程设大作业“大富翁”实验报告

一. 程序功能介绍

二. 各模块与类设计细节

三. 项目分工情况及总结

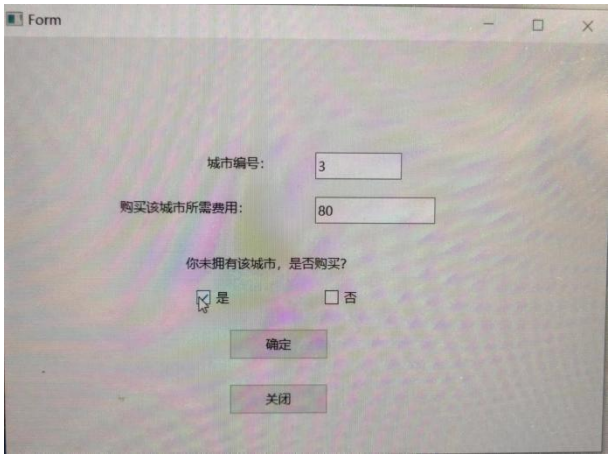
一. 程序功能介绍

本项目以 qt 为基础实现了一个可供 4 位用户使用的大富翁游戏。用户在打开程序后将会进入 begin 界面（如图所示），在点击开始游戏后进入 bigricher 主界面，bigricher 主界面包含如下内容：

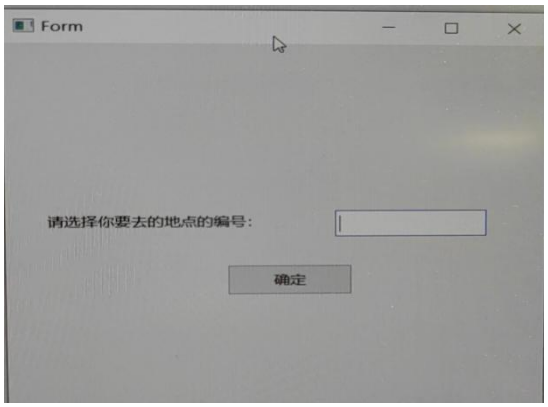
1. 主界面左边是每个方块的简要介绍，包括编号，类型，以及名称，玩家可借此简要了解左边的地图内容。
2. 介绍下面是游戏规则的选项，玩家可通过此处查看游戏规则。
3. 主界面右边外圈则是大富翁的地图，每个方块根据类型和编号的不同有着不同画面，在每个方块上有着“查看”按钮，玩家可以点击此处查看此方块的具体信息。
4. 而在地图中央左上角是每个玩家的存活状态（是否参与游戏或者是否已经退出游戏），右上角是每个玩家所在的位置，中间部分记录了当前游戏轮数，以及此时是哪位玩家的回合，骰子的投掷也在这里实现，而下方则是每个玩家的详细信息按钮，玩家可以通过这个查看自身信息。

玩家主要依靠 bigricher 主界面完成游戏各个操作，在投掷骰子后，会自动触发玩家应到格子的事件，其中包括城市、机场、银行、工厂、监狱、事件以及奖励等格子。每个种类的格子可以进行如下操作：

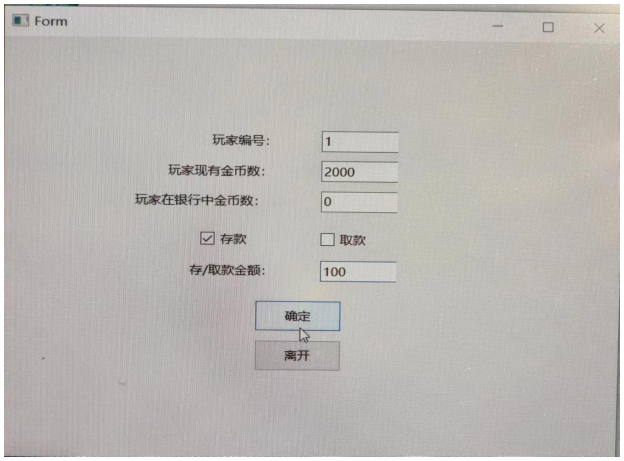
1. 城市：玩家可以购买地皮和房产，借此达到收过路费和增加过路费的目的，当非持有者来到这个格子时，需要向持有者缴纳一定的过路费。



