2022-11-19

算法与编程

项目报告

课程报告

骰子街

PENG JIANCHENG、WANG YUXING、SONG PENGYU、LIU DINGWEI、GAO ZIQI

目　录

一、项目概况

在一个学期的学习过程中，我们设计完成了骰子街桌游的简略版本，并完成了项目所提出的要求，实现了选择游戏版本，添加 AI,选择游戏人数等功能。实行 2-4 或 2-5（拓展包版本）名玩家通过对操作台的控制，自由进行对抗。

首先，项目引用了单例模式，保证 class Jue 只能产生一个实例，并面向整个系统提供；

同时，我们引入了牌桌与牌库，在牌桌中生成所有玩家需要的卡牌，保证不同模式下生成完全不相同的卡牌；牌库的存在可以使得我们添加删减卡牌非常的便捷，满足模块化编程的同时，也大大降低了代码的理解难度。

最后，我们为程序进行了拓展，加入了 AI 玩家，并解锁了港口拓展包的新玩法。整个程序可以实现三种玩法：人机对抗、玩家对抗以及 AI 对抗，两种模式：标准版和港口拓展版，都可以由玩家自行选择。

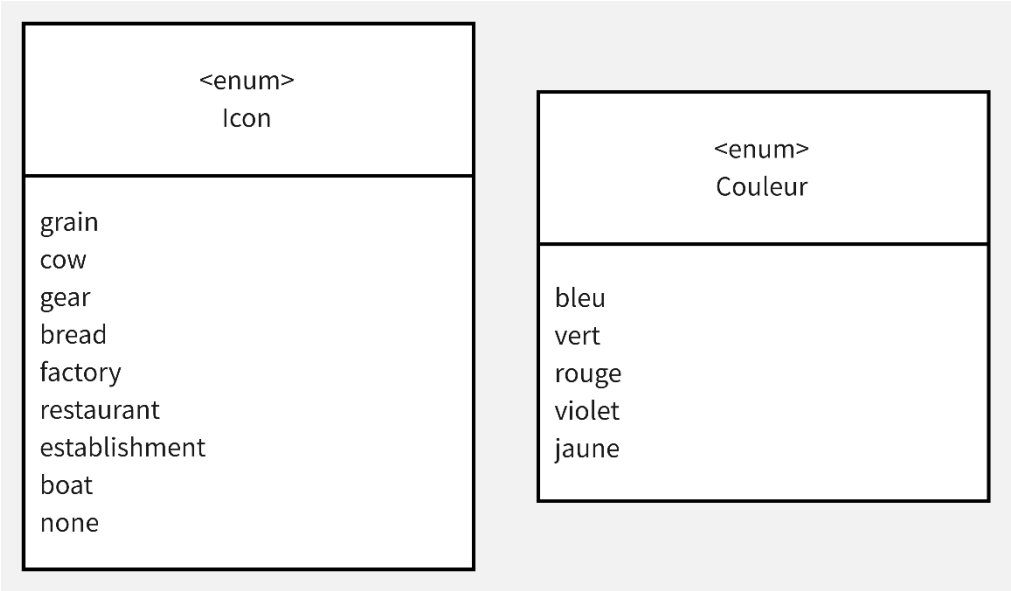
稍有瑕疵的是，目前游戏还没有图形化交互界面，卡牌的显示不够人性化，没有实现存档等功能。

二、项目流程

1.小组成员了解游戏内容与规则，游玩骰子街，加深对规则等的了解
2.熟悉桌游之后构想程序的大致框架（所要实现的功能对应的类、所需分配的文件与头文件、如何实现模块化编程等）
3.分配小组任务，小组成员各司其职负责本职的编译内容。
4.整合小组编程内容，对漏洞进行修复与补充原本缺少的功能。
5.实现游戏代码运行

