

**2019年春季学期  
计算机学院《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 李奕鹏 |
| 学号 | 1170800716 |
| 班号 | 1737101 |
| 电子邮件 | 17895987@qq.com |
| 手机号码 | 18800418032 |

**目录**

[1 实验目标概述 1](#_Toc7905864)

[2 实验环境配置 1](#_Toc7905865)

[3 实验过程 1](#_Toc7905866)

[3.1 待开发的三个应用场景 1](#_Toc7905867)

[3.2 基于语法的图数据输入 2](#_Toc7905868)

[3.3 面向复用的设计：CircularOrbit<L,E> 3](#_Toc7905869)

[3.4 面向复用的设计：Track 3](#_Toc7905870)

[3.5 面向复用的设计：L 3](#_Toc7905871)

[3.6 面向复用的设计：PhysicalObject 3](#_Toc7905872)

[3.7 可复用API设计 3](#_Toc7905873)

[3.8 图的可视化：第三方API的复用 4](#_Toc7905874)

[3.9 设计模式应用 5](#_Toc7905875)

[3.9.1 factory method 5](#_Toc7905876)

[3.9.2 abstract factory 5](#_Toc7905877)

[3.9.3 Iterator method 5](#_Toc7905878)

[3.9.4 facade method 5](#_Toc7905879)

[3.9.5 strategy method 5](#_Toc7905880)

[3.9.6 state and memento method 6](#_Toc7905881)

[3.9.7 proxy method 6](#_Toc7905882)

[3.10 应用设计与开发 6](#_Toc7905883)

[3.10.1 TrackGame 6](#_Toc7905884)

[3.10.2 StellarSystem 8](#_Toc7905885)

[3.10.3 AtomStructure 8](#_Toc7905886)

[3.10.4 PersonalAppEcosystem 9](#_Toc7905887)

[3.10.5 SocialNetworkCircle 9](#_Toc7905888)

[3.11 应对应用面临的新变化 11](#_Toc7905889)

[3.11.1 TrackGame 11](#_Toc7905890)

[3.11.2 StellarSystem 12](#_Toc7905891)

[3.11.3 AtomStructure 12](#_Toc7905892)

[3.11.4 PersonalAppEcosystem 12](#_Toc7905893)

[3.11.5 SocialNetworkCircle 12](#_Toc7905894)

[3.12 Git仓库结构 13](#_Toc7905895)

[4 实验进度记录 14](#_Toc7905896)

[5 实验过程中遇到的困难与解决途径 15](#_Toc7905897)

[6 实验过程中收获的经验、教训、感想 15](#_Toc7905898)

[6.1 实验过程中收获的经验和教训 15](#_Toc7905899)

[6.2 针对以下方面的感受 15](#_Toc7905900)

# 实验目标概述

本次实验覆盖课程第3、5、6 章的内容，目标是编写具有可复用性和可维护性的软件，主要使用以下软件构造技术：子类型、泛型、多态、重写、重载，继承、代理、组合，常见的OO 设计模式，语法驱动的编程、正则表达式，基于状态的编程，API 设计、API 复用。

本次实验给定了五个具体应用（径赛方案编排、太阳系行星模拟、原子结构可视化、个人移动App 生态系统、个人社交系统），学生不是直接针对五个应用分别编程实现，而是通过ADT 和泛型等抽象技术，开发一套可复用的ADT 及其实现，充分考虑这些应用之间的相似性和差异性，使ADT 有更大程度的复用（可复用性）和更容易面向各种变化（可维护性）。

# 实验环境配置

实验环境配置已经在前面几个实验完成，这里不再赘述。

在这里给出你的GitHub Lab3仓库的URL地址（Lab3-学号）。

<https://github.com/ComputerScienceHIT/Lab3-1170800716>

# 实验过程

请仔细对照实验手册，针对每一项任务，在下面各节中记录你的实验过程、阐述你的设计思路和问题求解思路，可辅之以示意图或关键源代码加以说明（但千万不要把你的源代码全部粘贴过来！）。