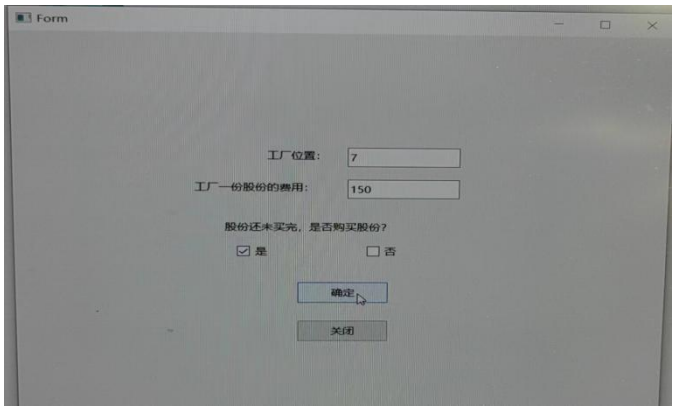
2. 机场：玩家可以选择任意一个格子直接到达，并触发目标格子的事件。

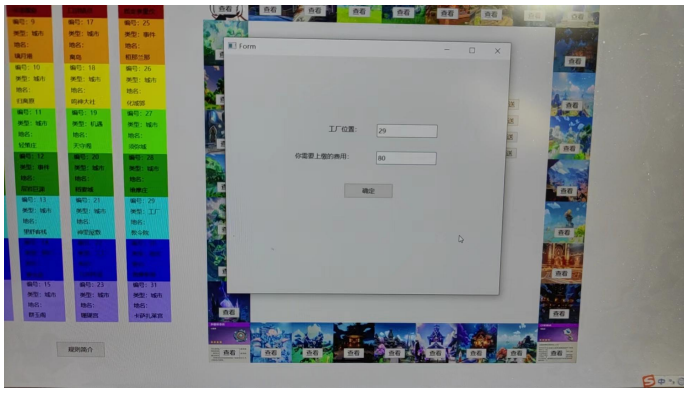


3. 银行：玩家可以在此处进行存款和取款操作，存入的钱将会以一定的比例在经过每轮后增加。

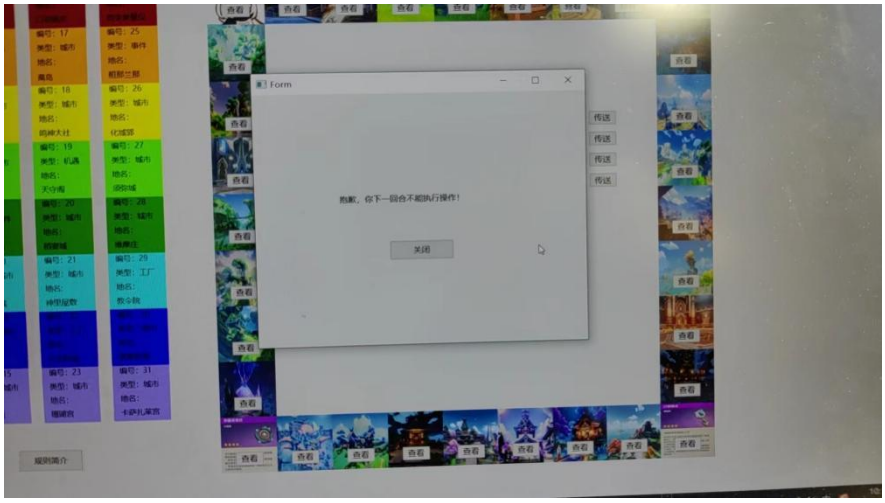


4. 工厂：玩家可以购买工厂的股份来收过路费，未持有工厂股份的玩家在来到这个格子后将付给持有股份的玩家一定费用。

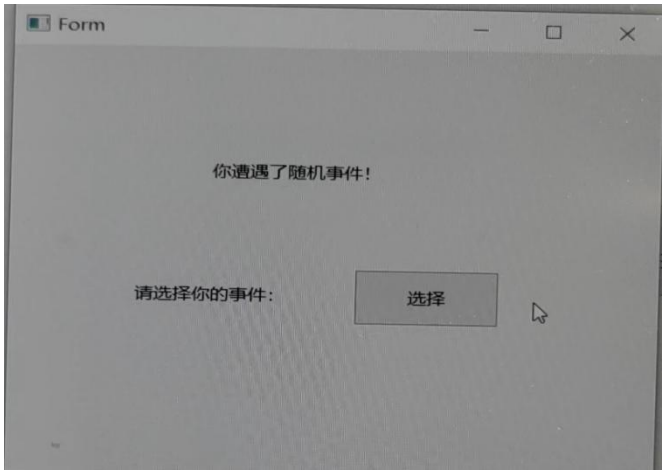


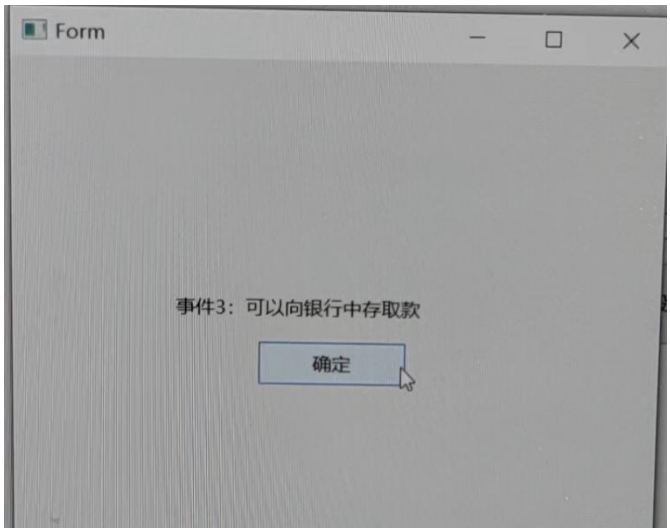


5. 监狱：来到此处的玩家将暂停一个回合（下一回合实行）。

