一、 项目分工

学号	名字	角色	班级	职责	贡献
16340025	陈慕远	组长	下午班	负责游戏项目卡牌，地图等素材的提供，以及游戏开发中的测试，游戏玩法的设计和建议，实验报告的编写	30%
16340024	陈铭涛	组员	下午班	负责游戏服务端代码的编写，提供相应的服务器，实现网络游戏通信，游戏玩法的设计和建议，参与开发测试	32%
16340023	陈明亮	组员	下午班	负责游戏 Cocos 部分前端代码的编写，包括游戏逻辑，界面效果，以及各项 UI 的设计和实现，参与开发测试，以及整体设计的构思，编写实验报告	33%
16340007	蔡湘国	组员	下午班	负责游戏设计的建议	5%

二、开发环境

Windows 10, Visual Studio 2017, Cocos2d-x 3.x

三、 项目阐述

项目名称：Schwifty

项目简介：What is Schwifty?

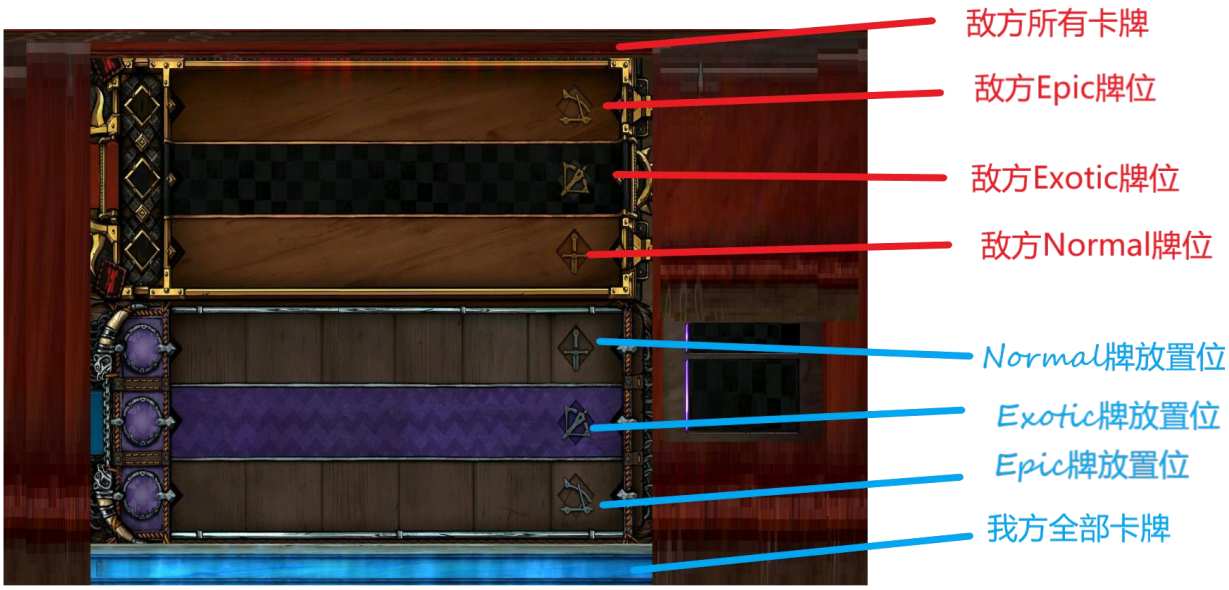
1. 基于 Cocos2d-x 建立的多人在线网络卡牌游戏 -- Schwifty，该游戏旨在为用户提供有趣的卡牌对战体验，支持多人在线实时发送游戏邀请，双方同意后便进入对战界面，开始简单又不失创意的 Morty 牌积分战。本游戏的目的在于将著名的昆特牌轻量化，满足玩家不熟悉昆特牌却又想入门，同时时间不充足的需求，简化游戏逻辑，却不失策略感和趣味性。

2. 游戏的角色建立在动画片《Rick And Morty》之上，游戏中的每一位玩家在开始之前都可以选择自己喜欢的 Rick 角色，进入游戏后系统会自动分发 24 张 Morty 卡牌，每一把游戏开始前系统随机挑选 13 张卡牌用于对战。每个选择不同 Rick 的玩家所对应的角色显示都互不相同，玩家同时也可以在游戏界面上看到其他在线玩家，也可以向他们

发出游戏邀请。

3. 游戏的卡牌对战机制建立在昆特牌之上，采取三局两胜制，每一局的胜负取决于发出的 Morty 手牌的总积分。在牌局上，我们需要采取策略击败对手，必要时需要主动放弃跟牌，只有赢得了两局的玩家才是最终的胜者。

