2023年11月13日

C++ 项目报告

SCHOTTEN-TOTTEN

ZHAO ZHENYU
ZHANG LINGYU
YANG ZIJIE
ZHANG YIFAN

课程报告

目录

一、项目概况

本小组成功设计出游戏的经典版本和战术版本，并用QT实现了游戏经典版本的交互页面，完成了项目中的大部分要求，能够实行两名玩家通过各自对操作台的控制，自由进行对抗，同时还可以选择一名玩家与人机进行游戏，在经典对战的基础上，游戏还引入了一种“战术”变体，其中包含Joker小丑卡，Spy间谍，Shield-Beare持盾者，Blind-Man's Bluff盲人牌局，Mud Fight泥斗，Recruiter招募者，Strategis战略家，Banshee女妖，Traiter叛徒等十张战术牌的独立牌组。玩家被允许手中持有七张牌，并且在打出一张牌并抽牌时，可以选择从战术牌组中抽取。战术牌会以某种方式改变游戏规则。玩家只能打出比对手多一张战术牌。游戏的胜利条件保持不变。稍有瑕疵的是，在QT实现方面由于没有经验，只设计了经典对战的页面交互，完整游戏只能通过c++代码的终端进行，需要改进。

二、项目流程

1.小组成员搜寻资料，熟悉游戏玩法

2.设计游戏战术变体

3.构想程序大致框架

4.设计类，画出UML图

5.整合各成员编程内容，完善润色代码

6.学习QT相关知识，将代码部分整合至QT

7.撰写项目说明书

三、代码实现

1、Card

Card
+ showCard(): void + showDisCard(): void

Card类是一个抽象基类，表示一张卡片

showCard()和showDisCard()，分别用于显示卡片和弃牌。

vector<Card*>是一个存储Card对象指针的向量。

vector<ClanCard*>是一个存储ClanCard对象指针的向量。

2、ClanCard

ClanCard
- number: int - color: string
+ ClanCard() + ClanCard(number: int, color: const string&) + getNumber(): int + getColor(): string

ClanCard类是一个表示氏族卡的类，它是从Card类派生而来。它具有存储氏族卡的数字和颜色的属性，并提供了获取和设置这些属性的方法。ClanCard类还提供了展示氏族卡信息的功能，包括展示卡片内容和被弃置时的信息。此外，还有一些辅助函数用于操作和展示存储氏族卡的向量。ClanCard类在游戏中扮演着管理和展示氏族卡的角色，为玩家提供了更好的游戏体验和互动。通过ClanCard类，玩家可以轻松地操作和管理氏族卡，了解卡片的属性和状态，以及进行相关的游戏决策。

3、ClanGame

ClanGame
- cards: const Card*[54]
+ getCard(i: size_t): const Card& + getNbCards(): size_t + getIterator(): Iterator + ClanGame() + ~ClanGame() + ClanGame(const ClanGame&) = delete + operator=(const ClanGame&) = delete + static getClanGame(): ClanGame& + static freeClanGame(): void + isDone(): bool + next(): void + currentItem(): const Card&

ClanGame类是一个表示氏族游戏的类，它提供了管理和操作游戏中的卡牌的功能。该类包含了卡牌的指针数组和一个内部结构体Processor，用于处理游戏逻辑。ClanGame类还定义了一个内部类Iterator，用于遍历卡牌。通过ClanGame类，玩家可以方便地获取游戏中的卡牌，并进行遍历和操作。该类还提供了获取卡牌数量的方法，以及获取ClanGame对象的引用和释放对象的静态方法。

4、TecGame

TecGame
- m_name: const string
+ TecGame() + ~TecGame() + getCard(size_t): const Card& + getNbCards(): size_t + getIterator(): Iterator + Iterator() + isDone(): bool + next(): void + currentItem(): const Card&

TecGame类是一个表示战术游戏的类。它包含了存储卡牌的指针数组和一个内部结构体Processor，用于处理游戏逻辑。TecGame类还定义了一个内部类Iterator，用于遍历卡牌。通过TecGame类，可以获取卡牌、遍历卡牌以及管理战术游戏的操作。

5、TecCard

TecCard
- m_name: const string
+ TecCard(const string&) + getName(): const string& + showCard(): void + showDisCard(): void

TecCard类是一个表示战术卡的类，它继承自Card类。该类具有存储战术卡名称的属性，并提供了获取名称和展示卡片信息的方法。该类在游戏中扮演着管理和展示战术卡的角色，为玩家提供了更好的游戏体验和互动。通过TecCard类，玩家可以更好地掌握战术卡的属性和功能。

6、BattleCard

BattleCard
- name: string
+ BattleCard(const string&) + play_BlindBluff(Stone*) + play_MudFight(Stone*) + showCard() const + showDisCard() const