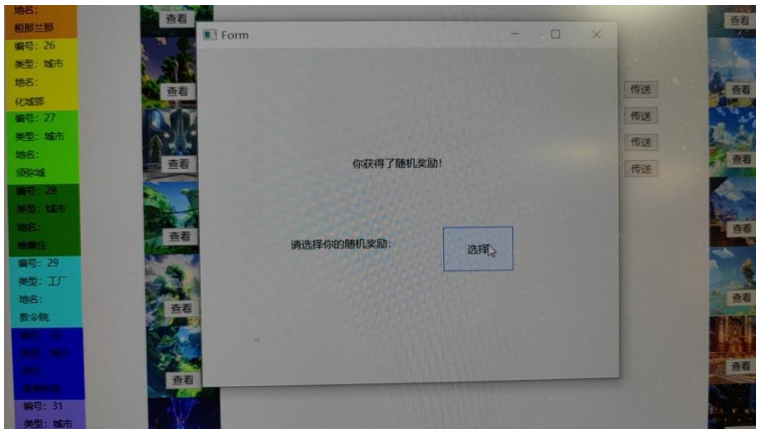


6. 事件：在十个随机事件中抽取一个执行。





7. 奖励：在十个随机奖励中抽取一个执行。



8. 起点：在经过起点后，玩家的金币数将会增加一定数量。

玩家轮流投掷骰子进行行动，当玩家身上携带金币不足以支付过路费时，会进入濒死状态，玩家可以选择卖出自己拥有的房产、取出自己银行中的钱、卖出自己工厂中的股份以及直接进行破产清算。每个选择对应的内容如下：

1. 卖出房产：程序将会展示玩家所拥有的房产以及卖出的收益，并展示当前玩家的金币来便于玩家选择。
2. 取出银行中的钱：程序将会展示玩家在银行中存储的金币数，并以一定的折扣比例予以取出。
3. 卖出工厂股份：程序将会展示玩家是否在工厂中持有股份，并供玩家卖出实际拥有的股份。
4. 破产清算：程序将会把玩家的房产、工厂股份以及银行中存储的金币收为国有，并将玩家的状态定义为离开游戏。

