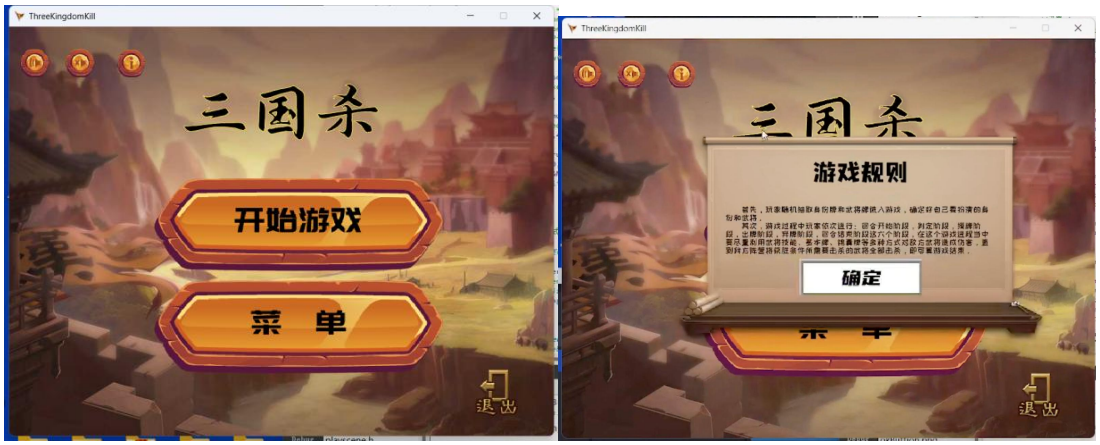
QT 大作业报告

yyy 队

程序功能介绍

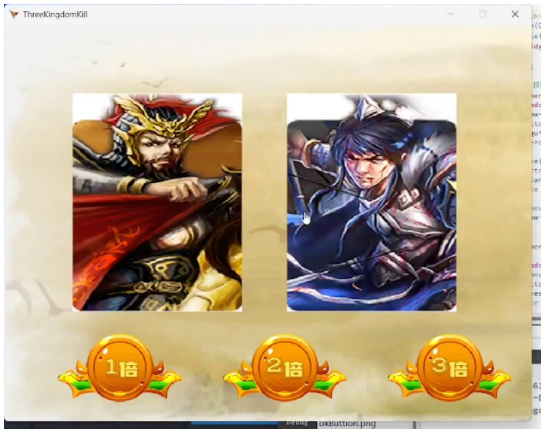
主菜单

- **查看游戏规则**：点击按钮可查看游戏规则。
- **游戏音效**：解包三国杀获取原声大碟，提供至尊体验。选将、游戏开始、背景音乐、出牌、阵亡、游戏结束都有配音，有那味了！
- **静音**：你也可以选择折断歌王的那支麦克风。



游戏主进程

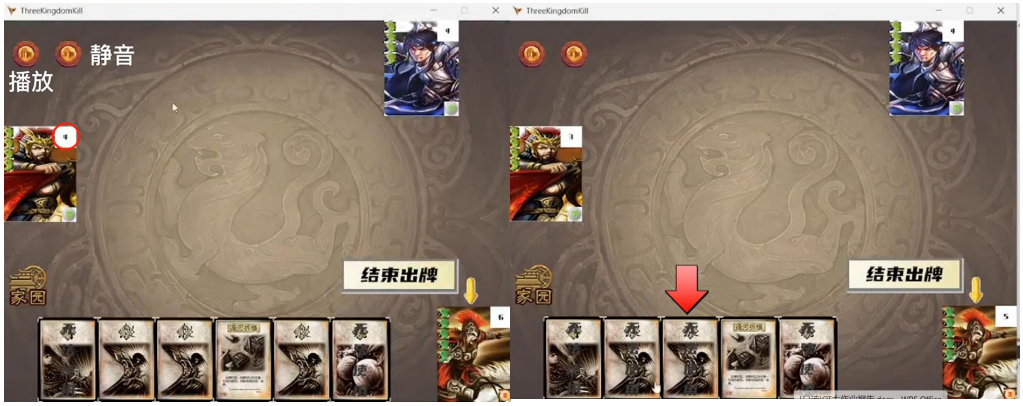
- **游戏规则**：三人对局，一地主两农民，一玩家两 AI。地主死亡农民胜利，农民全部死亡地主胜利。
- **选择武将**：将池中有五名武将，随机抽取两名武将供玩家选择。AI 在将池中随机选择。
- **叫地主**：与传统斗地主类似，叫 3 倍自动成为地主。AI 根据武将强力值（内置）决定倍数。强力值由三国杀官方斗地主胜率决定。



