南开大学
Nankai University

南　　开　　大　　学

计　算　机　学　院

高级语言程序设计课程设计报告

信任的进化

莫骐骥

年级：2020 级

专业：计算机科学与技术

2022 年 5 月 7 日

摘要

《信任的进化》是一个由新加坡独立游戏开发者 Nicky Case 于 2017 年使用 flash 制作的以游戏为形式的合作博弈论的互动讲解手册，共 10 关。它改编自罗伯特·阿克塞尔罗德的著作《合作的进化》。我使用 Qt 实现了这款游戏的全部功能。

关键字：信任的进化，博弈，合作

目录

一、 游戏介绍

(一) 游戏背景及初衷

《信任的进化》是一款获奖颇丰的小游戏。在游戏中，开发者通过有趣的博弈环节教给我们许多关于博弈论，人性，进化论的知识，引发了我们对一些历史事件和生活现象的思考。也正因为这款游戏具有鲜明而易于提取的逻辑体系，以及深刻的教育意义，我决定以此游戏为基础，融入我自己的理解与思考，去编写我的 c++ 大作业。

不论是基于博弈论的囚徒困境，还是有些科幻的黑暗森林法则，在只进行一轮的博弈中，欺骗永远是正解。然而，当今社会中，命运共同体与产业链的连接使零和博弈越来越少。与此同时，在世界被全面信息化缩小的今天，与相同对象博弈次数持续增多。那么，正确的选择又会是什么呢？首先，基于信任的一些了解。信任是人类社会的基本组成要素，不同领域的学者针对信任的定义进行了研究。心理学认为，信任是个人或者组织依赖另一方的语言、口头或书面承诺的意愿。管理学认为，信任是减小不确定性、降低交易成本、提高满意度的一种组织控制形式。营销学中定义信任是依赖自己所信赖的交易方的意愿，信任会帮助交易双方建立长期的交换关系和合作关系。那么，在什么条件下，信任会充分发挥它的积极作用呢？

这款游戏会告诉我们答案。

(二) 游戏设定

1. 游戏场景

我们将人与人交流时选择合作或者欺骗具象化为两人互相赠与对方硬币，将交流称为博弈。那么在一次交流中只有以下三种结果。

1. 双方均选择赠与对方硬币, 表示二人都选择合作。
2. 双方均选择不赠于对方硬币，表示二人都选择欺骗。
3. 二人中一人选择赠与对手硬币而另一位选择不赠与对手硬币，表示赠出者受到欺骗，而自私者获利。

在本游戏中，大致会进行如下游戏环节：

1. 游戏玩家（我/你）与不同游戏角色的博弈。
2. 多个不同种类游戏角色在不同环境参属下进行博弈。

2. 游戏参数设置

- 奖励值机制，定义当双方均选择合作是收获多少，均选择欺骗时失去多少，被骗者失去多少，获利者获得多少。
- 误解机制，还原人与人之间的误会，在现实生活中，由于沟通不到位等原因，我们通常误会对方的行为，我们看到的可能不是对方的初衷。在游戏中表现为，设置一个参数值，表示游戏角色本意是赠与硬币缺表现为没有将硬币赠出的概率。
- 交流充分性机制，还原人与人之间沟通的匮乏与充足，例如在人与人之间只进行一次交流时，总选择欺骗的人显然会获利。在游戏中的表现是：设置两名博弈轮数。

3. 游戏角色

游戏共设置 8 类游戏角色，他们都有自己的博弈策略，代表现实生活中形形色色不同性格特征的人。

- 复读机，先进行合作，之后会模仿对手上一轮博弈的选择。
- 小粉红，永远选择合作。
- 老油条，永远选择欺骗。
- 黑帮老铁，第一次选择合作，一旦被对方欺骗过，就永远选择欺骗。
- 福尔摩星，前四轮博弈选择合作，欺骗，合作，合作，花费这四轮博弈的时间去观察对手的行为，如果对手一直选择合作，就会化身老油条永远欺骗，如果对手有过欺骗行为，则会化身复读机模仿对手上一轮博弈的行为。
- 复读鸭，复读机的宽容版本，只有对手连续欺骗他两次才会反击回去。
- 一根筋，第一轮先合作，只要对手在上一轮合作，一根筋就会选择合作；如果对手在上一轮选择欺骗，一根筋会选择自己上一轮相反的行为。
- 胡乱来，对于每一轮博弈，胡乱来选择合作和欺骗的概率都为 1/2。

