \text{tango faded}) = \text{new tango}, \text{fade time}$
$g(\text{tango faded}, \text{answer failed}) = \text{tango faded}, \text{answer time}$
$g(\text{tango faded}, \text{answer success}) = \text{tango faded}, \text{answer time}$
$g(\text{answer failed}, \text{stop}) = \text{failed}, g(\text{answer success}, \text{stop}) = \text{success}$

图 5: GA 自动机

2.2.2 AGA 信号发射

MainWindow 就是根据这个自动机的输出完成一系列的游戏动作。

2.2.3 GameConfig 的功能

GameConfig 允许定制下面几个参量：

1 每次显示单词的定时器时间变化

2 每次回答单词的定时器时间变化

3 每次回答失败是否真的失败的选项

4 每次单词回答成功经验如何变化

5 初始的显示单词的定时器时间

6 初始的回答单词的定时器时间

7 单词缓存是否独立

前四个选项分别需要传入四个函数指针.

```
/* 消失时间变换函子，每次单词作答完以后作用一次 */
std::function<void(int&)> fade_functor;
/* 回答时间变换函子，每次单词作答完以后作用一次 */
std::function<void(int&)> ans_functor;
/* 经验计算函子，传入一个单词和当前轮数作为参考 */
std::function<void(int&,TangoPair,int)> exp_functor;
/* 每次失败事件触发以后，调用失败函数函子判断是否仍有机会 */
std::function<bool()> if_failed_functor;
```

例如默认的经验值函数子为：

```
void default_exp_functor(int &x, TangoPair tango, int)
{
    x += tango.first.length();
    return;
}
```

这个经验值函数选择将单词的长度作为经验值增加。当我们切换游戏模式为困难模式时。

```
void hard_exp_increment(int &exp, TangoPair tango, int success_count)
{
    exp = exp + static_cast<int>(
        tango.first.length() * (1.4 + success_count * 0.03) + 0.99
    );
}
```

直接修改这个函数为一个更快非线性的增长即可。

GA 接受一个 GameConfig 参数构造，每次在特定的时刻调用 Config 中的函子或者使用其中的参量。

以此为基础，本文设置了四个不同的难度和经验等级，四个不同的单词选取模式，一共 16 种组合，并在难度较大的情况下，让玩家拥有多次复活重新答题的机会（斟酌给了一次）。

如此看来，将 GA 和 GameConfig 解构，确实是一个不错的选择。

2.3 MainWindow 在客户端中的功能层次

MainWindow 将每个不同的界面组织为场景。将自己身为客户端界面的主动承担者化为被动服务者。MainWindow 将 CenterWidget 让诸场景类，只提供与 Client 的交互或其他长期存在的全局服务。

MainWindow 现在仅有一个供使用的函数 `switch scene`，用于切换 CenterWidget 内含的场景实例.

当进入场景时, MainWindow 会调用 scene 的 `on_incoming` 无状态函数，当离开场景时，MainWindow 会调用 scene 的 `on_exiting` 无状态函数。

2.4 重新设计传输层协议

在 QT 自带的 TcpSocket 传输层的基础上，本文添加了新的传输层协议功能。

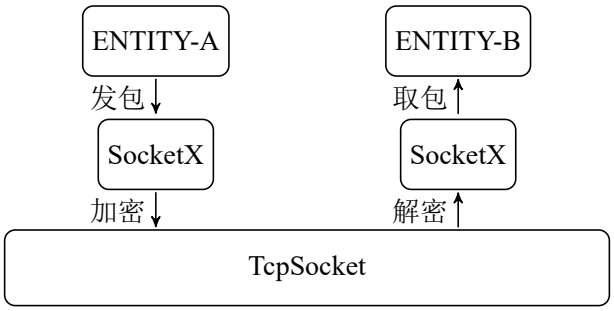


图 6: 包的传输流

SocketX 使用 Base64 可逆的功能性加密算法，避免不可见字符代表的字节出现在字节流中，并约定一个包界限符 (如 SocketX 使用的 0x1 字节)，完成无需等待的异步请求和响应。这种协议的好处是，既拥有线性的时间复杂度（Base64 的时空代价极小），可靠的服务（TCP）特性，又拥有报文式请求特性。

2.5 使用 json-rpc 网络服务

关于 json-rpc 的详细内容，这里不做介绍。json-rpc 就是以 json 字节流服务为基础的 rpc 远程请求服务。本文前面举例的远程服务，将 json-rpc 的无状态服务以约定的方式，完成了服务器端与客户端的同步行为。所谓约定的方式，就是双方约定共同的函数编号，游戏设置编号，信息交换请求。

2.6 难度和等级计算公式举例

以初级难度为例。Fetch 模式为为在长度区间在 [0%, 80%] 比例的单词中随机选取连续的 N 段。回答时间随着单词个数增加每次减少 50ms，至少保证 1s 的单词显示时间。经验见计算公式。

$$atime = \text{dfunc}(x) = \max\{1000, x - 50\}ms, exp = \sum_i \lfloor length[tango[i]] * (1 + i * 0.01) + 0.99 \rfloor$$