BattleCard类是一个表示战斗卡的类，它是从TecCard类派生而来。它提供了几个公有静态成员函数来定义战斗卡

7、CunningCard

CunningCard
- name: string
+ CunningCard() + Play_Recruiter() + Play_Strategis() + Play_Banshee() + Play_Traiter() + Play_Recruiter2() + Play_Strategis2() + Play_Banshee2() + Play_Traiter2() + Play_RecruiterAI() + Play_StrategisAI() + Play_BansheeAI() + Play_TraiterAI() + ShowCard() + ShowDisCard()

CunningCard类是一个表示诡计卡的类，它是从TecCard类派生而来。它提供了几个公有静态成员函数来定义诡计卡

8、CardEliteGroup

CardEliteGroup
- name: string
+ CardEliteGroup(name: string，number: Number, color: Color) + getName(): string + player_Joker(): CardEliteGroup + player_JokerAI(): CardEliteGroup + player_ShieldBeare(): CardEliteGroup + player_ShieldBeareAI(): CardEliteGroup + player_Spy(): CardEliteGroup + player_SpyAI(): CardEliteGroup + ShowCard(): void + ShowDisCard(): void

CardEliteGroup类是一个表示精英卡的类，它继承自ClanCard类。该类具有存储精英卡名称的属性，并提供了获取名称和展示卡片信息的方法。CardEliteGroup类还提供了玩家选择不同精英卡的操作，包括选择颜色和强度的功能。该类还重写了展示精英卡信息和被弃置时信息的方法。

9、Hand

Hand
- m_cards: vector<Card*> - m_nb_tec_card_played: size_t - m_nb: int - m_joker: bool
+ getNbCards(): size_t + getNbTecCardPlayed(): size_t + add(Card*) + remove(size_t) + getCards(): vector<Card*> + getCard(size_t): Card* + getJoker(): bool

Hand类是一个用于管理手牌的类。它可以存储卡牌的指针向量，并提供了添加、移除和获取手牌的方法。Hand类还记录了已经打出的科技卡牌数量和是否有Joker牌。通过Hand类，玩家可以方便地管理自己的手牌，进行卡牌的增加、移除和查看操作。同时，Hand类还提供了获取手牌数量和已打出科技卡牌数量的功能，方便玩家进行游戏策略的决策和判断。

10、Stone

Stone
- m_num: int - m_revendique: int - m_cardp1: Combination* - m_cardp2: Combination*
+ Stone(int, bool, bool) + getnum(): int + setnum(int): void + getrevendique(): int + setrevendique(int): int + getcardp1(): Combination* + getcardp2(): Combination* + setBlindBluff(bool): void + setMudFight(bool): void + add_Card(ClanCard*): void

Stone类是用于表示石头的类。它包含了石头的编号、归属权以及玩家在该石头上的卡牌组合的信息。Stone类提供了一些方法来获取和设置石头的属性，以及向石头上添加卡牌。它在游戏中起到管理石头状态和相关操作的作用，为游戏提供了石头的基本功能和特性。通过Stone类，玩家可以在游戏中操作和管理石头，进行策略和决策，为游戏增添了更多的变数和乐趣。

11、Table

Table
- player1: Player* - player2: Player* - m_stones: Stone** - m_current_player: int - discards: vector<Card*>
+ Table(size_t, bool, bool) + getPlayer1(): Player* + getPlayer2(): Player* + getCurrentPlayer(): int + getDiscards(): vector<Card*> + getStones(int): Stone* + setCurrentPlayer(int): void + put(Stone, ClanCard*): void + showPlayerHand(int): void + showHandBack(int): void + showStones(int): void + showTable(int): void

Table 类是一个用于管理玩家、卡片和石头等游戏元素的类，提供了一些用于展示和处理游戏状态的成员函数。

12、Player

Player
- m_hand: Hand* - m_is_ai: bool - Joker: bool
+ Player(size_t, bool) + getHand(): Hand* + getIsAI(): bool

Player类用于表示游戏中的玩家，可以通过getHand()函数获取玩家的手牌对象，并通过getIsAI()函数获取玩家是否是AI玩家的信息。

13、SetException

SetException
- info: string
+ SetException(const string&) + getInfo(): string

SetException类用于在集合操作中抛出异常，并提供了获取异常信息的方法。

14、Rest

Rest
- m_cards: const Card** - m_nb: size_t
+ Rest(const ClanGame&) + Rest(const TecGame&) + ~Rest() + setM_nb(size_t) + isEmpty(): bool + getNbCards(): size_t + getCards(): const Card**& + draw(): const Card& + drawClanCard(): const ClanCard& + drawTecCard(): const TecCard& + drawCardEliteGroup(): const CardEliteGroup& + whendrawpayattention(): int + push_back(const Card*) + push_back2(const Card*) + getRandomInt(int, int): int