4. 游戏的核心在于如何搭配 Morty 牌组取得胜利，其主要的玩法可以参考昆特牌，下面给出游戏的卡牌面板：



每一张 Morty 牌都具有相应的点数，按照类型可以划分为：Normal，Exotic，Epic，玩家必须按照类型放置对应的牌，前端实现了类型的检测，不通过则视为无效操作，返回手牌中。点击某一张牌时，可以在右边栏看到对应的卡牌详情。

项目涉及的知识点：

1. 游戏场景之间的跳转，包括登录界面，用户移动的主界面，以及卡牌对战界面，增加了基本的游戏窗口大小的自动调整，以及过渡效果。

2. 基本游戏元素的添加，以及游戏角色移动的帧动画调度，还有角色的有效邀请行为 Rect 的判断，即两个角色之间距离和方向达到条件时，在对方上空显示动画，提示邀请。

3. 使用调度器监控服务端 WebSocket 的实时信息，处理网络消息实现相应行为，如接收对战请求，添加上线角色等等。屏幕触摸事件，按钮事件监听器的添加，实现卡牌对战界面发卡，取查看详情，取消发卡，中止本局等等的逻辑；主界面用户角色移动，按下空格键发送请求的行为。

4. TileMap 地图的创建，以及自定义的数据结构 -- MyDialog 的实现，方便了游戏中弹出窗口的多次使用。本地存储用户角色信息，以及其他玩家的对应信息（实现其他玩家物理坐标的变换，同时也加载其动画）。

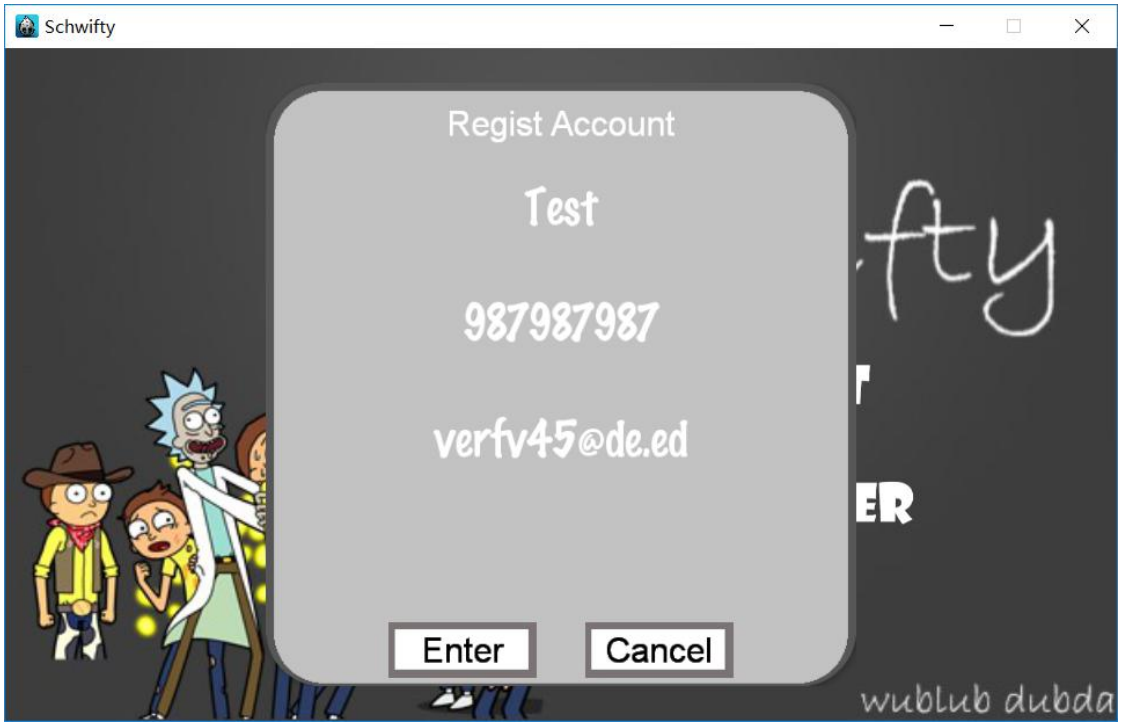
5. 键盘事件对用户角色的控制，以及多个场景之间的不同音乐切换，卡牌界面胜利或失败，平局的音效播放。卡牌事件的处理，监听器的使用，包括拖动牌的位置判断（合法出牌，非法则回到原来位置），鼠标点击的事件触发。