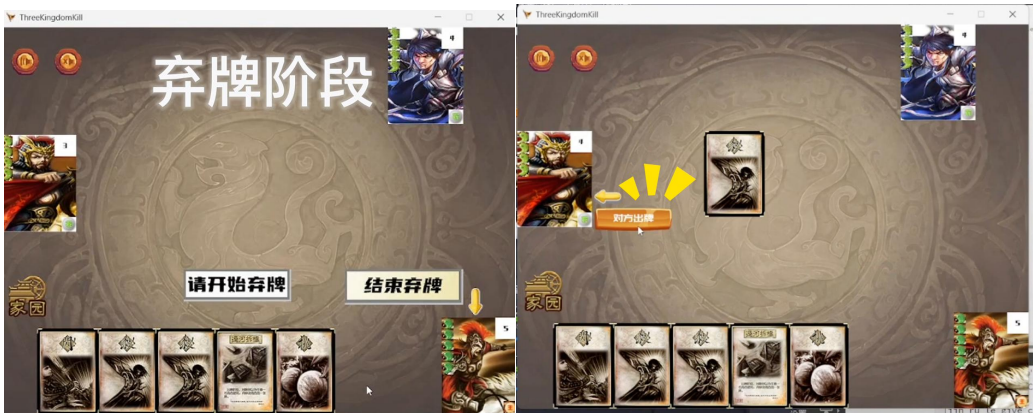
❖ **<u>玩家使用牌、弃牌、响应牌</u>**：基本还原三国杀原版体验。鼠标选中牌，如需要并选择使用对象。详见“卡牌实现”部分。