Rest类，一个用于管理卡牌剩余的类。它可以存储卡牌的指针数组，并跟踪剩余卡牌的数量。Rest类提供了从剩余卡牌中抽取卡牌的功能，并可以根据不同类型的卡牌进行抽取。此外，它还可以将新的卡牌添加到剩余卡牌中。Rest类还包含一个全局函数，用于生成指定范围内的随机整数。

15、SetCard

SetCard
- m_info: string
+ SetCard(const string&) + getInfo(): string + toString(Color): string + toString(Number): string + operator<<(ostream&, Color): ostream& + operator<<(ostream&, Number): ostream& - Colors: initializer_list<Color> - Numbers: initializer_list<Number

SetCard类是一个表示集合卡的类。它具有一个存储卡片信息的私有成员变量m_info，并提供了构造函数和获取卡片信息的方法。Color枚举类表示卡片的颜色，Number枚举类表示卡片的数字。这两个枚举类用于表示集合卡的属性。toString函数和ostream重载运算符用于将枚举值转换为可读的字符串形式，方便输出和显示。initializer_list用于初始化枚举值的列表，方便进行初始化操作。这些功能的组合使得SetCard类能够方便地表示和操作集合卡的属性和信息。

16、Combination

Combination	
- m_force_combi: int - m_total_number: int - m_mudfight: bool - m_blindbluff: bool - m_first_completed: int - m_cards: vector<ClanCard*>	
+ Combination(cards: vector<ClanCard*>) + DeleteCard_Stone(pos: int) + getCards(): vector<ClanCard*> + getForceCombi(): int + setTotalNumber(total_number: int + setMudFight(mudfight: bool)	+ setBlindBluff(blindbluff: bool) + getFirstCompleted(): int + setFirstCompleted(premier_complet: int) + calculateForceCombi(): int + addCard(card: ClanCard*)

Combination类是一个用于管理卡牌组合的类。它可以计算组合的力量值，并跟踪组合中卡牌的总数。此外，它还可以记录组合是否进行泥斗和盲注，并记录首次完成的卡牌数量。通过添加和删除卡牌，Combination类可以方便地管理卡牌组合。