6. 卡牌界面粒子效果的显示和切换（本方出牌显示蓝色火焰，敌方出牌显示红色火焰）。网络访问方面则结合了基本的 POST 行为实现登录和注册功能，用户进入到主界面就接入服务端运行的 WebSocket，实现实时更新，接收信息的功能，是一个实在的网络型轻量级 2D 卡牌游戏。

四、 项目展示

1. 游戏初始界面：

2. 进入游戏需要登录账号，账号可以注册获得：



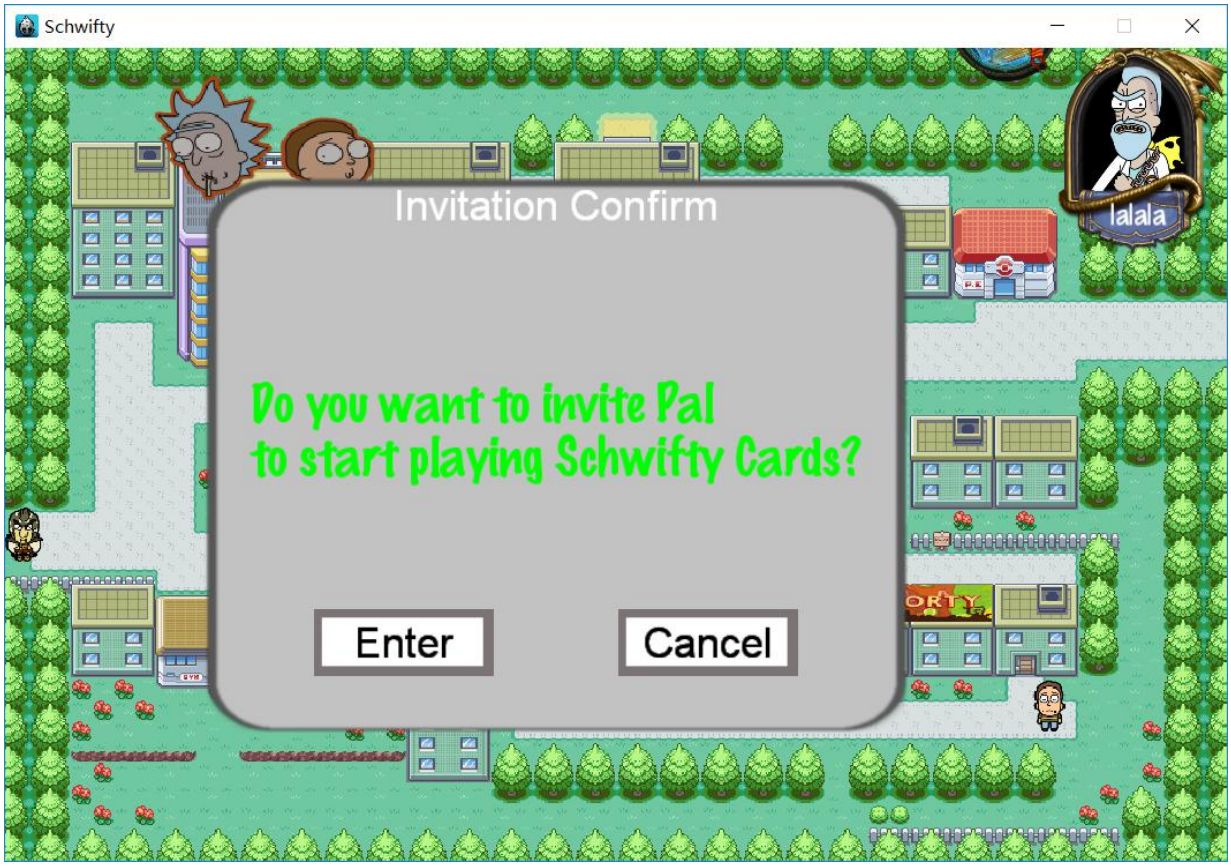
3．登录后的游戏界面：

游戏界面（多个用户）：

4. 可以看到界面右上方除了用户信息还有一个隐藏的按钮，鼠标移上去之后：