## 待开发的三个应用场景

首先请列出你要完成的具体应用场景（至少3个，1和2中选一，3必选，4和5中选一，鼓励完成更多的应用场景）。

* 径赛场地赛程编排（TrackGame）
* 原子结构模型（AtomStructure）
* 社交网络的好友分布（SocialNetworkCircle）

分析你所选定的多个应用场景的异同，理解需求：它们在哪些方面有共性、哪些方面有差异。

共性：都是多轨道系统，都有轨道，所有物体附着在轨道上面，所有物体，包括轨道都有id。

差异：TrackGame没有中心点物体且每个轨道上只有一个元素，AtomStructure有中心点物体但是轨道上物体元素的类型都一样，且一个轨道上有多个元素。SocialNetworkCircle元素与元素之间有联系，而且轨道结构可以根据输入文件改变。

## 基于语法的图数据输入

这里我使用的是正则表达式，在一个静态方法里面，将数据按标签的方式读取出来，然后存入一个list里面传出来，详细代码如下：



图一 图数据输入函数

## 面向复用的设计：CircularOrbit<L,E>

这里我就参考了实验手册上面的方法将其设计为了一个接口，设计最初始的方法，然后再设计类去实现它。然后在后期根据需要，又往里面添加了几个必要的方法。

## 面向复用的设计：Track

这是一个非常简单的ADT，总共就两个数据，就是track本身的id以及这个track的半径，这里不再过多赘述。

## 面向复用的设计：L

这个代表中心点物体，由于这是一个泛型，所以在ConcreteCircularOrbit里面不能带有自己的属性，所以在ConcreteCircularOrbit里面都基本上是靠数据结构的支撑来实现相关操作的。

## 面向复用的设计：PhysicalObject

这个代表轨道物体，这是一个抽象类，其中包括了这个物体的所在轨道id也就是说创建这个物体的时候必须已知这个物体在哪一个轨道，里面的属性还有这个物体的所在轨道上面的角度，这样的话就可以计算出来该物体的绝对位置。当然，还有该物体的id来表示该物体的唯一的身份。

## 可复用API设计

这个里面我写了相当多的API，其中有一部分是可视化设计的元素，这里就先不介绍，在下一节里面再详细介绍相关内容。

首先是计算信息熵，这个就是完全套用公式，简单计算一下轨道上面的物体数量，然后计算一下每个轨道上面的物体对于所有物体的概率，然后对于每个轨道上面概率与log的概率相乘最后再加和，就是对于这个轨道系统来说的话就是信息熵。

然后是计算逻辑距离，这个只对第三个应用有用（当然按老师的说法可能有第四五六个），就是用了实验二里面的图方法，计算一下最短路径。

然后是计算物理距离，这个有一点玄学，因为每一个物体的位置都在输入文件里面没有给，当然可以计算相关出随机给出的位置的位置距离，但是这样的话完全没有意义，所以这里，我就是单纯计算了两个物体的轨道距离之差，这样的话比较好计算。

然后是获得差异，对于这个方法，我单独设计了一个Difference类来保存和输出差异，在实现这个方法的时候就是把每个轨道上的物体储存到一个Map里面，这个map就是一个简易的轨道，然后遍历这个轨道来获得并储存差异，最后保存在Difference里面。