三、代码实现

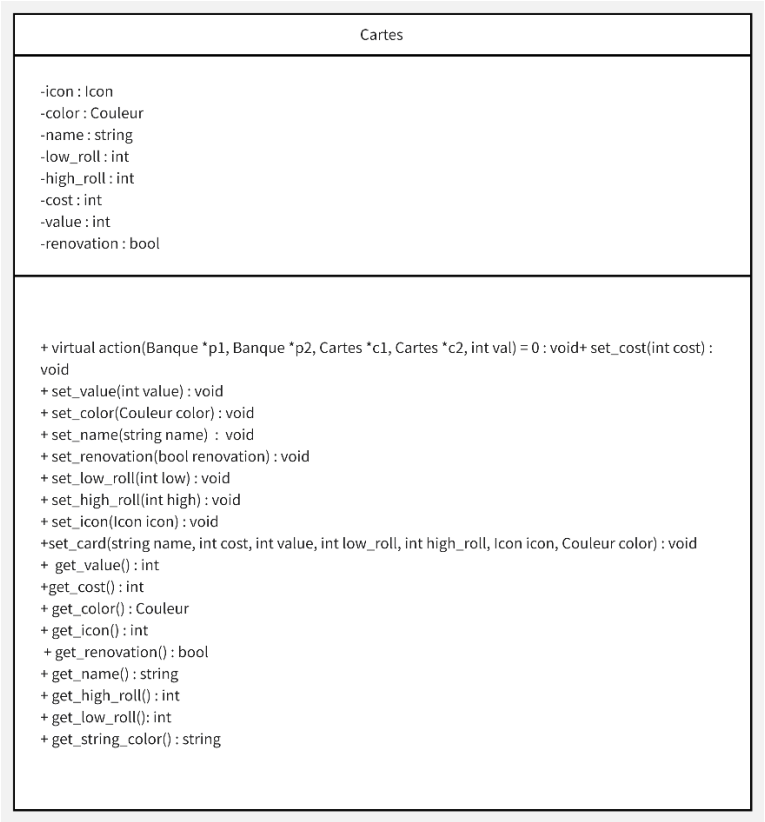
1.Couleur et Icon

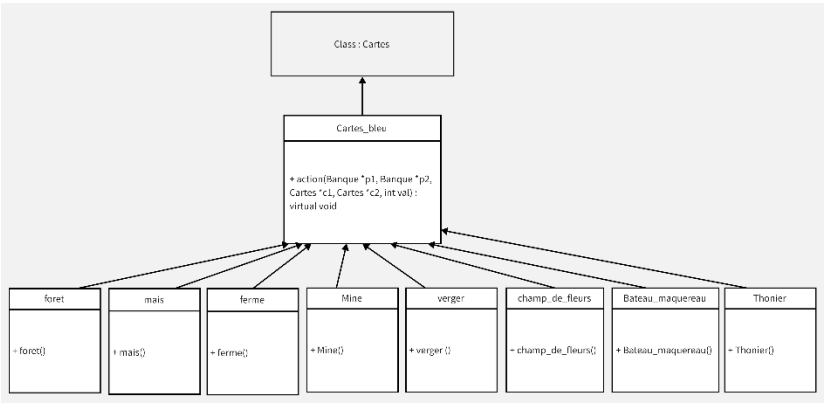


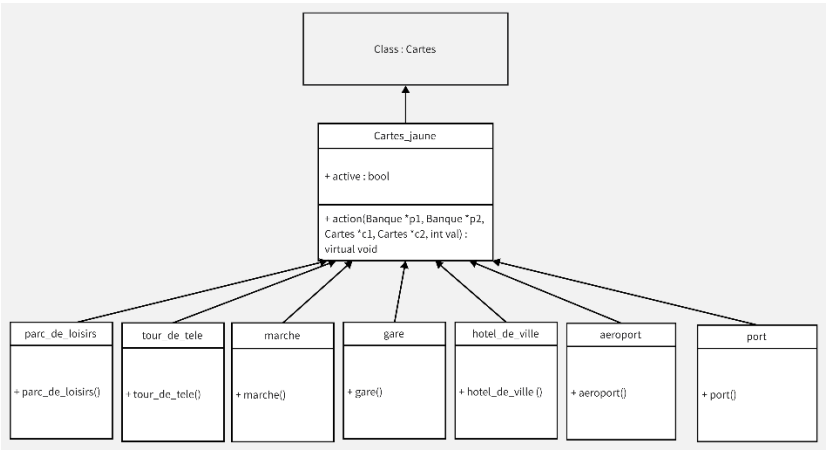
命名为Couleurs et Icones的文件是对游戏中所有出现的卡牌颜色和属性进行枚举。方便在Cartes系列文件里对每种不同卡牌颜色和属性的定义。