2.7 自定义日志管理

QT 的日志管理还是很原始的。因此我重写了一个通用的 C++ 日志管理。

Logger 支持 5 级日志分级管理，支持单例日志申请。暂时不支持支持多线程，不支持自定义文件输出。日志类全程采用 type traits 特性和模板元百编程，因此任意信息进入日志只需要一级转移，速度非常快。

3 v3 基础功能

主界面如图六，比较土。

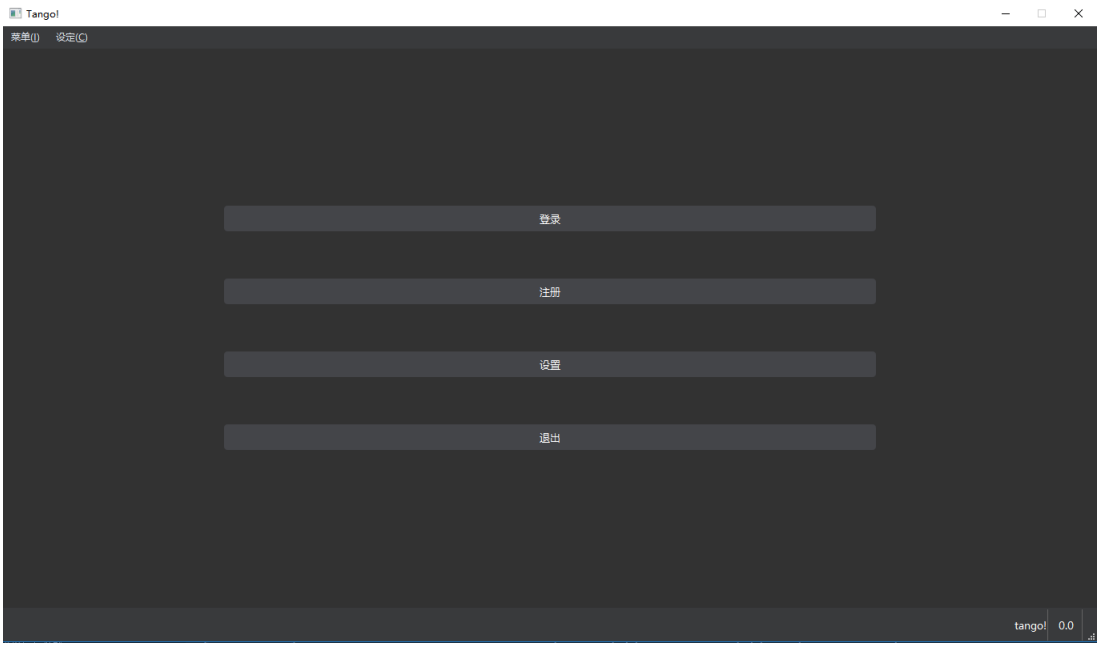