为避免游戏一直进行，程序采取如下胜利条件：若在 80 个回合前场上只有一个玩家存活，则该玩家获胜；若 80 个回合后场上有两个及以上玩家存活，则现有金币最多者获胜。获胜玩家的编号将由 end 界面展示。

此外，在游戏过程中，程序对一切输入进行了判定，当玩家的输入不合法时，将会弹出 wrong_input_sign 界面，来提醒玩家输入的不合法性。

以上便是对此程序所实现功能的简要介绍。

二. 各模块与类设计细节

本程序大致分为以下几个板块：

1. block_event

block_event 板块中包含了所有格子的事件，包括城市，工厂等，以及十个奖励和事件的内容。

其中，各个类的设计如下：

i. airport_event 定义了一个 airport_event 类，实现了根据玩家输入的数字转移到对应的格子的功能。

ii. bank_event 定义了一个 bank_event 类，依靠不同的按钮的槽函数实现了金币展示、存钱以及取钱等功能。

iii. city_event 定义了一个 city_event 类，与其后的 city_event2，city_event3 的类，共同实现了城市在不同情况下的展示界面，其中，city_event 实现了在已经拥有的城市上修建房屋的功能，city_event2 实现了玩家到达未被持有的城市是否购买的功能，city_event3 实现了向经过玩家收取过路费的功能。

iv. factory_event 和 factory_event2 分别定义了 factory_event 和 factory_event2 类，共同实现了玩家经过工厂的不同情况，其中，factory_event 实现了玩家到达未被购买的工厂时，提供购买的功能，factory_event2 实现了玩家经过工厂缴纳过路费的功能。

v. prison_event 和 prison_event2 分别定义了 prison_event 和 prison_event2 类，共同实现了玩家到达监狱和遭受监狱惩罚的不同情况，其中，prison 实现了玩家到达监狱后将停止行动的功能，prison_event2 实现了玩家遭受监狱惩罚依然无法行动的功能（prison_event2 通过试玩发现影响游戏体验，所以这个类的功能并没有实现）。

vi. re 以及 re_event 等类共同实现了随机事件的功能，其中 re_event 中初始化了 re_event1～10 的所有事件便于展示，而 re 实现了随机事件的抽取功能，re_event1～10 则实现了各个事件的展示以及对应操作。

vii. rr 以及 rr_event 等类共同实现了随机奖励的功能，其中 rr_event 中初始化了 rr_event1～10 的所有奖励便于展示，而 rr 实现了随机奖励的抽取功能，rr_event1～10 则实现了各个奖励的展示以及对应操作。

2. information_show

information_show 板块中包括了所有的信息展示，其中有玩家，工厂，城市等信息展示的代码。

其中各个类的设计如下：

i. bank 定义了一个 bank 类，实现了展示各个玩家在银行中存储的金币数量的功能，其中的 fresh 函数起到了初始化和展示数量的作用。

ii. city_show 定义了一个 city_show 类，实现了展示城市全部信息（例如修建房屋费用，拥有者，已经建有几座房屋等）的功能，其中的 fresh 函数起到了初始化和同步展示的功能。

iii. factory_show 定义了一个 factory_show 类，实现了展示工厂是否被持有以及持有者的功能，其中的 fresh 函数起到了初始化和同步展示的功能。

iv. player_information_show 定义了一个 player_information _show 类，实现了展示玩家全部信息（例如玩家编号，是否在游戏中，银行存款，拥有城市，拥有金币等）的功能，其中的 fresh_information 函数通过传入玩家编号，起到了初始化和同步展示的功能。