然后就是CircularOrbitHelper里面的一些静态方法这个完全就是为了方便使用而打造的一些小方法来提高代码复用，以及调高效率。

首先是读取文件的函数readContent，这里就是上面的读取图输入，可以在所有的application里面使用，方便文件的读入以及设置数据结构。

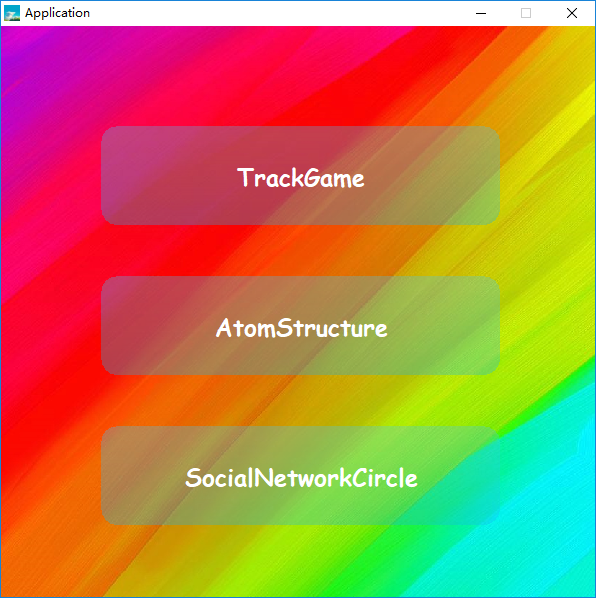
然后就是isDigital，设计这个方法原因是在图形化里面的输入框，有时候会输入一些奇奇怪怪的东西，这时候需要判断一下输入的东西是不是数字，然后再转化成数字再进行进一步的判断，否则的话，如果不是数字，很可能转化以后符号数值要求，然后会出现奇怪的结果。

Render这个函数就是单纯的包装函数，每次复制粘贴太麻烦了，就造了这么一个函数来使代码看起来短一点哈哈哈。

visualize这个函数，说实话，我一开始为了能够把所有的方法都放到可视化里面实现，就把可视化深度集成到了application里面，这个函数只是单纯地启动使得窗口可见而已，没有什么技术含量。

## 图的可视化：第三方API的复用

说实在的，手册里面给的第三方库，都用不成，自定义的程度都太低了，都不太能用到这个实验里面，都要去使用的话相当的麻烦，于是我索性就用比较初级的swing来实现了，这个swing自定义程度比较高，而且支持将函数托管出来，可以实现相当多得方法，相应的数据也可以比较容易的显示出来，比较适合这个实验，最重要的是比较好看哈哈哈，这个我还写了主方法，渲染了一下，在main函数里面实现，这里贴一张主函数的图，下面详细介绍应用的时候再介绍应用的图。



图二 主方法界面

## 设计模式应用

### factory method

在构建对象的时候，可以使用工厂模式来包装构造函数，这样的话对对象类型的选择就延迟到了子类来实现，这样的话就可以在构造一些简单的对象的时候使用工厂方法，使client在使用ADT的时候不用关心具体的类。

### abstract factory

在ConcreteCircularOrbit里面构造实例时并不知道对象的类型是什么，可以使用抽象工厂类，来将实例的创建延迟到子类执行，还可以绑定相关的关系。但是对于ConcreteCircularOrbit来说我设计的是一个抽象类，没有必要在抽象类里面创建实例，可以直接在子类里面创建后再加入，用抽象工厂反而会麻烦一些。

### Iterator method

这里我是继承了iterable然后在ADT里面根据轨道由内向外的顺序遍历，这里把所有的东西放到一个list里面，然后通过比较器比较轨道的id然后来进行排序，最后返回list的迭代器，这样的话就可以实现对所有的物体对于特定的顺序进行遍历。

### facade method

关于这里，就是我上面说的，就是把一些复杂的代码封装起来变成一个函数，来减少代码的复用，并且使得client端使用ADT更加简单和方便。具体内容上面已经说过了，这里就不再赘述了。

### strategy method

这个就是实现对于方案选择不同的算法，在TrackGame里面，我分别写了两个函数对于一个数据结构进行不同的操作，最后再将这个数据结构表示出来，最终可以达到我想要的效果。

### state and memento method

在AtomStructure里面，我是使用了一个map来存放整个轨道的数据信息的，在修改轨道的时候，就修改该数据结构就可以了，基于这个思路，我每次把这个数据结构拷贝保存一份到备忘录里面，需要的时候恢复就可以了。

### proxy method

这个主要体现在我的gui里面，由于我只能吧jframe提供给我的客户端，而jpanel无法提供，我就只能通过jframe来代理jpanel上面的操作，来达到咋jpanel上面画图的效果，这里和adoptor模式有一点像，可以说两种都用到了吧，下面就不说adopter了。