图 7: 主界面

点击登录按钮进入登录界面，点击注册按钮进入注册界面，点击设置按钮进入设置界面，点击退出按钮退出游戏。设置界面是 v3.2 新加入的功能。

登录界面如图七。远程连接是初期定下的功能，将来会移动至设置界面。远程连接的 checkbox 被按下时，选择远程连接，否则选择本地连接服务。

输入正确用户名和密码进入游玩界面。

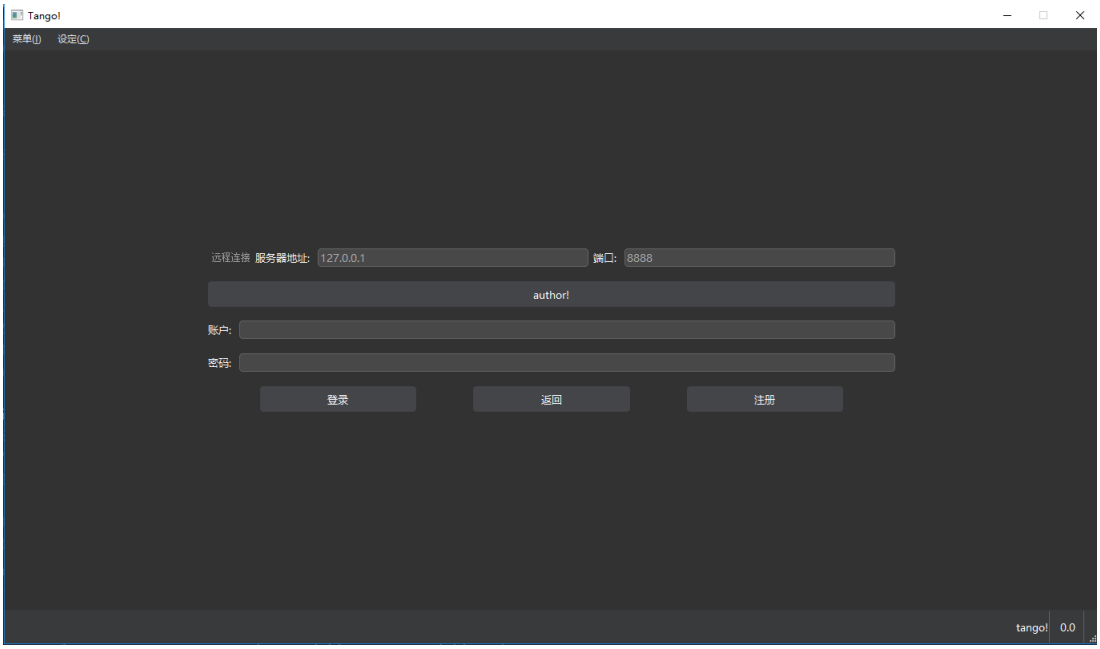


图 8: 登录界面

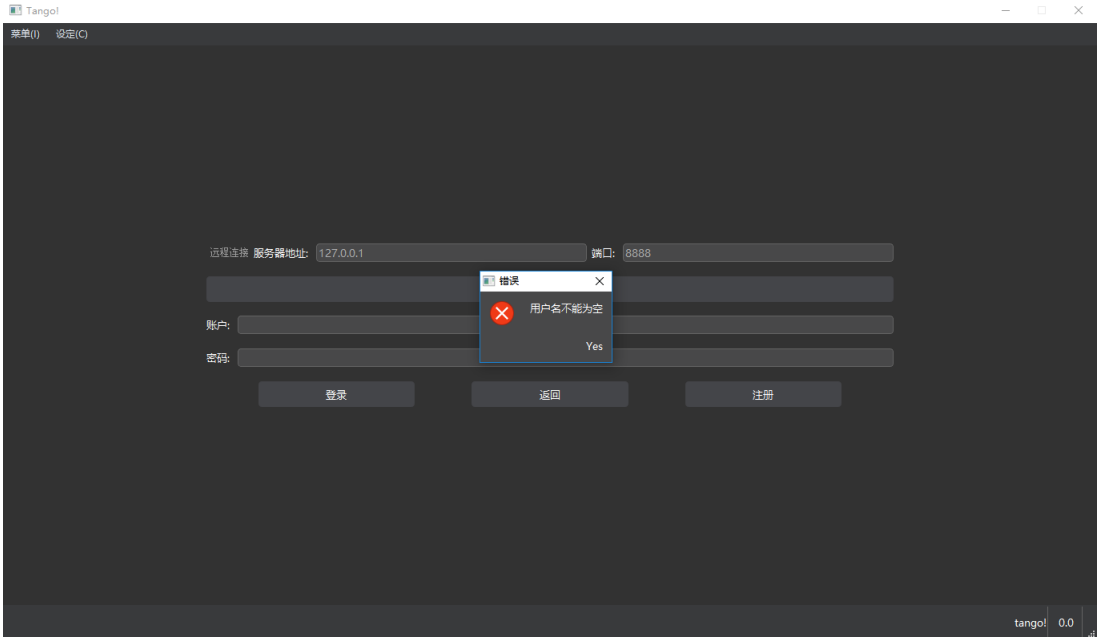


图 9: 用户名为空

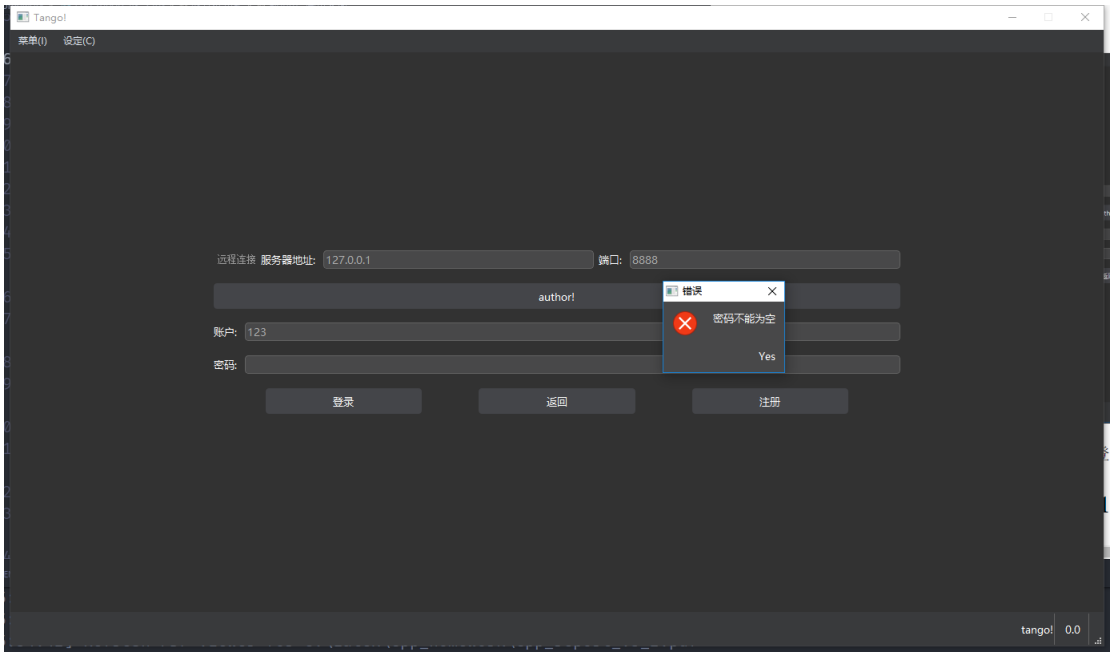


图 10: 密码为空

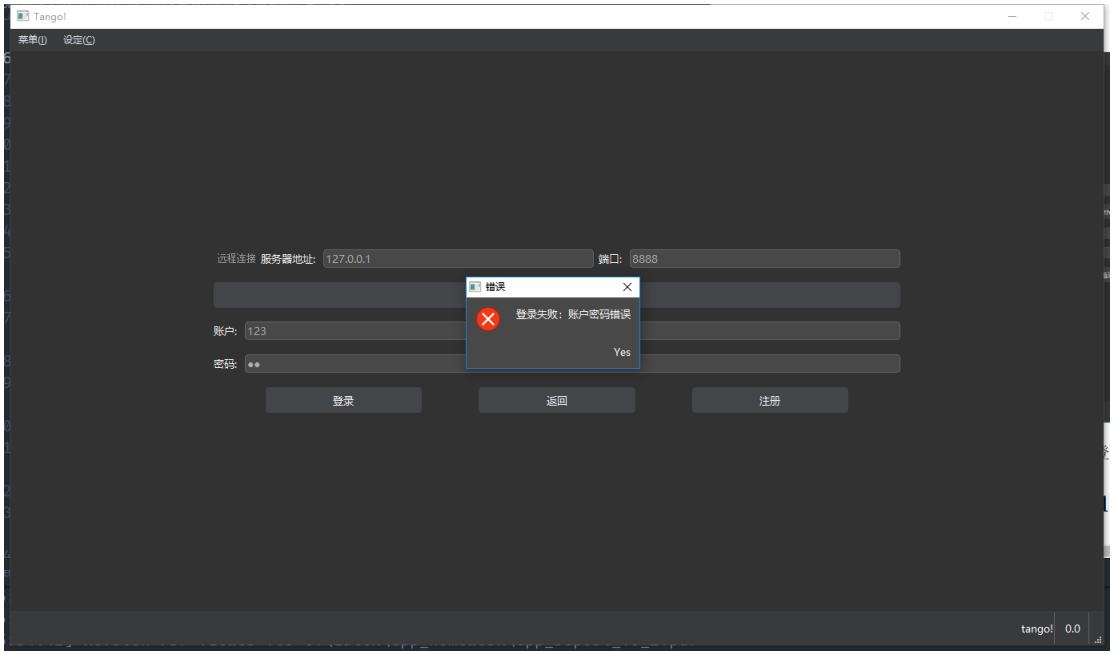


图 11: 密码错误