❖ **<u>手牌显示</u>**：完美复刻原版。

AI 武将图像右上角**只展示手牌数**，玩家牌组中会显示牌名、牌面。玩家出牌阶段，部分手牌会有“无法使用”标记，此时该牌无法被选中。

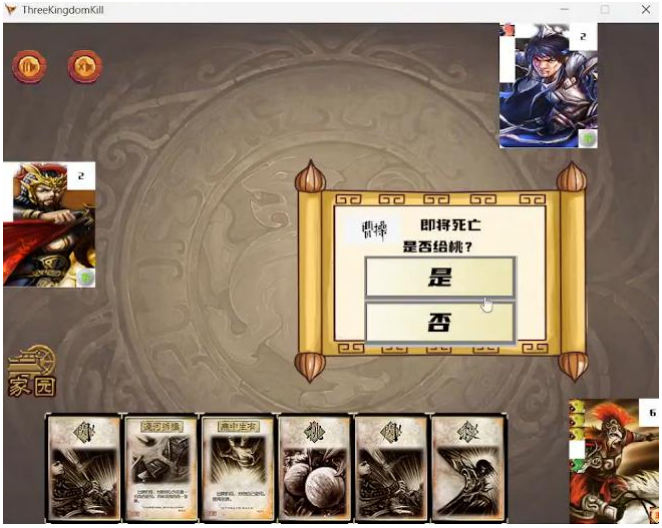


玩家因出牌、弃牌导致手牌数变动时，牌组会自动移动卡牌位置，重新整理手牌。进入弃牌堆的卡牌会在屏幕中央展示；玩家回合结束后，主界面会显示当前出牌角色。

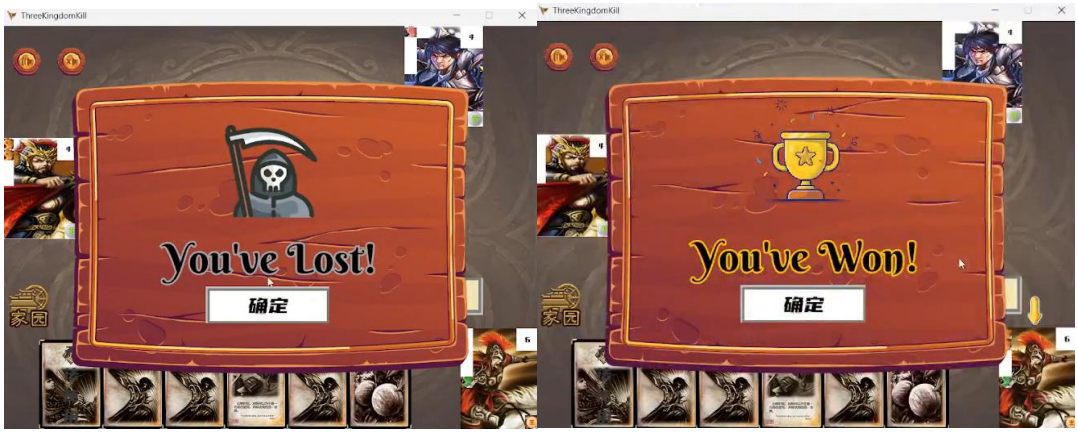


- **血量显示：**血量健康为绿色、亚健康为黄色、一血为红色。显示在武将牌左上角。
- **主农身份显示；当前出牌者显示**

- **AI：**以特判方式实现出牌、弃牌。出牌时能够分辨敌友，对敌人使用伤害牌；弃牌时优先保留防御牌。
- **濒死求桃：**复刻武将进入濒死阶段后顺时针求桃的场景，同时第一位农民阵亡，另一位农民可选择摸两张牌或回一滴血。



❖ **结算界面:** 设立胜利图像，播放相关音乐



卡牌实现

❖ **基本牌**:

闪、桃已实现使用逻辑，出牌音效；

选中杀后，会跳出按钮，提示玩家选择出杀对象。

❖ **锦囊牌**: 【有待进一步扩充】

-- 南蛮入侵、万箭齐发、桃园结义等 AOE 锦囊：复用 Warrior 类的 be_assaulted,be_slashed,be_peached 函数。

```
//以下是playscene的成员函数
void playscene::SavageAssaultAction(){
    for(int i=1;i<3;i++){
            int cur_idx = (cur_player_idx+i)%3;
            if(players[cur_idx] && players[cur_idx]->alive)
                players[cur_idx]->be_assaulted();
        }
}
void playscene::ArrowRainAction(){
    for(int i=1;i<3;i++){//cur_player不应该被自己的AOE攻击到
            int cur_idx = (cur_player_idx+i)%3;
            if(players[cur_idx] && players[cur_idx]->alive)
                players[cur_idx]->be_slashed(players[cur_player_idx]);//owner即为伤害来源
        }
}
```

-- 乐不思蜀、闪电、兵粮寸断：由于现有 UI 界面存放判定区会破坏美感，暂时将判定区移入后端。已实现后端逻辑，延时锦囊牌使用后自动置入判定区。判定区有牌的武将，经过后台判定后，决定是否跳过相应阶段。

```
void HappinessDrown::Action(Warrior * enemy){
    //ask for target
    Warrior * target;
    if(!target->fate_tell_zone[1])//对方判定区没有乐不思蜀，则：
        target->fate_tell_zone[1] = this;//把乐不思蜀置入目标的判定区
}
SupplyShortage::SupplyShortage(QString f,bool hide, QWidget* parent
void SupplyShortage::Action(Warrior * enemy){
    //ask for target
    Warrior * target;
    if(!target->fate_tell_zone[0])//对方判定区没有乐不思蜀，则：
        target->fate_tell_zone[0] = this;//把乐不思蜀置入目标的判定区
}
```