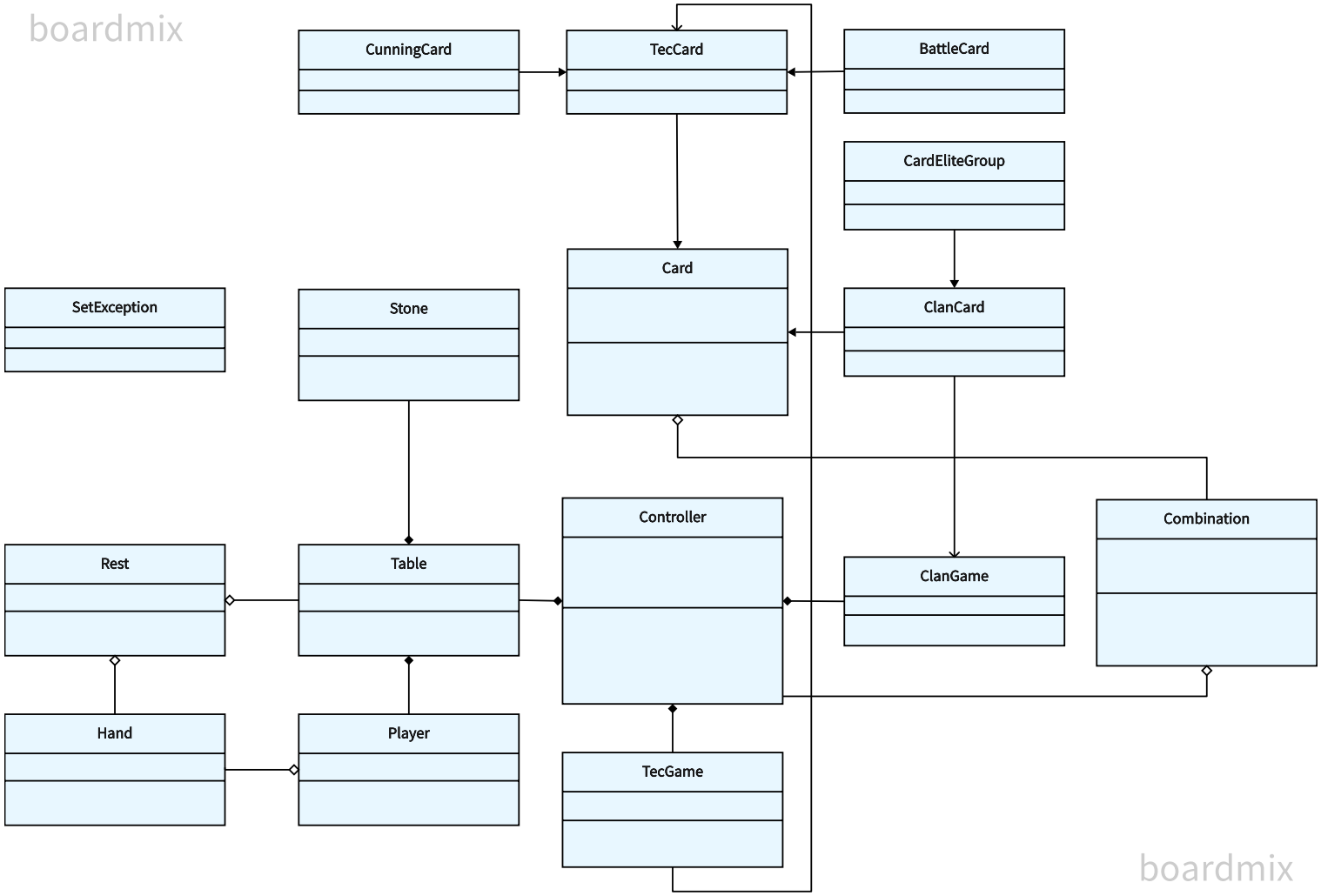
17、Controller

Controller	
- m_winner: int - m_tec: bool - m_clan_deck: Rest* - m_tec_deck: Rest* - m_table: Table* - m_clan_game: ClanGame& - m_tec_game: TecGame& - m_unplayed_cards: vector<ClanCard*>	+ setWinner(winner: int) + PlayTurn1() + PlayTurn2() + PlayClassicAI() + PlayAI() + getClanRest(): Rest* + getTecRest(): Rest* + claim_gate(gate_num: int, human: bool) + startClassicGame()

+ Controller(tec: bool) + ~Controller() + getWinner(): int + getTec(): bool + endGame() + getUnplayedCards(): vector<ClanCard*> + displayUnplayedCards() + Display_Gate1() + Display_Gate2()	+ getTable(): Table* + starttecGame() + test() + checkEndGame(): bool + remove_card_played_v1(card: ClanCard*) + remove_card_played_v2(card: ClanCard*) + PlayClassicTurn1() + PlayClassicTurn2()

Controller类是整个游戏控制器，用于管理游戏的逻辑和状态。它负责跟踪游戏的获胜者和当前玩家是否为Tec玩家，管理Clan玩家和Tec玩家的牌堆，以及跟踪游戏桌面上的卡片。此外，它还提供了一些操作函数，用于处理游戏的不同轮次和模式。Controller类还维护了一个未使用的卡片列表，用于跟踪游戏中尚未使用的卡片。通过提供统一的接口，Controller类使得游戏逻辑更加清晰和易于维护。

四、UML


boardmix
CunningCard
TecCard
BattleCard
CardEliteGroup
Card
SetException
Stone
ClanCard
Controller
Combination
Rest
Table
ClanGame
Hand
Player
TecGame
boardmix

五、优缺点

优点：

1. 采用继承、迭代器设计模式、适配器设计模式等，确保代码的简洁性，提高运行过程中的安全度

2. 引入战术变体牌，较为完整的实现整个游戏的运行

3. 在编写代码的过程中将每一个类放在不同的头文件中处理，便于维护代码

4. 实现玩家与玩家、玩家与人机共同进行游戏，提高游戏的可玩性和趣味性

缺点：

5. 代码编写过程中由于不同人设计格式不尽相同，注释较少，不方便阅读

6. 终端中呈现游戏的页面较为简陋，在QT前段实现的方面进展不足，游戏与玩家的交互不够人性化

7. 考虑采取策略设计模式调用各个函数实现对桌面上卡牌的控制

8. 占用内存过大

六、个人贡献

姓名	学号	负责内容	时长	贡献度
赵振宇	21124653	主编文件： Hand.h Rest.h Table.h Player.h Controller.h Combination.h 其他工作： 撰写报告 编写QT代码 代码测试与反馈 代码优化	40h	35%
张凌宇	21124649	主编文件： Card.h ClanCard.h ClanGame.h BattleCard.h CunningCard.h CardEliteGroup.h Combination.h 其他工作： 编写QT代码 代码测试与反馈 代码优化	40h	35%
杨子杰	21124630	主编文件： TecCard.h TecGame.h Stone.h 其他工作： 撰写报告 代码测试与反馈	25h	15%
张一凡	21124691	主编文件： main.h SetCard.h SetException.h 其他工作： 代码测试与反馈 演示视频制作 部分报告编写	25h	15%