v. re_show 定义了一个 re_show 类，实现了展示全部随机事件的功能。

vi. rr_show 定义了一个 rr_show 类，实现了展示全部随机奖励的功能。

vii. begin_information 定义了一个 begin_information 类，实现了展示经过起点的操作的规则的功能。

viii.airport_show 定义了一个 airport_show 类，实现了展示机场操作规则的功能。

ix.prison_show 定义了一个 prison_show 类，实现了展示监狱操作规则的功能。

3. near_death

near_death 板块是濒临破产操作的代码集合，其中包含破产清算，卖出股份等代码。

其中各个类设计如下：

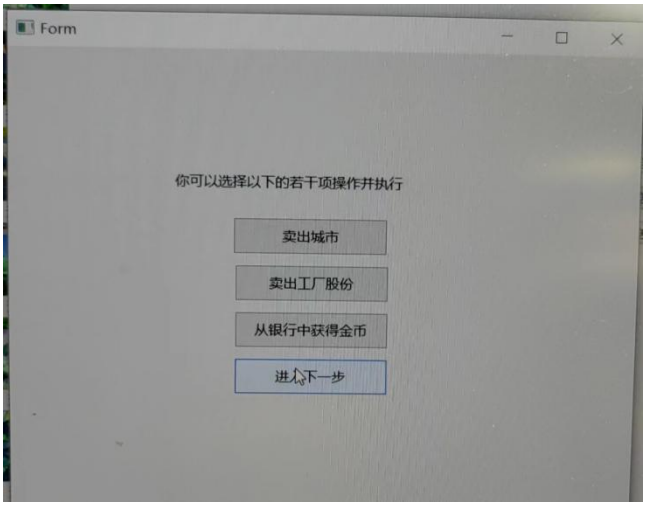
i. get_money_from_bank 定义了一个 get_money_from_bank 类，实现了玩家在濒临破产时，从银行中取出存储的金币的功能，其中的 fresh 函数起到了展示银行当前金币数的作用，在槽函数中也进行了简单的输入判断来确保取出的钱的合理性。

ii. near_death_choose 定义了一个 near_death_choose 类，实现了玩家在濒临破产时进行选择的功能，提供了 get_money_from_bank，sell_city，sell_factory，以及结束游戏四个选项。其中的 fresh 函数起到了展示欠款数的功能。

iii. sell_city 定义了一个 sell_city 类，实现了玩家在濒临破产时出售房产的功能，其中的 fresh 函数起到了展示当前持有房产和卖出所得收益的功能，并且通过槽函数实现卖出城市的功能，并简单进行了输入判断来确保卖出城市的合理性。

iv. sell_factory 定义了一个 sell_factory 类，实现了玩家在濒临破产时出售工厂股份的功能，其中的 fresh 函数起到了展示当前持有工厂股份的功能，并且通过槽函数实现卖出工厂股份的功能，并简单进行了输入判断来确保卖出者确实拥有股份。