武将实现

❖ 图像、血量、手牌数、身份、牌组实现

❖ 技能后端实现：

张飞咆哮：出杀无限制；马超铁骑：出杀，判定为红则无法闪避

```
ZhangFei::ZhangFei(int player, QWidget *parent):Warrior("ZhangFei", 4, player, parent){//张飞类
    name = "ZhangFei";
    hp = 4;
    totalhp = 4;
    slash_limit = 100;
    talent.push_back(new Roaring(this));
}
MaChao::MaChao(int player, QWidget *parent):Warrior("MaChao", 4, player, parent){//马超类
    name = "MaChao";
    hp = 4;
    totalhp = 4;
    slash_limit = 1;
    talent.push_back(new IronHorse(this));
}
```

美工、音效：华丽的视听效果实现

- **音效**：来源于解包三国杀，并巧妙嵌入后端逻辑
- **美工：**UI 中确认、取消、退出等全部按钮，静音、箭头、主农等所有图标，杀闪桃、无法使用、出牌弃牌等一切指示，游戏规则、退出确认、输赢窗口等界面背景，全部来源于组员 PS 手工制作

正在开发的功能

我们考虑到程序应具有良好的可扩展性，

因此列出以下容易扩充的功能

- **武将主动技能**：Warrior 类有 Talent * 成员变量，存储武将技能。复用手牌是否可使用判定，点击按钮、选择对象，发动技能。
- **拆、顺**：复用“对 ta 使用”按钮以及玩家的牌组显示系列函数，使得玩家可以选中 AI 手牌进行“过河拆桥”。
- **判定区可视化**：调整 UI 界面现有按钮占比，为判定区挪空间。
- **托管**：复用 AI 逻辑，实现自动出牌、弃牌、响应。摸鱼专用！

- **游戏结算**：记录武将伤害、治疗分数，结算局势分数统计
- **将池扩容：**更多武将，更多技能。
- **更多玩家：**playscene 使用 players 数组存放参与者，用 cur_player_idx 跟踪当前出牌者。优秀的后端设计使得加入更多玩家变得非常容易。

项目各模块与类设计细节

菜单

- **mymenu**：主菜单，配备 bgm，信号-槽机制实现按钮交互

```
signals:
    void readyforquit(); //单击退出游戏的信号
    void helpclicked(); //单击帮助菜单的信号
    void optionclicked(); //单击选项菜单的信号
    void chooseperson(); //
    void chooseMultiple();//单击选人界面（开始游戏）的信号
```

- **choosemenu**：选将界面，配备 bgm，向 playscene 传参
- **choose_multiple**：倍数界面，配备 bgm

```
//倍数1按钮设置
MyPushButton* Multiple1=new MyPushButton(this,true,":/menu/res/Multiple1.png");
Multiple1->move(125,125); //这是该按钮在窗口所处的位置
connect(Multiple1,&MyPushButton::clicked,[=](){
    bgm->stop();
    emit chooseMultiple1();
});
//倍数2按钮设置
MyPushButton* Multiple2=new MyPushButton(this,true,":/menu/res/Multiple2.png");
Multiple2->move(320,125); //这是该按钮在窗口所处的位置
connect(Multiple2,&MyPushButton::clicked,[=](){
    bgm->stop();
    emit chooseMultiple2();
});
//倍数2按钮设置
MyPushButton* Multiple3=new MyPushButton(this,true,":/menu/res/Multiple3.png");
Multiple3->move(515,125); //这是该按钮在窗口所处的位置
connect(Multiple3,&MyPushButton::clicked,[=](){
    bgm->stop();
    emit chooseMultiple3();
});
```