5. 点击它即可显示用户当前拥有的牌组：

6. 靠近别的用户时，按下空格键可以发出对战邀请

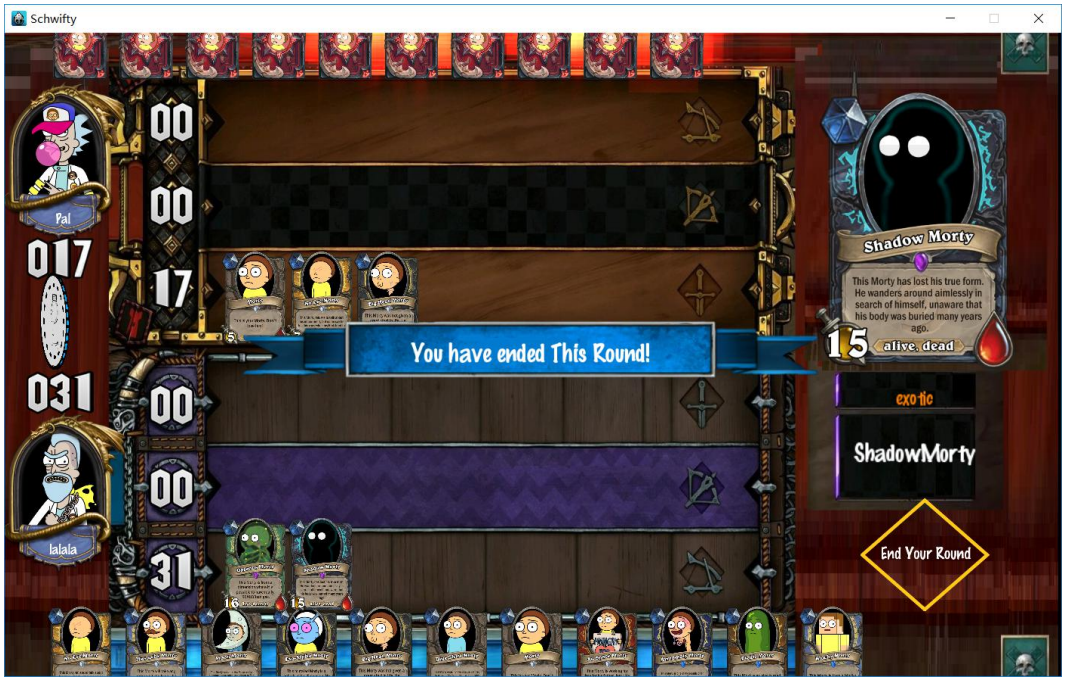


7．同样，也可能收到来自别人的对战邀请

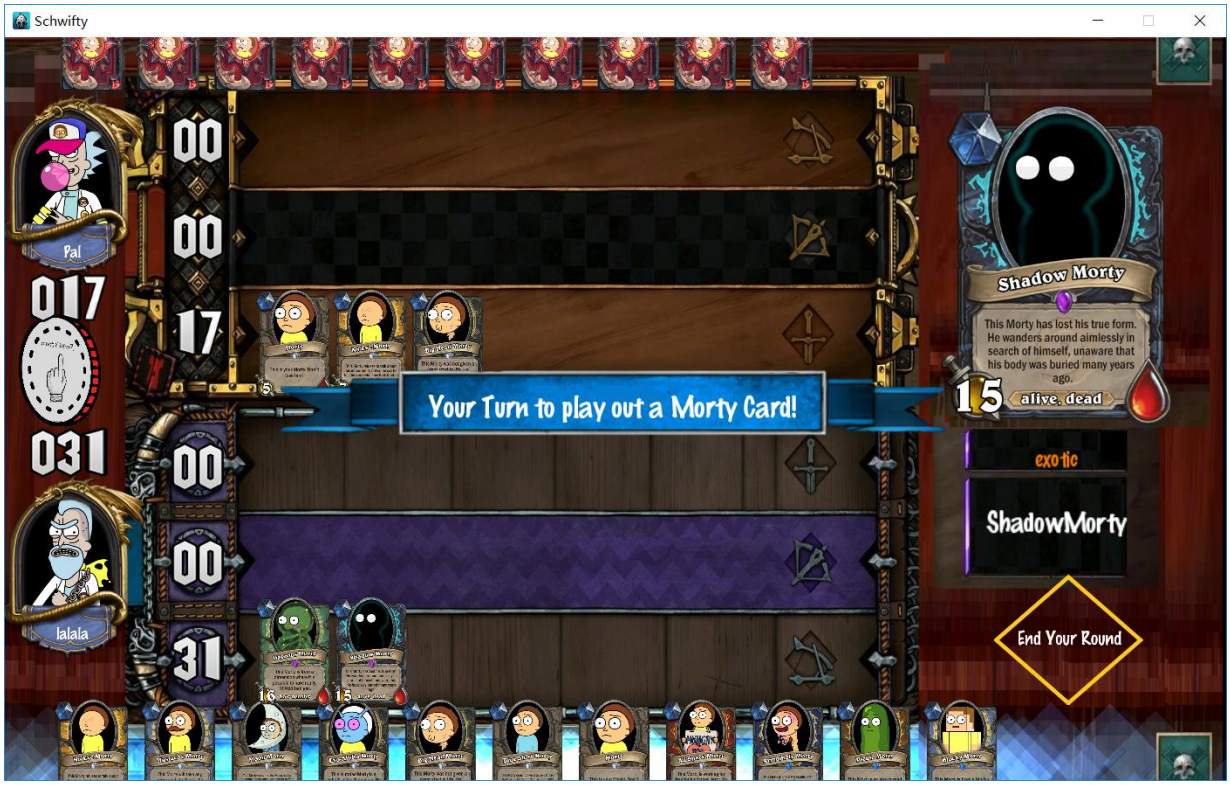
8．双方同意后，可进入牌局

9. 牌局界面：左边一栏为角色以及总分对比，下方为己方手牌，上方为对方手牌，中间每个用户各有三行可出牌位置，相同稀有程度的牌只能出在同一行。在出牌区右方显示的是现在点击的牌的详情页面，同时点击“End Your Round”可以结束这一局（三局两胜制）