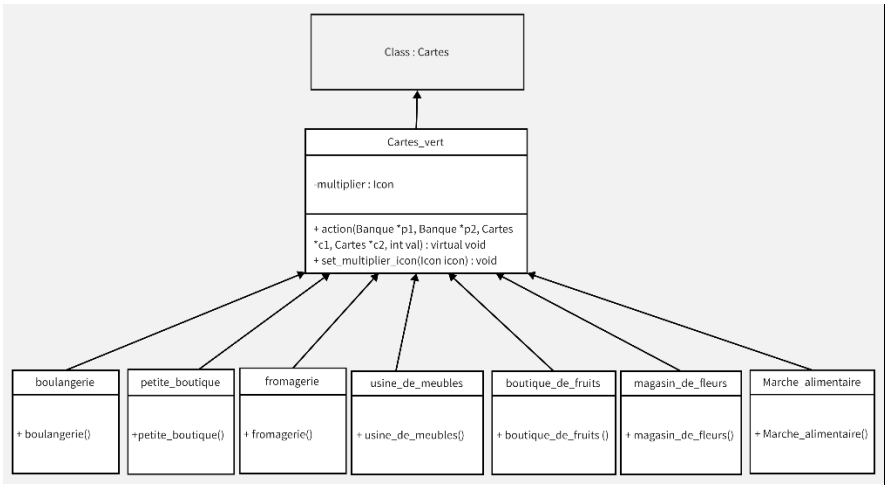
创建这个文件的本意是能便于对游戏进行扩展操作，在游戏标准版中只有【牧场】【农场】【面包房】【工厂】【餐厅】【建筑】【商店】【none】几种卡牌属性。在添加港口拓展包后我们添加了游戏中对应的【船】属性，对于之后的拓展包开发，若出现其他的卡牌属性，可以直接在该文件中添加。

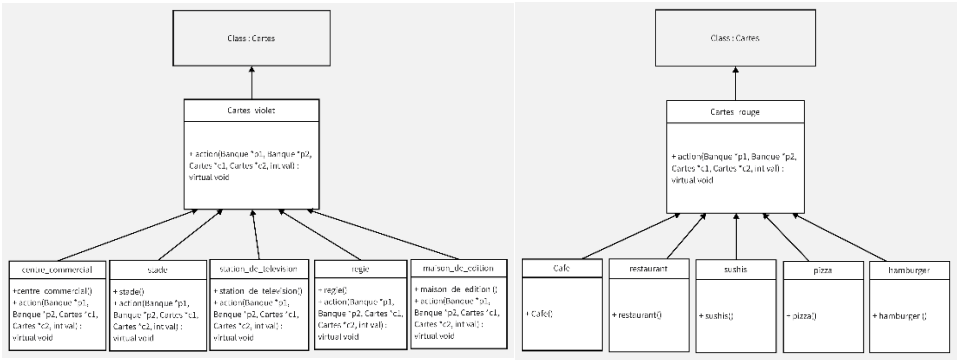
2. Cartes 等


Cartes
-icon : Icon
-color : Couleur
-name : string
-low_roll : int
-high_roll : int
-cost : int
-value : int
-renovation : bool
+ virtual action(Banque *p1, Banque *p2, Cartes *c1, Cartes *c2, int val) = 0 : void+ set_cost(int cost) : void
+ set_value(int value) : void
+ set_color(Couleur color) : void
+ set_name(string name) : void
+ set_renovation(bool renovation) : void
+ set_low_roll(int low) : void
+ set_high_roll(int high) : void
+ set_icon(Icon icon) : void
+set_card(string name, int cost, int value, int low_roll, int high_roll, Icon icon, Couleur color) : void
+ get_value() : int
+get_cost() : int
+ get_color() : Couleur
+ get_icon() : int
+ get_renovation() : bool
+ get_name() : string
+ get_high_roll() : int
+ get_low_roll(): int
+ get_string_color() : string