- **playscene**：游戏主进程，配备 bgm，负责出牌、弃牌、响应逻辑、角色死亡检查、局面胜负检查

sethero 函数用于初始化 players[3]成员变量

---

setcardshop 函数显示玩家牌组

connect-emit 机制实现出牌->弃牌->下一玩家出牌的游戏进程

throwcard,delcard 函数管理玩家和 AI 在出牌、弃牌后的手牌显示

draw_cards 等其他成员函数负责发牌、模拟 AOE 锦囊的实现

❖ **AIDecision**：AI 决策，决定 AI 选将、抢地主、出牌弃牌逻辑

```
class AIDecision{
public:
    //询问是否使用技能
    static void be_asked_use_talent(Warrior* self,Warrior* p1,Warrior* p2);
    //被求桃 true给 false不给
    static bool be_asked_for_peach(Warrior* self,Warrior* other);
    //要“闪” true出 false不出
    static bool be_asked_for_dodge(Warrior* self);
    //要"杀"（如别人出南蛮入侵时会被调用）true出 false不出
    static bool be_asked_for_slash(Warrior* self);
    static int if_want_to_be_landlords(Warrior* self);
    //弃牌
    static void give_up_cards(Warrior* self);
    //出牌
    static void give_cards(Warrior* self,Warrior* p1,Warrior* p2);
    //返回任意一个敌人
    static Warrior* getEnemy(Warrior* self,Warrior* p1,Warrior* p2);
    //返回任意一个队友
    static Warrior *getFriend(Warrior *self, Warrior *p1, Warrior *p2);

    static void stop(int time = 3);
```

❖ **Warrior**：武将类

通过 be_slashed/be_peached/sleepforever 等函数存放音效并实现出牌效果、推进游戏进程

通过 flush_card 函数在出牌、弃牌前后刷新牌组状态，模拟手牌显示

❖ **Cards**：卡牌类

不同卡牌均继承于 Cards，以 Action 为虚函数重载不同功能。

卡牌记录 owner 和 target 两个 Warrior 类变量，调用 Warrior 的响应函数实现交互。

❖ **endscene**：结算界面，设立胜利图像，播放相关音乐

小组成员分工

❖ **姚懿迅 2200013132**

AI 逻辑设计: AIDecision 类

C++框架向 QT 转换

QButton 设计：图片素材添加、交互逻辑设计

弹窗设计：各类弹窗的添加、弹窗控件功能设计、位置调整

QT 界面逻辑：选将逻辑、选择倍数逻辑、游戏主界面显示

所有函数的总调试（debug 的神）

- **丁昱菲 2200013098**

代码部分：choose_multiple, choosemenu, endscene,mainwindow, mymenu,mypushbutton,shapedwindow 初稿；QT 的 ui 文件及界面设计；playscene 中 winning 等界面函数，动画及音频实现

素材部分：使用 ps 等设计软件 res 中大部分图片的绘画制作，包括确认、取消、退出等按钮，静音、箭头、主农等图标，杀闪桃、无法使用、出牌弃牌等指示，游戏规则、退出确认、输赢窗口等界面背景

- **徐灵昀 2200013215**

代码部分：Warrior 类、Card 类、Talent 类初稿(C++)；playscene.cpp: 游戏主进程后端信号-槽逻辑设计；牌与武将的交互：吃桃、出杀、被杀等逻辑

素材部分：原游戏背景图、所有音频提供：解包+格式转换

项目总结与反思

- **困难 I：规划失误**

项目初始时，我们的分工设想为：徐灵昀负责主流程部分代码，姚懿迅负责 AI 策略部分代码，丁昱菲负责 QT 及 ui。由于我们对 QT 特

殊语法的缺乏了解，最初的流程代码以 c++的模式书写，而无法直接嵌入 QT 框架，也间接导致了后期 debug 困难的陡增。幸而我们较早地发现了这个问题，后期着重针对 playscene 进行了大力修改，最终实现了全流程的搭建。

❖ **困难 II：环境问题**

由于 QT 不同版本之间及 QT5 与 macOS 系统的不兼容性，我们三位成员中仅有一位电脑可以正常编译运行程序，这给我们的合作带来了极大的困难。为了应对这一问题，我们改变了先前按功能划分的分工模式，转而采取“接力”的方法，绘图、ui 框架、代码 debug 多管齐下，大大提高了效率。

❖ **困难 III：内部逻辑复杂**

三国杀本身规则繁琐且包含大量的特殊判定，对于代码书写的仔细度和整体性提出很高的要求。我们在过程中遇到的 bug 包括但不限于：

1.各类初始化问题 2.函数的相互调用 3.判定条件遗漏

4.信号与槽之间的无效对接

❖ **总结**

编程路漫漫，QT 是一座里程碑，而我们将继续上下求索！