二、 编程实现

(一) 主要数据结构

- Qpainter 对象绘制背景、线条等。
- 基类对象指针指向派生类对象实现多态、动态联编，开辟二维动态对象数组。
- Qlist 存储处于博弈列表中的对象。
- QHash 存储繁衍的最高的几个角色类型及数量。

(二) 玩家类设置

基类 player

```
class player
{
public:
    player();
    bool myact ;//自己的决策
    bool evercheated = false;//标记自己是否被欺骗过
    bool mylastchoie = 1;//自己上一次的选择
    int twocheat = 0;//复读鸭需要两次以上才反击
    int my_score = 0;//自己的得分
    int t_act = 0;
    bool getact() {return myact;};
    virtual QString getname() = 0;
```

```
    virtual QString getchinesename() = 0;
    virtual void act(bool opact) = 0;
};
```

派生类举例：福尔摩星

```
class detective:public player{
public:
    detective(){
        myact  = 1;
        evercheated = false;
        t_act = 0;//进行决策的次数
        mylogic = new bool[3]{0,1,1};
    };
    QString getname(){
        return "detective";
    }
    QString getchinesename(){
        return "福尔摩星";
    }
    bool *mylogic;
    void act(bool opact){
        if(t_act<3){
            myact = mylogic[t_act];
        }else{
            if(evercheated == true){//化身复读机
                if(opact == 1)
                    myact = 1;
                else
                    myact = 0;
            }else{//化身老油条
                myact = 0;
            }
        }
        t_act++;
    }
};
```

(三) 页面技术点

1. intro：介绍页

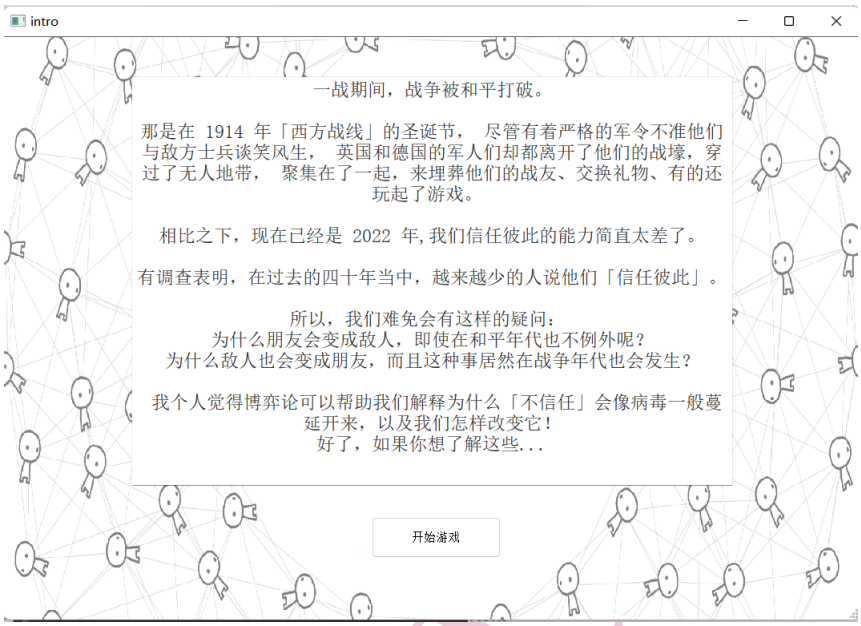

图 1: intro：介绍页

使用 qpainter 绘制背景图片，自适应大小。

2. onegame：游戏 demo

图 2: onegame：游戏 demo

介绍游戏 demo 规则，对手的行为使用 qrand 随机数生成，若为偶数则合作，奇数则欺骗，我的行为使用下方的两个按钮来控制。

3. repeatedgeme: 和多位角色逐一博弈

图 3: repeatedgeme: 和多位角色逐一博弈

头文件

```
    copycat obj1;//创建五个对手对象
    cheater obj2;
    pink obj3;
    grudger obj4;
    detective obj5;
    QList<player*> optlist;
    player* curropt = nullptr;
    }
```

构造函数主要代码

```
    qsrand(QTime(0,0,0).secsTo(QTime::currentTime()));
    round = qrand()%5+3;//初始化和第一个对手的对战轮数
    //qDebug() << "total round !"<<round;
    optlist.append(&obj1);
    optlist.append(&obj2);
    optlist.append(&obj3);
    optlist.append(&obj4);
    optlist.append(&obj5);
    curropt = optlist.first();//当前的对手对象
```