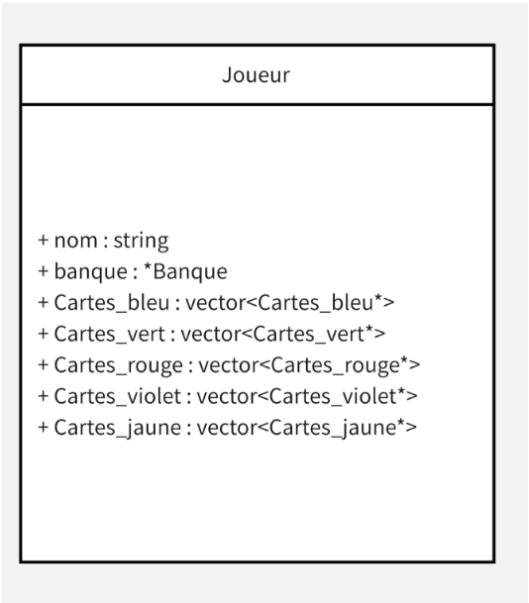

Class : Cartes
Cartes_bleu
+ action(Banque *p1, Banque *p2, Cartes *c1, Cartes *c2, int val) : virtual void
foret
+ foret()
mais
+ mais()
ferme
+ ferme()
Mine
+ Mine()
verger
+ verger ()
champ_de_fleurs
+ champ_de_fleurs()
Bateau_maquereau
+ Bateau_maquereau()
Thonier
+ Thonier()


Class : Cartes
Cartes_jaune
+ active : bool
+ action(Banque *p1, Banque *p2, Cartes *c1, Cartes *c2, int val) : virtual void
parc_de_loisirs
+ parc_de_loisirs()
tour_de_tele
+ tour_de_tele()
marche
+ marche()
gare
+ gare()
hotel_de_ville
+ hotel_de_ville ()
aeroport
+ aeroport()
port
+ port()





命名为Cartes的系列文件中定义了【红色卡】、【蓝色卡】、【紫色卡】、【绿色卡】、【黄色卡】。每张卡牌的属性至少包含cost，val，high_roll、low_roll、icon、coulour、name几种属性。我们通过在每个对应的颜色文件中对这些卡牌进行设置。这种方法同样是便于进行模块化的编程和易于对代码的理解。在添加其他新卡牌时，只需要在对应的颜色文件中重新生成定义即可。