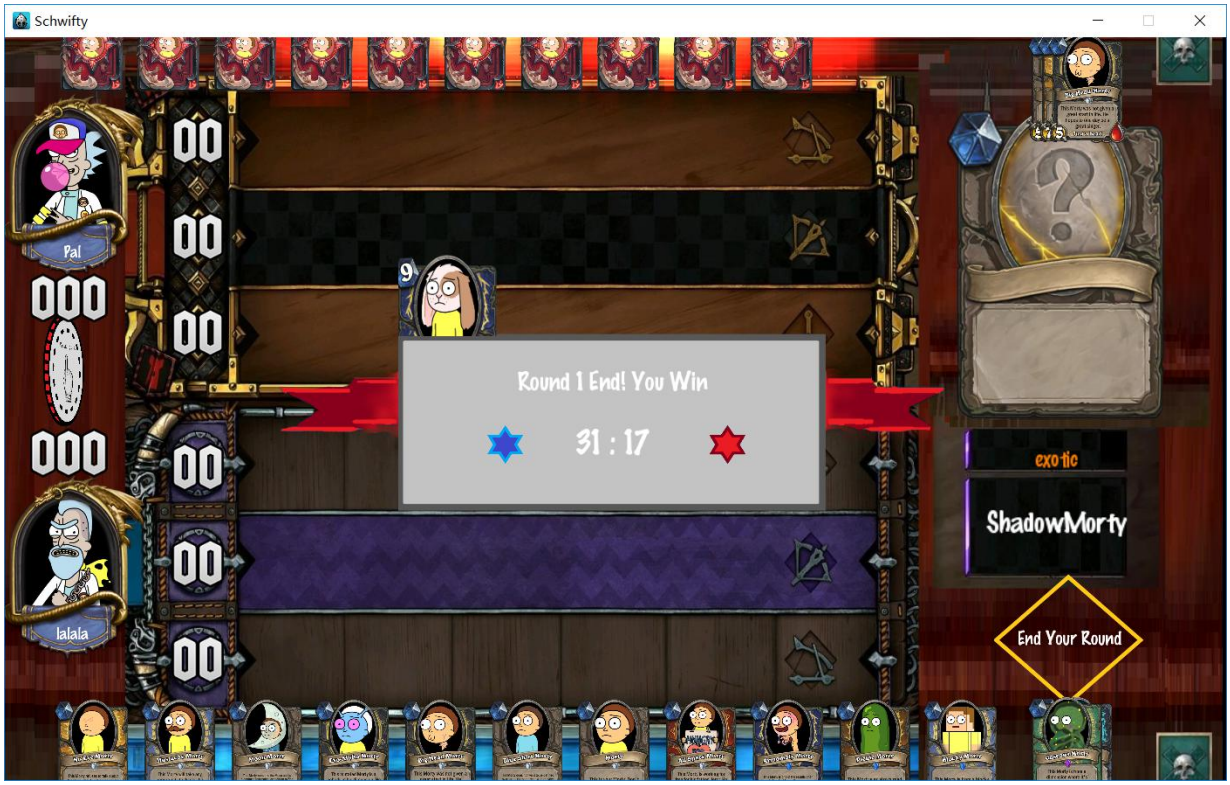
10. 点击“End Your Round”：



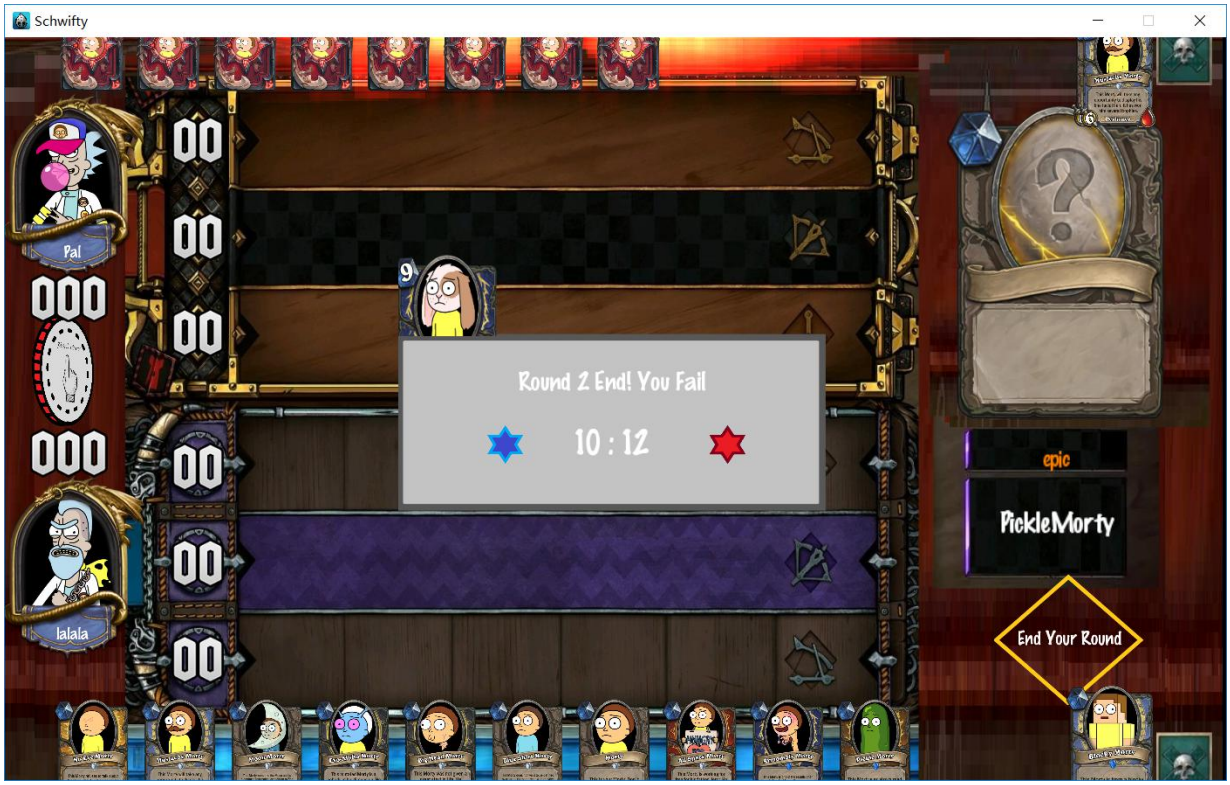
11. 轮到己方出牌：



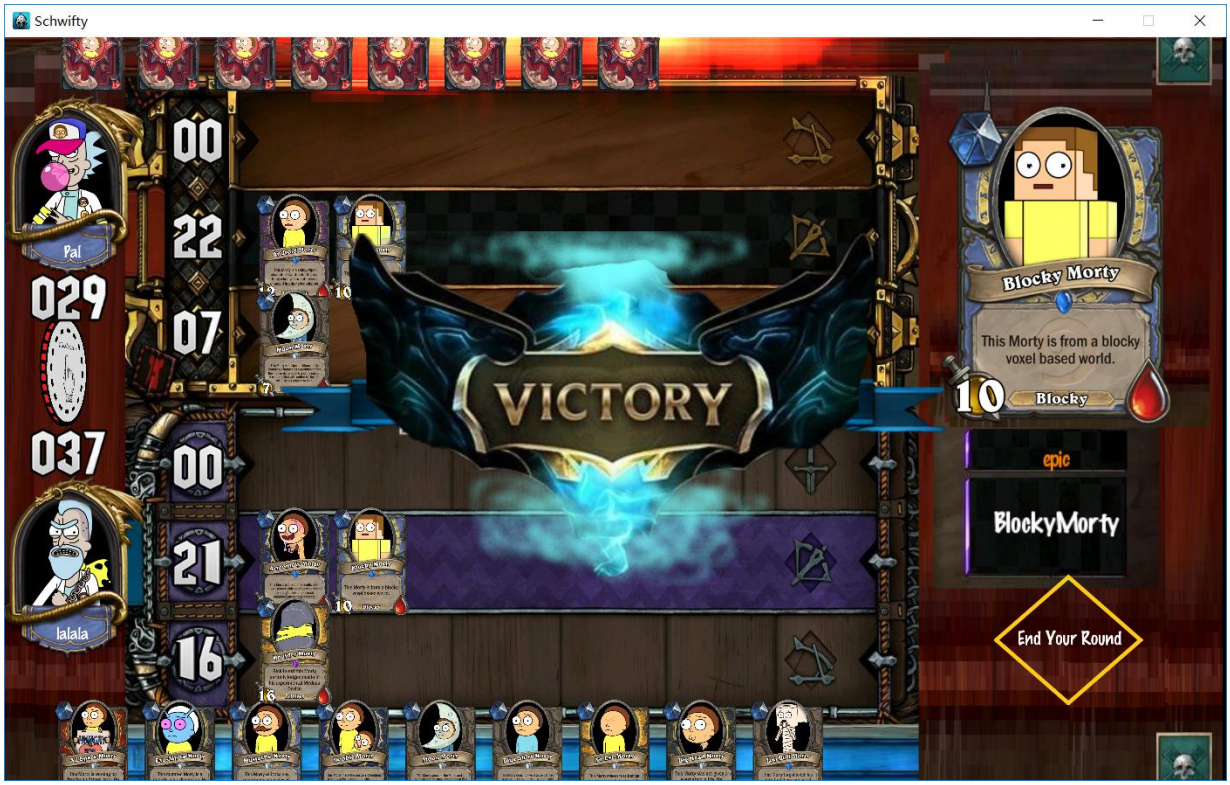
12. 赢得一局：



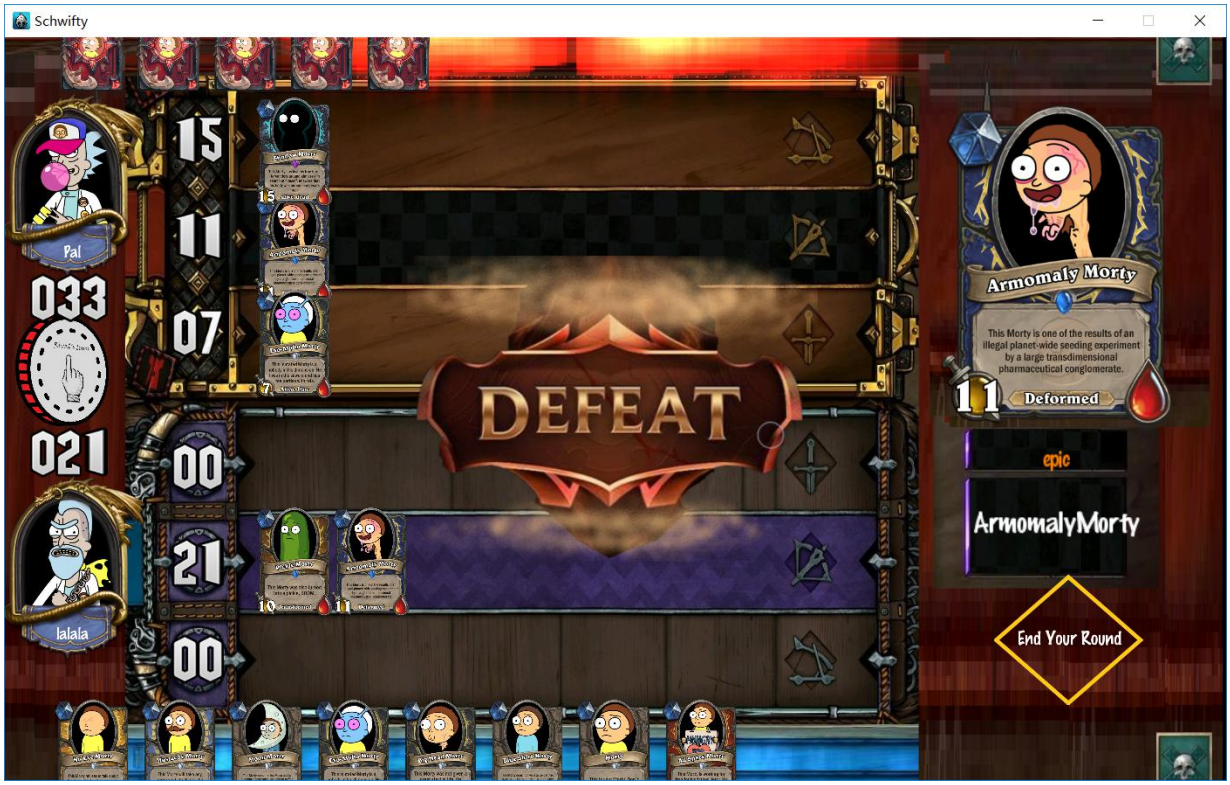
13. 输掉一局：



14. 赢得两局取得胜利：



15． 输给对手：



五、 项目难点及解决方案

1. Cocos 游戏前端实现登录 UI

问题描述：对于 Cocos2d 游戏开发者来说，编写小窗口内的登录界面实际上还是比较难做出选择的一件事情 -- 想要能够在同一个界面内实现好看轻巧的 UI，切换界面显得冗余麻烦。考虑到弹窗的轻便性，实际上将登录界面写成弹出 Layer 会更好一些，同时也需要结合代码的可重用性。