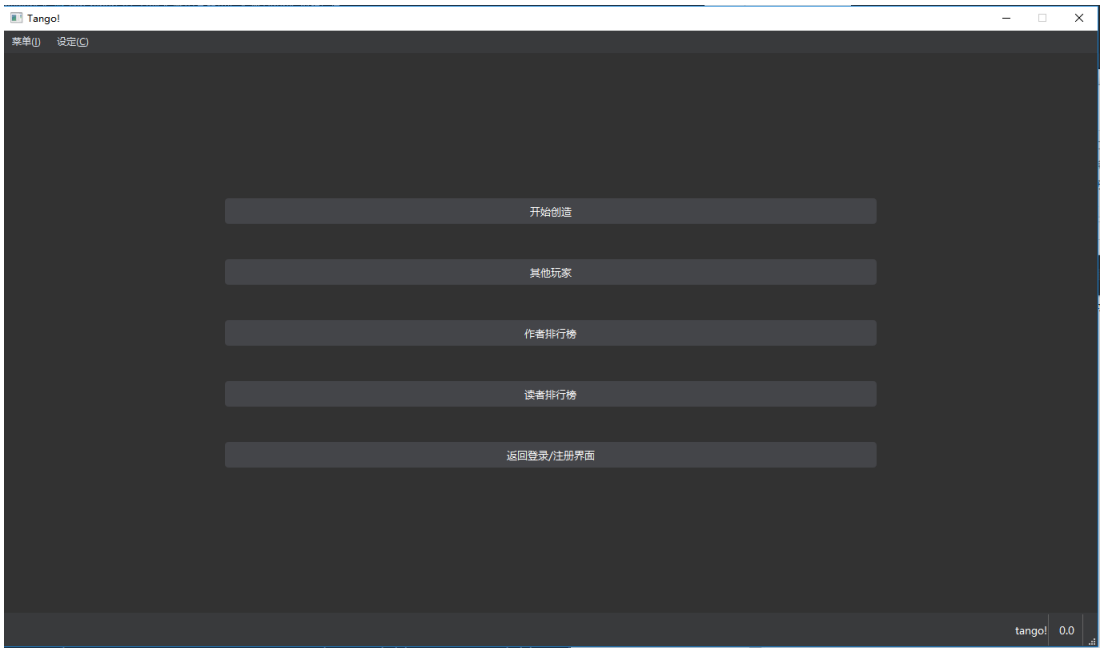


图 12: 作者主界面

当用户为作者时，可以选择创作，查看本机其他玩家，查看作者排行榜，查看玩家（读者）排行榜等。具体内容逻辑比较简单。下面仅显示部分界面内容。

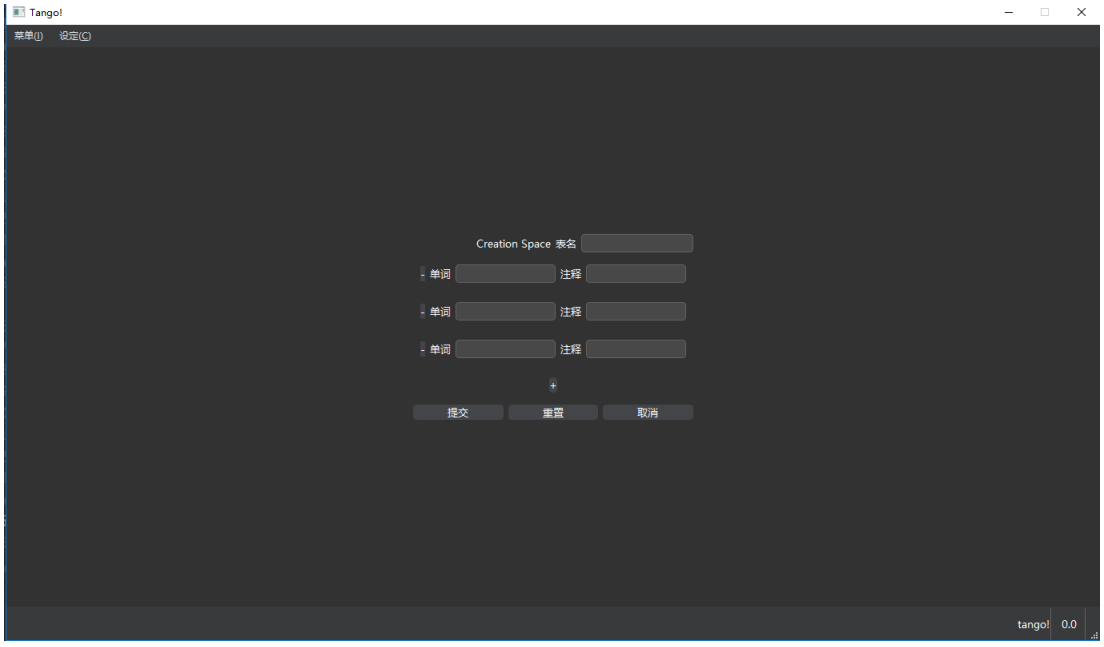


图 13: 创建单词表

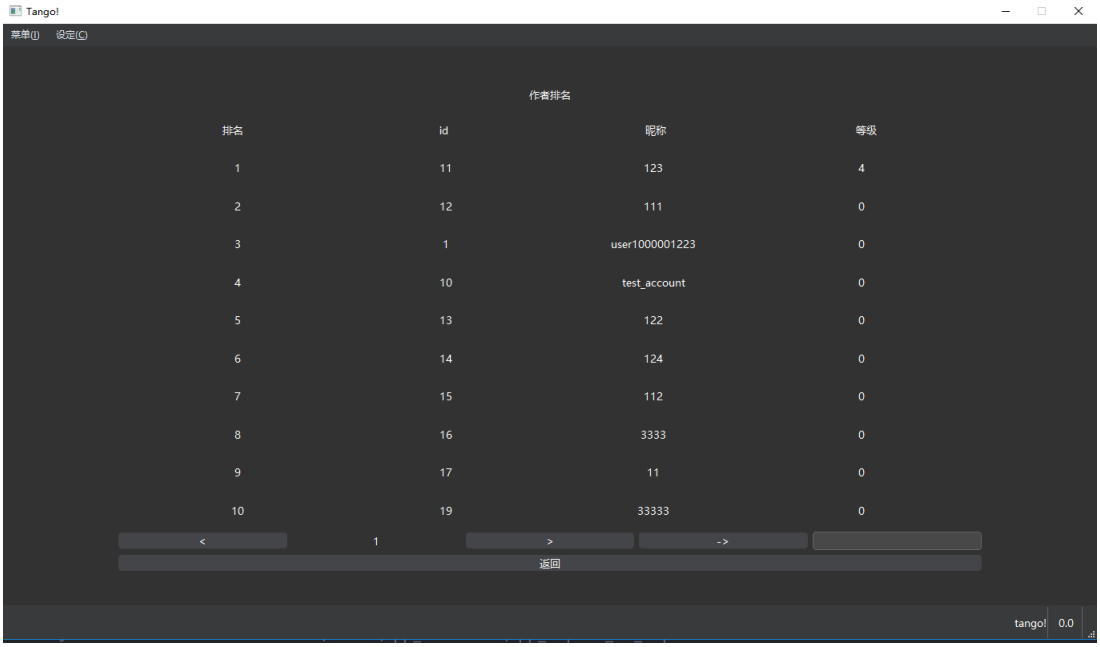