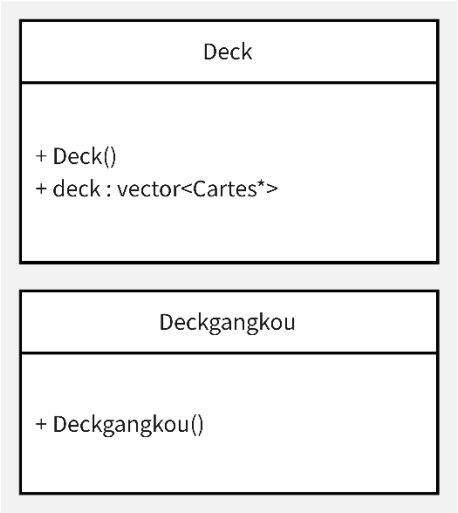


在Jeu文件中，我们创建了玩家类，并使用vector容器存储每个玩家的各类卡牌。

Cartes_jaune对黄色建筑类卡牌的生命周期负责，其中包含了6种游戏中需要玩家建造的建筑卡。当满足所有黄色卡被建造之后，当前玩家获得胜利。

针对游戏中不同卡牌的执行效果，由 Jeu 中的 Check_Cartes 代码执行。

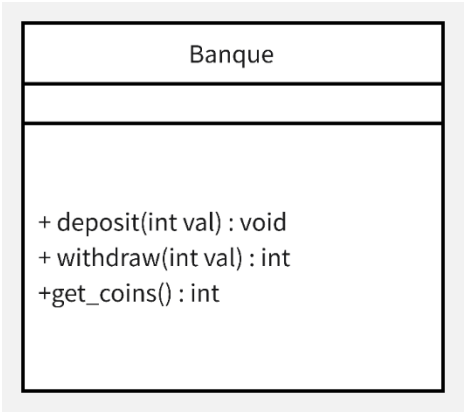
3. deck



命名为 deck 的文件包含了生成所有所需卡牌的代码，这个文件主要用于创建所有需要的卡牌在桌面上，通过 Jeu.cpp 中 deal()实现发牌，由 Jeu 中的输出代码打印。

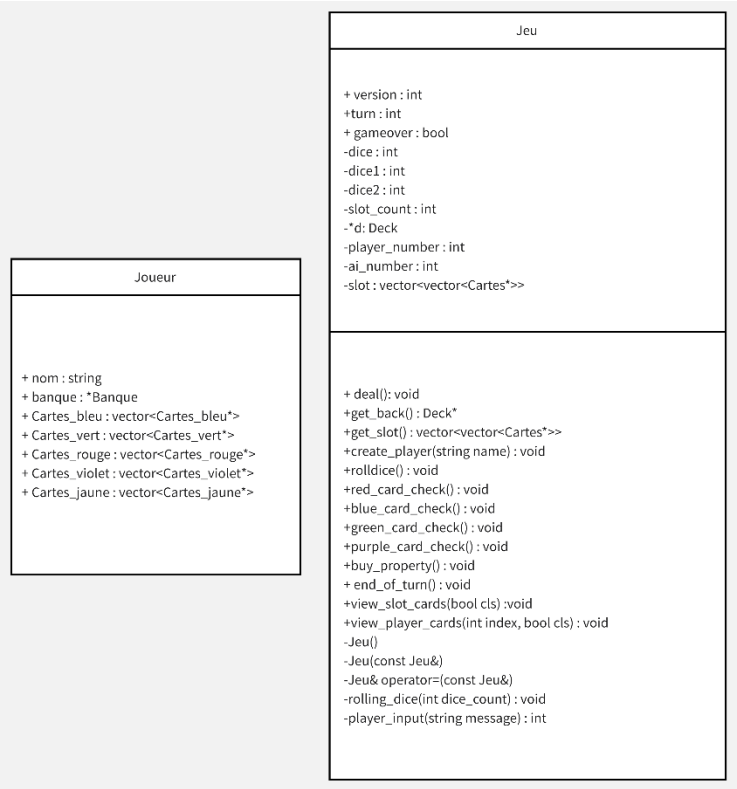
游戏开始时会选择此次游戏的版本，当玩家选择港口拓展包时，deck 文件执行 Deckgangkou();代码，生成所有卡牌和港口拓展包卡牌，在 vector 容器储存。当玩家选择购买卡牌时，会由上文中 Joueur 类中的对应颜色的 vector 容器储存。

5. Banque



命名为 Banque 的文件主要实现游戏内玩家金币数的操作，get_coins()实现对玩家当前金币数量的获取，withdraw 实现游戏中玩家消耗金币的数量。Deposit 实现游戏中玩家金币的获取。

6. Jeu



命名为 Jeu 的文件中包含了 Jeu 和 Joueur 这两个类。其中，Joueur 主要用来存储玩家信息、金币数以及储存游戏中玩家购买的所有卡牌。Joueur 类不对 Cartes 类的生命周期负责。

命名为 Jue 的文件是对游戏中桌面上的卡牌以及的一系列操作的实现 在游戏开始时，Jue 会在桌面上打印所有本次游戏中出现的卡牌，并在每个玩家对应的桌面上打印玩家当前已经建设的【建筑牌】。

游戏开始时玩家先选择是否选择港口拓展包版本，再选择玩家数量和 AI 玩家数量，并且为每个真人玩家游戏名字。根据选择的玩家和 AI 玩家人数可以实现，人人对战，人机对战，AI 对战三种游戏模式。AI 玩家可以根据桌子上的卡牌和金币数量随机购买或者不买卡牌，对火车站、游乐园等特殊卡牌也会做出随机操作，实现 Joueur 的基本操作。Jeu 根据输入结果对每个玩家的初始牌桌进行初始化，创造未建造的黄色建筑卡区域和各种颜色卡牌区域。