解决方法：手动实现一个 Dialog 类，继承于 Layer，弹出效果自定义，Dialog 上的输入框和按钮，可以通过 chooseMode 函数进行调整，不仅仅解决了登录界面的问题，实际上还为用户之间的邀请，接受请求界面的实现提供了可重用代码。

2. 多角色在线共享位置变化的难题

问题描述：作为多人在线游戏，游戏制作中遇到的难题主要是其他用户角色的走动需

要更新当前用户的游戏界面，我们同样是使用 WebSocket 作为实时传输数据的桥梁，但我们希望的每个用户游戏界面上其他用户的移动，均能够显示出动画效果，以及其 Rick 的不同类型。

解决方法：使用 WebSocket 传输每个用户移动之后的位置，并建立缓冲区，每一次的其他用户更新位置行为会与其之前的位置进行比较，达到了前端处理显示不同方向动画的效果。Rick 类型采用数据库存取，方便显示。

3. 卡牌游戏的逻辑实现和前后端 Debug

问题描述：卡牌游戏的实现难点就在于游戏逻辑的建立，以及前后端沟通“协议”的规定，以及对 WebSocket 消息的发送，接收相关信息后做出的行为。

解决方法：我们将前后端之间的信息交流以一定的格式建立，如“userLogin||Hello”，是服务端告诉全体玩家，新用户 Hello 登入；“play&&8”是玩家发送给服务器，表示他打出了 ID 为 8 的牌。如此约束交换的信息，使得前后端都很容易解析并获取相关内容。

4. 服务端的 WebSocket 对个体用户的单独传输数据

问题描述：游戏中的数据是没有必要对任何玩家都传输的，不仅没有必要，还会导致服务器工作量加大，泄漏信息。如此一来，服务端对单独玩家的信息传输就显得很有必要。

解决方法：在每个玩家登入游戏之后，服务端会维护一个将用户名与 connection 号相映射的 Map。每当遇到只能对特殊用户传输的数据时，就用其用户名作为 key，拿到其所在的 connection 发送消息。

六、 项目总结

这次项目，从构思，分工，到实现，整合，测试，修复 bug，再测试，继续修复 bug 的循环中暂时告一段落，项目完成过程中有遇到各种各样的问题，前端和后端各自出现的 Bug，在各自完成对应部分，相互协助修复后进行合并，又会出现其他新的问题。开始测试，断点调试，确保自己负责部分没问题之后定位错误在谁负责的哪个位置发生，查找相应资料进行解决，主要是发生在前端 UI 和后端代码的衔接，客户端与服务端之间的问题。但总算是都逐一解决了。

构思部分:我们要做一个怎样的游戏，要有什么样的功能，满足怎样的需求？组内的核心成员们都是《Rick And Morty》的粉丝，对卡牌游戏也是情有独钟。经过一番思索，最终决定做一个轻量级的多人在线卡牌对战游戏，操作简单，但游戏又不失趣味性。

实现部分：有了 idea，要怎样去实现？要考虑卡牌游戏最重要的游戏逻辑和玩法机制，此处主要是服务端的组员负责构建核心玩法，组内的其他成员负责提供建议，Cocos 端组员负责磨合和实现。大家齐心协力，虽然在多人联网，卡牌对战方面遇到了很多 Bug，但经过努力探索，以及不停地断点调试，github 上的 push 和 pull，我们最后也是成功地做出了合格的成品。

后期完善和修改：做出大致功能后要考虑对应的细节，比如界面的美观性问题，要在原来的基础上进行一定的调整，同时进行不同的测试，不同人测试的结果往往不一样，所以往往会发现不同的问题，再进一步完善。

总而言之，这个项目是团队齐心协力的结果。