图 14: 作者排行榜 1∼10

图 15: 作者排行榜 11∼12

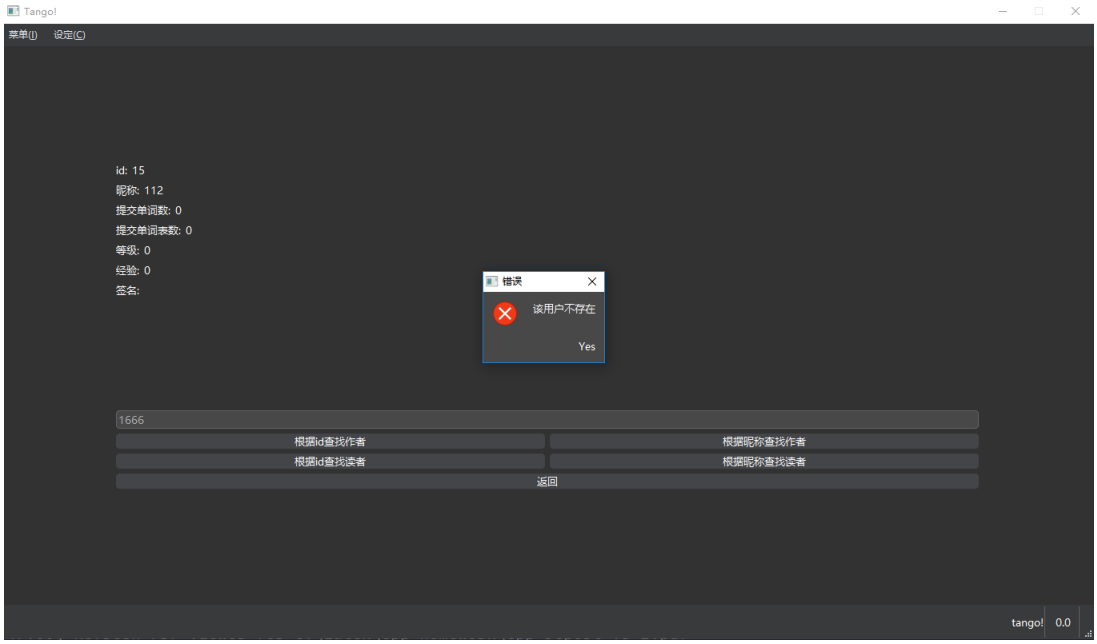


图 16: 查询错误

图 17: 查询成功

我们继续介绍读者身份进入游戏以后会有什么样的功能。远程模式下，如果多用户登录相同账号，那么会登录失败，返回主界面。3.2 以前的版本的主界面为登录界面，这里未做调整，不过不妨碍游戏功能。