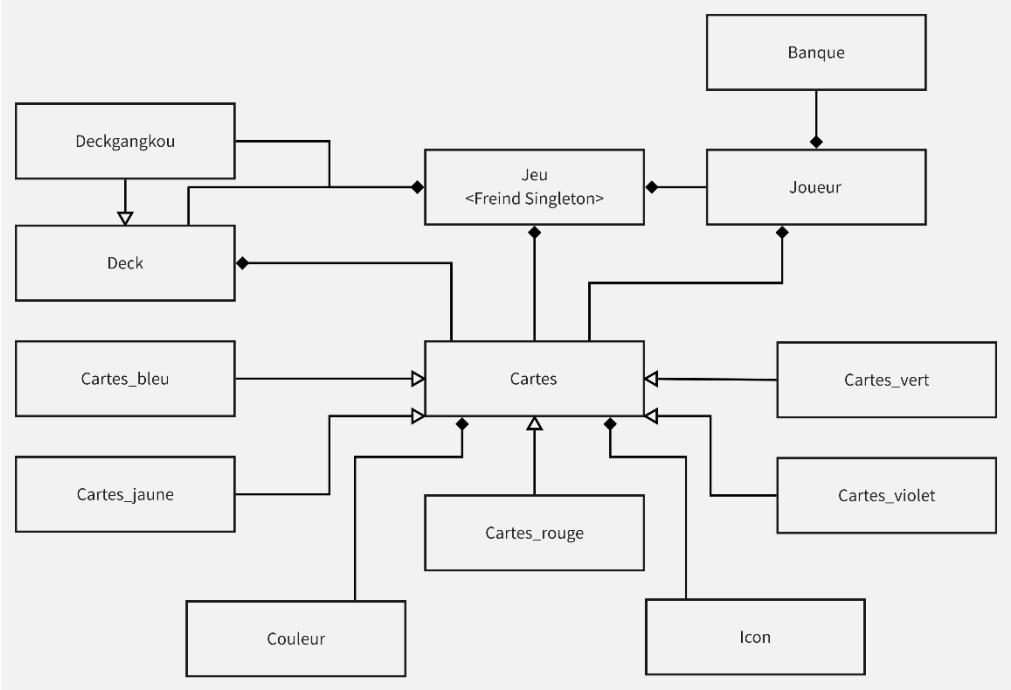
```
void Jeu::rolldice()
{
   if(version==1) {
      for (int i = 0; i < players.size(); i++) {
         players[i]->Cartes_jaune[6]->active = true;
      }
   }
       system("cls");
       this->deal();
       this->view_slot_cards(true);
       this->view_player_cards(this->turn, false);

       int dice_count = 1;

       // 判断火车站是否被建造，并选择骰子数
       if (this->players[this->turn]->Cartes_jaune[0]->active)
       {
      cout << "1 or 2 des: "<<endl;
      if(this->turn < player_number-ai_number) {
         cin >> dice_count;
      }
       }
```

游戏过程中，每轮玩家回合开始 Jue 会自动执行 rolldice 代码，由于【建筑卡】中【火车站】会对玩家掷骰子的操作产生影响，所以会先进行一次对当前玩家是否拥有该卡牌进行一次-判断，最后由 rolling_dice 代码执行一次通过 rand（）实现的掷骰子操作。

由于骰子的结果会对游戏产生不同的影响，并且游戏卡牌的结算顺序是根据【红色卡】、【蓝色卡】、【绿色卡】、【紫色卡】顺序结算的。所以本项目选择在 Jeu 中按顺序检索所有卡牌并且执行对应效果。其中，紫色卡执行特殊效果，我们将所有紫色卡结算整合在 purple_card_check 中。