我选择合作

```
void repeatedgame::on_coop_clicked()//我选择合作
{
    QFont font;
    font.setPointSize(15);//字体大小
    ui->mycurr->setFont(font);
```

```
6       ui->optcurr->setFont(font);
7       ui->optchoice->setFont(font);
8       ui->mytotalscore->setFont(font);
9       ui->mytotalscore->setStyleSheet("color:blue");
10      //qDebug() << "curr round !"<<currround;
11      my_act = 1;
12      if(currround<round){//如果还没到最大轮数
13          if(currround>=1){//如果不是第一轮
14              curropt->act(my_last_act);//让对手根据我上一次行动来获取当前的选
                    择
15          }
16          getresult(curropt);
17          my_last_act = my_act;
18          ui->optchoice->setText("对手是否选择合作: "+QString::number(curropt->
                getact()));
19      }
20      currround++;
21      ui->mycurr->setText(QString::number(mycurrscore));
22      ui->optcurr->setText(QString::number(curropt->my_score));
23      ui->mytotalscore->setText(QString::number(mytotal));
24      if(currround==round){
25          qsrand(QTime(0,0,0).secsTo(QTime::currentTime()));
26          round = qrand()%5+3;
27          currround = 0;
28          num_fighted++;
29          optlist.removeAt(0);
30          if(optlist.isEmpty()==false){
31              curropt = optlist.first();
32              curropt->getname();
33              ui->optid->setText("对手"+QString::number(num_fighted+1)+"/5");
34              ui->mycurr->setText(QString::number(0));
35              ui->optcurr->setText(QString::number(0));
36              ui->optpicture->setPixmap(QPixmap(displayopt(num_fighted)));
37              mycurrscore = 0;
38          }
39          else{
40              ui->cheat->hide();
41              ui->coop->hide();
42              ui->summary->show();
43          }
44      }
45  }
```

4. showscore: 展示分数过渡页

图 4: showscore: 展示分数过渡页

唯一值得一说的是如何接受上一个页面的变量值，做法可以选择使用槽函数收发信息，也可以使用指针来传递。将 showscore 的 ui 指针声明为 public 类型，`public:Ui::showscore *ui;`，并定义指针：`showscore * p = this;`，在上个页面 repeatedgame 中声明`extern showscore*p;`，这样在 repeatedgame.cpp 中可以实现任意调用 showscore 对象的成员函数、变量、文本框对象等等。

5. tournament: 一次大赛

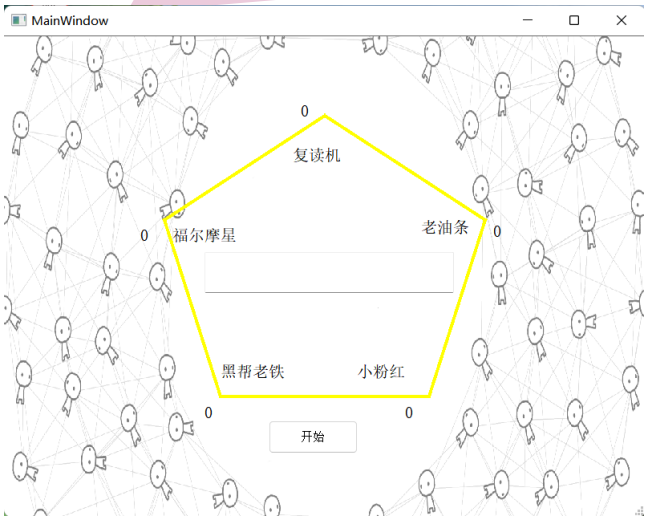

图 5: tournament: 一次大赛

这个页面实现的是五个角色之间两两对战，每次对战包括十轮博弈。

主要代码

```
while(list.length()>1){
    player* player_a =list.first();
    list.removeAt(0);
    for(auto curropt:list){
        battle(player_a,curropt);
        refresh(player_a);
        refresh(curropt);
        Sleep(1000);
    }
```

6. rpttounament: 沙盒模式多次大赛

这个页面的功能是整个游戏的核心功能。

图 6: rpttounament: 沙盒模式多次大赛布局 1

左侧是一个文本框，展示游戏规则。右侧进行三种类型的参数设定，分别是玩家成分设定、奖励值设定、进化参数设定（对局轮数、淘汰人数、误解概率），使用 stackedwidget 实现可以方便地进行窗口内小页面切换。