现在可以查询其他人的信息，但是挑战还没做，预计 v3.3 加入多人对战功能，因为框架设计比较好，所

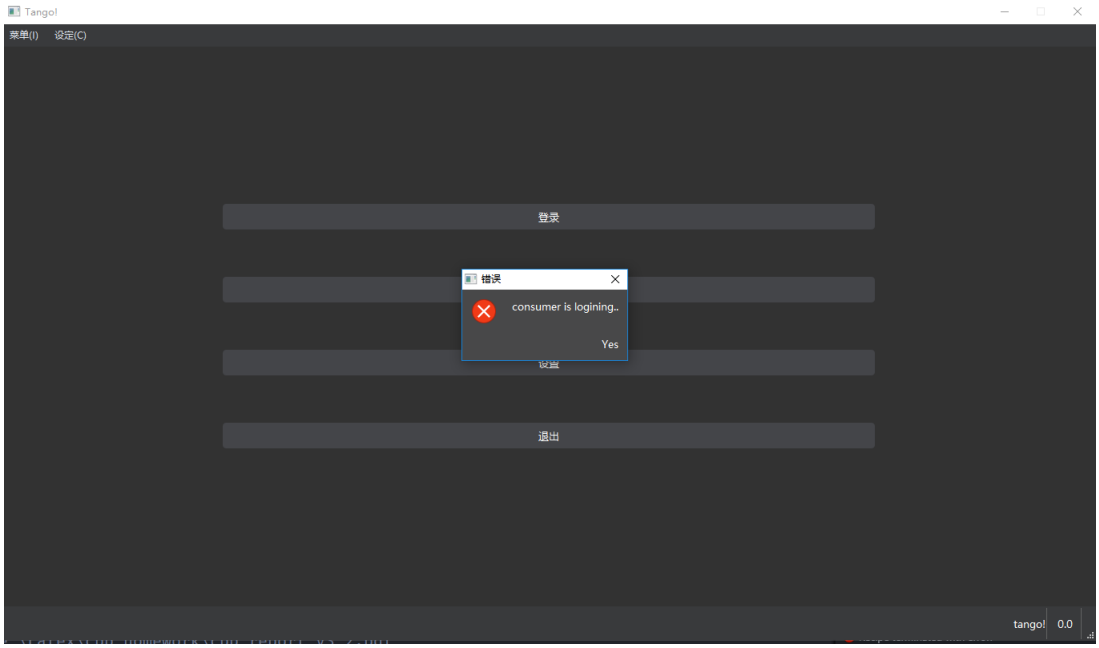


图 18: 登录失败

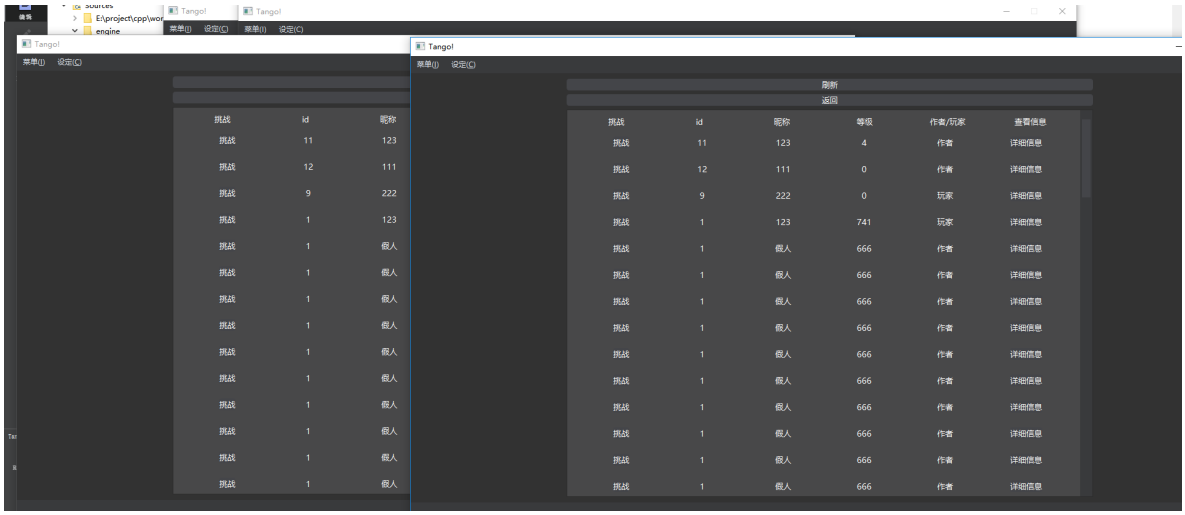


图 19: 多人登录查询

以很容易加。但是已经没时间了，作罢。

图 20: 多人登录查询（详细信息）

联机下，玩家可以使用本地连接上所有相同的功能。并且所有人可以同时与服务器交互。

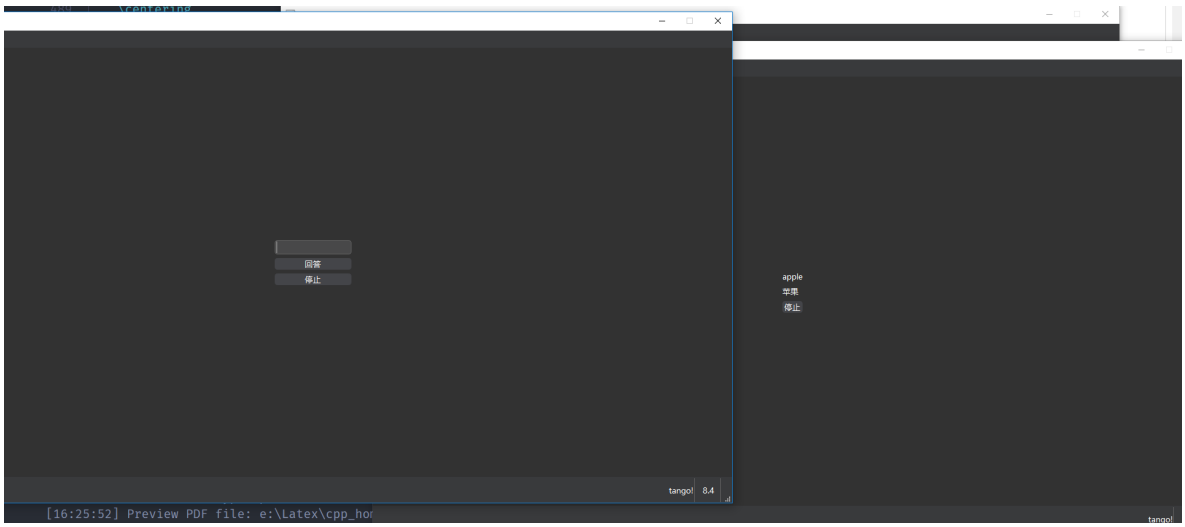


图 21: 多人同时游玩