另外对于玩家的动作，我们通过调用 buy_propetry 代码实现，玩家在游戏过程中的输入首先由 player_input();进行输入的判断，先由玩家输入一个 string 字符串，程序根据 regex view 的方法完成对输入内容的判断，若玩家选择购买则根据 buy_propetry();实现购买流程，并且将购买的卡牌记录在玩家对应颜色的 vector 容器中。

四、UML



五、优点

1.引用单例模式，保证 class Jue 只能产生一个实例。

2.引入牌库和牌桌，以此保证选择不同的版本可以生成不同的卡牌，同时在编程时可以直接在牌库中更改卡牌的相关数据，便于进行模块化管理与编程。

3.模块化编程，将整个程序分成 Jeu，Cartes 等，deck，banque，Couleurs et Icones 多个模块进行编写，方便修改。

4.添加扩展包内容，引入了港口拓展包，提供了新的卡牌和机制，加入扩展卡之后，游戏的策略度明显变高，增添了游戏的竞技性。

5.实现 AI 和玩家共同进行游戏，可以完成纯玩家玩法，AI 和玩家互战玩法以及 AI 玩法。

6.可扩展性，程序使用模块化编程，在 Cartes 文件和 Jeu 文件 red_card_check()、blue_card_check()、green_card_check()、purple_card_check()中对不同颜色的牌进行模块化编写。若需要添加新

版本的游戏，在对应颜色文件中添加牌，编写每张牌的属性，并且在check_card()中对相应卡牌新增功能进行补充即可。玩家每回合由roll_dice()开始，end_of_turn()结束，若新增卡牌需要玩家在回合开始或结束时手动操作，在前两文件中添加相应功能即可。对于修改新的AI也同理，只需在check_card()、roll_dice()、end_of_turn()中对需要玩家进行操作的牌编写随机反应即可。

六、可改进之处

1，可以改进AI策略，使得玩家与AI对战时选择AI的不同难度
2，可以考虑使用工厂方法模式，来集中实现对各个卡牌内存空间的开辟
3，针对于进行特殊操作的卡牌如紫卡和黄卡，可以单独封装简化主程序的逻辑。
4，图形化界面可以使游戏更美观，更易于新手理解
5，占用内存过大

七、bilibili视频链接

https://b23.tv/9jo6mMp

八、个人贡献

姓名	学号	负责内容	时长	贡献度
宋鹏宇	20124738	主编文件： Cartes_violet.h Cartes_violet.cpp 辅编文件： Jue.cpp Jue.h 重要功能实现： 游戏人数设计 输入错误判断(cin.fail()) 购买相关流程 buy_property() 部分紫色卡牌功能实现 其他工作： 撰写报告 代码测试与反馈	30h	19%
王宇星	20124757	主编文件： Cartes_rouge.h Cartes_rouge.cpp 辅编文件： Jue.cpp Jue.h 重要功能实现： Rolldice 随机掷骰子 黄色建筑卡牌火车站的功能实现 红色卡牌遍历以及功能实现 其他工作： 视频录制 代码测试与反馈	30h	19%
刘丁玮	20124715	主编文件： Cartes_vert.h Cartes_vert.cpp 辅编文件： Jue.cpp Jue.h	30h	19%

		重要功能实现： 卡牌浏览（view） 玩家界面游戏卡牌打印 绿色卡牌功能实现 其他工作： UML 图绘制 代码测试与反馈		
彭健程	20124717	主编文件： Cartes_jaune.h Cartes_jaune.cpp Deck.h Deck.cpp Singleton.h Jue.cpp Jue.h 重要功能实现： 单例设计模式 游戏版本选择 AI 玩家 紫色卡牌商业中心、港口、游乐园功能的实现 黄色卡牌的功能实现 其他工作： 框架搭建 代码测试与反馈	45h	24%
高子琪	19124660	主编文件： Cartes_bleu.h Cartes_bleu.cpp Banque.h Banque.cpp 辅编文件： Jue.cpp Jue.h 重要功能实现： AI 玩家 游戏玩家创建 Create_player 黄色建筑卡牌电视台功能实现 蓝色卡牌功能实现 其他工作： 代码测试与反馈	30h	19%