v. sell_out 定义了一个 sell_out 类，实现了在做出选择后进行新的步骤的功能。



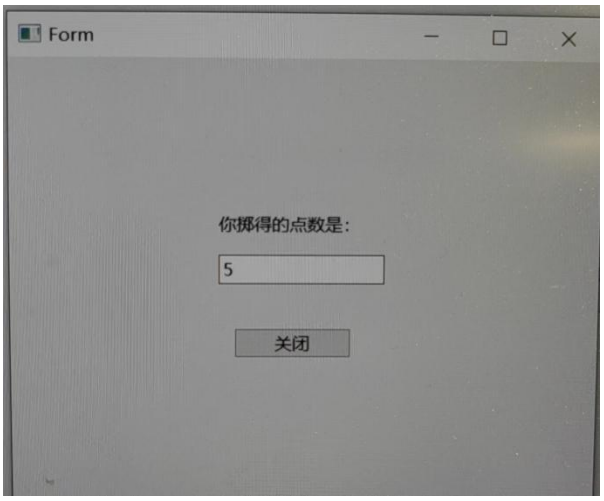
4. sign

sign 板块是所有提示的集合，包括现金不足，骰子点数以及输入不合法这三个提示。

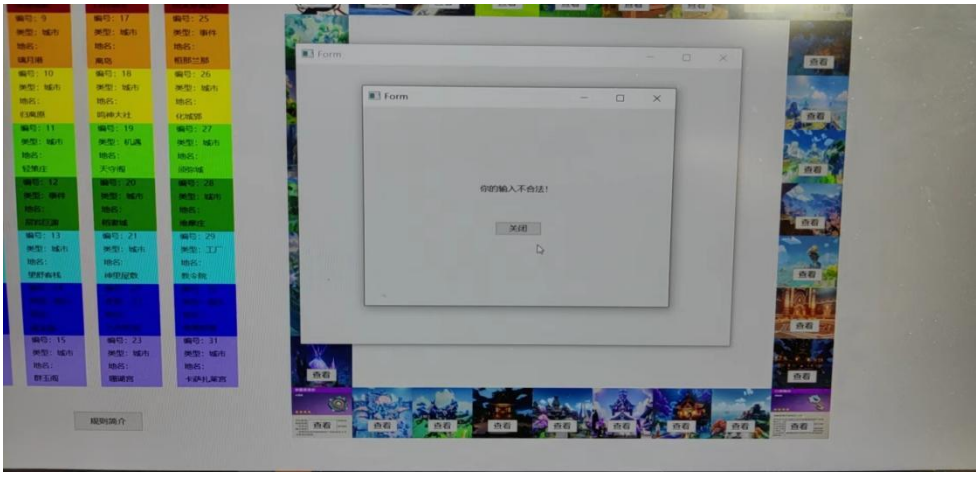
其中各个类设计如下：

i. lack_of_money 定义了一个 lack_of_money 类，实现了对濒临破产玩家以及购买不了城市或者股份的玩家的提醒功能。

ii. show_dice_num 定义了一个 show_dice_num 类，实现了骰子点数的展示，通过其中的 fresh 函数展示骰子的点数。



iii. wrong_input_sign 定义了一个 wrong_input_sign 类，实现了对于玩家输入不合法的提醒功能。



5. begin
   begin 定义了一个 begin 类，实现了进入游戏的初始界面的展示功能，其中有着开始游戏的槽函数。
6. bigricher
   bigricher 定义了一个 bigricher 类，起到了主界面的作用，其中初始化了 block_event，information_show 以及 sign 等类，并使用 fresh 函数将主界面的展示内容以及可供查看的内容初始化，同时定义了一些基本的地图设置（例如回合数，当前玩家数量，下一回合玩家编号，当前玩家编号等），其后的 take_event 函数实现了投掷骰子后的事件的发生，并包含多个查看信息的按钮可供玩家使用。
7. end
   end 定义了一个 end 类，起到了结束游戏后宣布胜利者的作用。
8. map
   map 定义了一个 map 类，实现了地图信息和基本数值的设置，初始化了各个格子的情况并实现了玩家 player 的初始化，以及工厂股份，工厂拥有者等格子内容，定义了 bank_rate 等全局使用的常量。与此同时定义了一些游戏进程中使用的基本的函数，例如 move 函数，展示当前位置的同时实现了起点的加钱功能，player_die 函数，实现了破产清算，财产清空的功能。
9. player
   player 定义了一个 player 类，实现了 player 的 num,position,coins,coins_inbank,if_extra_round,if_skip_round,if_play 以及 if_airport 等内容的定义，并在 map 中实现了初始化。
10. rules
    rules 定义了一个 rules 类，实现了大富翁游戏的规则展示。

以上的每个类都有对应的页面进行展示。

三. 项目分工情况及总结

项目分工情况如下：
代码编写：邹宇桓，陈思进，白林
代码整合：邹宇桓，陈思进

实验报告编写：白林

总结反思：

1. 在进行这种工程量较大的程序编写时，应当在编写前进行大致的规划，避免在编写过程中遇见代码过于冗杂的情况，同时提前进行规划有助于避免在过程中遇到难以调整的底层错误，从而必须重写的情况。
2. 在进行这种团队合作时，应当商量好分工，并合理分配任务，将可以并行进行的任务同时进行，这样极大的提高了代码的编写效率，在进行接力式编写时，要在交接的时候向后者指明要做的任务和需要注意的点，这样将提高后者的编写效率，加快团队的进度。
3. 编写函数以及命名变量时，要充分考虑其含义的明确，避免队友难以理解其中的内容。要提供合理必须的注释，这将帮助队友更好地领悟编写者的意图，避免认知错位导致的程序编写歧义。
4. 我们认为我们这个项目的优势在于它能够实现比较多的功能，已经是一个较为成熟的大富翁的游戏；劣势在于图形界面比较简陋，图片的移动并没有实现，所以玩家可能没有像在网上玩真正的大富翁那么有游戏体验。