4 v3.2 相比 v3.1 增加的额外功能

4.1 增加自定义设定功能

目前开放的自定义选项有：设置创作面板默认的单词项个数。默认的本地数据库访问参数，屏幕分辨率设定等。支持文件设置方式，恢复默认选项。

图 22: 设置界面

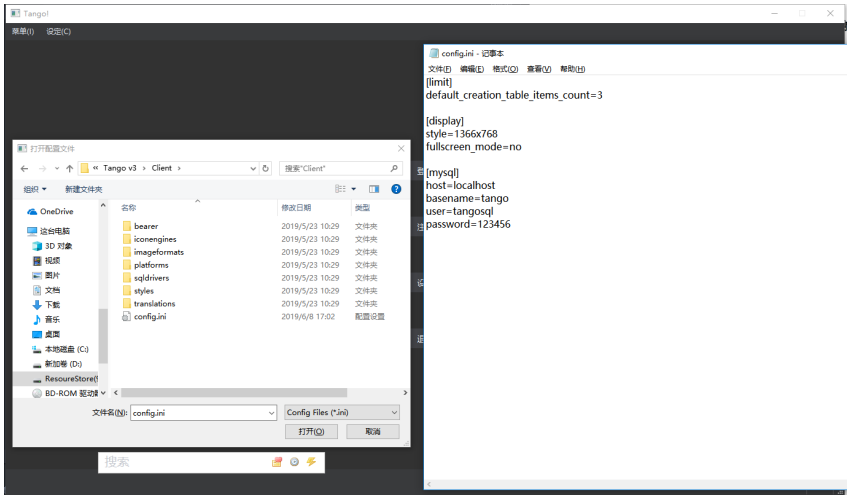


图 23: 选择文件设置

4.2　增加屏幕分辨率设定功能

按 F11 全屏或取消全屏，在设置界面选择合适的分辨率大小。

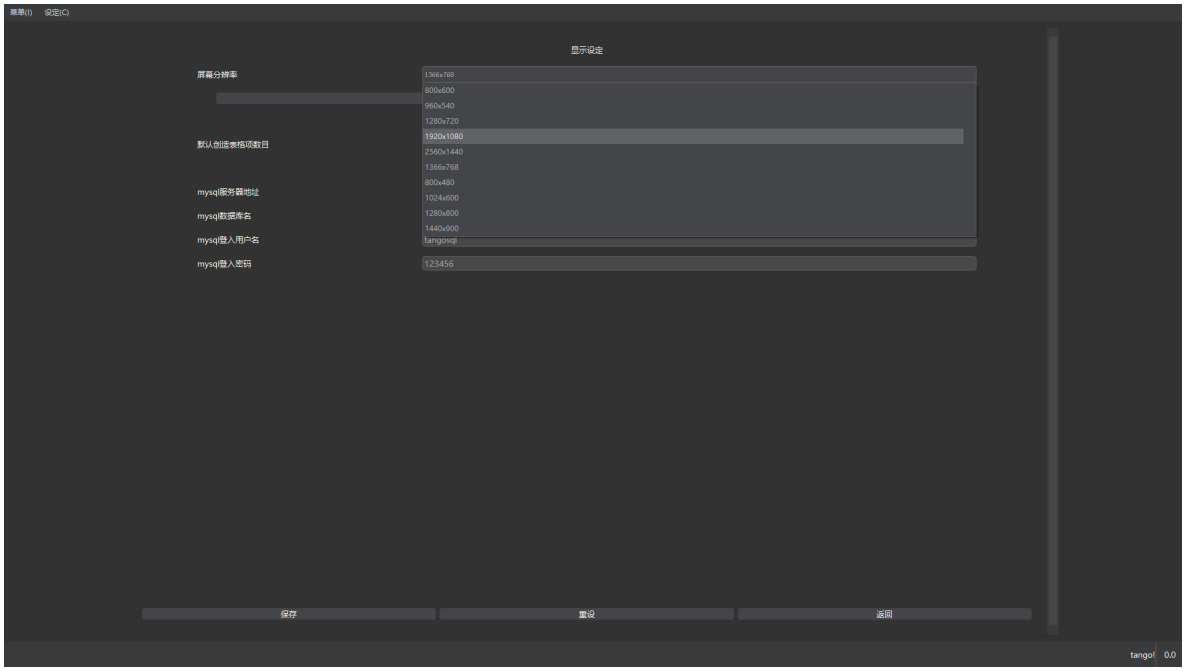


图 24: 设置界面 (分辨率)

图 25: 全屏 (快捷键及菜单栏)