参数设定代码

```
while(list.length()>1){
    player* player_a =list.first();
    list.removeAt(0);
    for(auto curropt:list){
        battle(player_a,curropt);
        refresh(player_a);
        refresh(curropt);
        Sleep(1000);
    }
```

图 7: rpttounament: 沙盒模式多次大赛布局 2

这张图展示的是进化流程框，依次按下进行对局、淘汰、繁殖，可以逐步显示进化流程，设置不同的参数可以观察到很多有意思的结果。

代码见文末。

7. lastpage: 最后页

图 8: lastpage: 最后页

三、 讨论

最终，根据最终输出的的结果，在进行多轮次博弈的过程中，根据环境的不同，即参加博弈的角色种类和人数不同的环境中，最终获得优势的角色也会有所不同。但是，根据我们多次的在

不同条件下的实验，我们发现，在人物行为模式固定，不出现犯错可能的情况下，在大部分条件下，尤其是在一个环境相对宽容，博弈轮数较多的环境中，最终胜出的角色会是“复读机”。而在引入了犯错可能的情况下，容易胜出的角色是增加的新角色复读鸭。在我们的设定中，复读鸭一样会模仿对方的决策，但是比复读机更加的宽容，因而随着犯错概率不断增加，复读鸭的优势会越来越明显地大于复读机。关于复读机行为的模式，毫无疑问是人类社会中一种常见的行为模式，“信任招致信任，不信任招致不信任”，在人类社会中，这是一种在长期“信任的进化”中被选择出来的行为模式，既可以避免被人无止境的剥削，也可以和更多人形成合作，实现双赢。最终，我们的程序运行出的结果，也证明了其合理性。普遍信任又称为特质信任，是指个体在不能充分地判断他人动机等情况下，个体预期与之打交道的对象是善意的，同时也愿意承担做出这一判断所需要承受的相应风险。于是，信任也是一种风险评估的结果。预期到的风险越大，信任水平也会越低。由定义可知，普遍信任会对个体人际信任产生广泛的影响，如：普遍信任对不同主题和风险情境中的信任圈有显著影响，普遍信任水平越高，信任圈的规模就越大。显然，复读机的行为模式，就是一种特定的普遍信任，是一种包含风险评估的信任模式，显然与游戏中其他几种角色的行为模式相比，更具有合理性。但是，如果我们考虑更为真实的社会环境，引入更多的不确定因素，会发生什么呢？关于误解，所谓误解就是对别人的言语或行动的真正含义认识错误，或者失之偏颇。萨特有一种说法叫“他人即地狱”，也许这是我们不得不承认的，但我们始终认为太过极端，它应该建立在一定的基础上。首先，如果不能正确对待他人，那么他人便是地狱。即倘若自己是恶化与他人关系的原因，自己就得承担地狱之苦的责任。其次，如果不能正确对待他人对自己的判断，那么他人的判断就是地狱。他人的判断固然重要，但也只能参考，不能依赖，不可看作最高裁决，更不是自己行为的最终目的。凡以追求他人对自己赞美的人，必定陷入精神困苦之中。再次，如果不能正确对待自己，那么自己也是自己的地狱。人生旅途，每出差错，人们很容易去找社会原因、客观原因和他人原因，往往看不到自己的原因，正确对待自己常为我们所忽略。于是误解也正是由此而来。由此我们可以知道误解在人类社会中广泛存在，而减少误解带来的负面影响的方法之一，就是选择适度的宽容，例如复读鸭采取的容许自己被欺骗一次，而不会立即欺骗回去。选择宽容在博弈中有利于形成共赢的局面，避免两败俱伤的后果。例如在古巴导弹危机中，以肯尼迪和赫鲁晓夫分别领导的美国和苏联，已经开始将大军开往前线，相互进行着核威慑，在美军击毁苏军一架飞机，苏军用训练鱼类逼迫美军核潜艇的情况下，美国和苏联的领导人都选择了宽容，进行进一步的沟通和协商，从而避免了爆发核战争，使大半个世界被核弹轰炸，沦为核辐射区的命运。结合我们的程序输出结果，我们可以得知，在更贴近现实的情况中，由于信息的缺乏导致对形势错误的判断很有可能导致误解的发生。而消除误解带来的不良影响的好方法，就是在模仿他人行为模式的前提下，选择适度的宽容。

代码在这里