## 应用设计与开发

利用上述设计和实现的ADT，实现手册里要求的各项功能。

以下各小节，只需保留和完成你所选定的应用即可。

### TrackGame

这个就是先用读入函数将数据读取进去，然后我用了一个List来保存出场顺序，先初始化为从id开始到结尾，然后在用一个数据结构保存相关分数。根据这个分数来排序seq或者随机打乱seq，然后根据这个seq来构造运动员并将其放到轨道上面，每次修改都初始化，然后通过seq来构造轨道，这样的话，Change也可以简单地只需要修改seq就可以了。

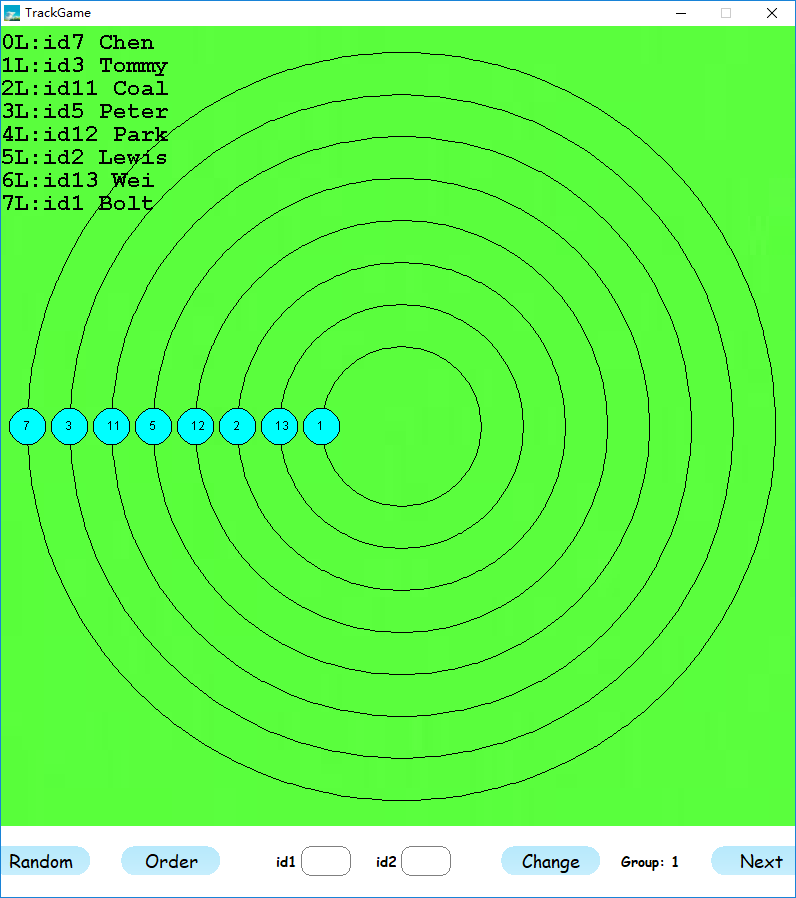


图3 TrackGame界面1

在这个图形化里面，我集成了所有的方法（312change之前的），可以切换分组，可以排序，可以使用两种算法，下面是切换分组的演示：

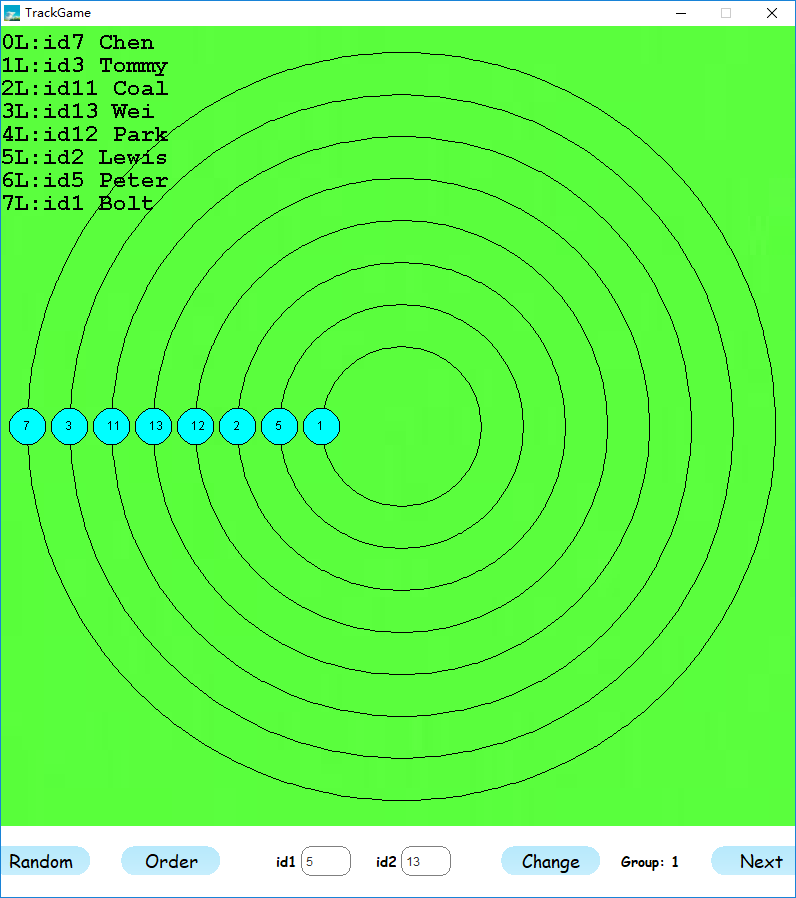
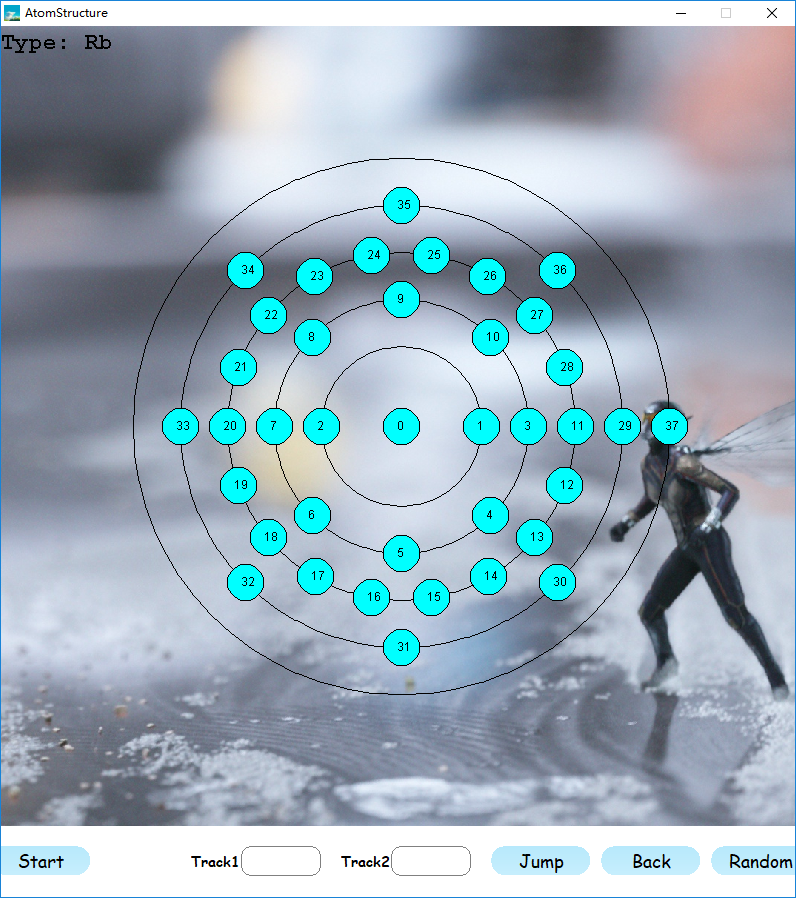


图4 TrackGame界面2

### StellarSystem

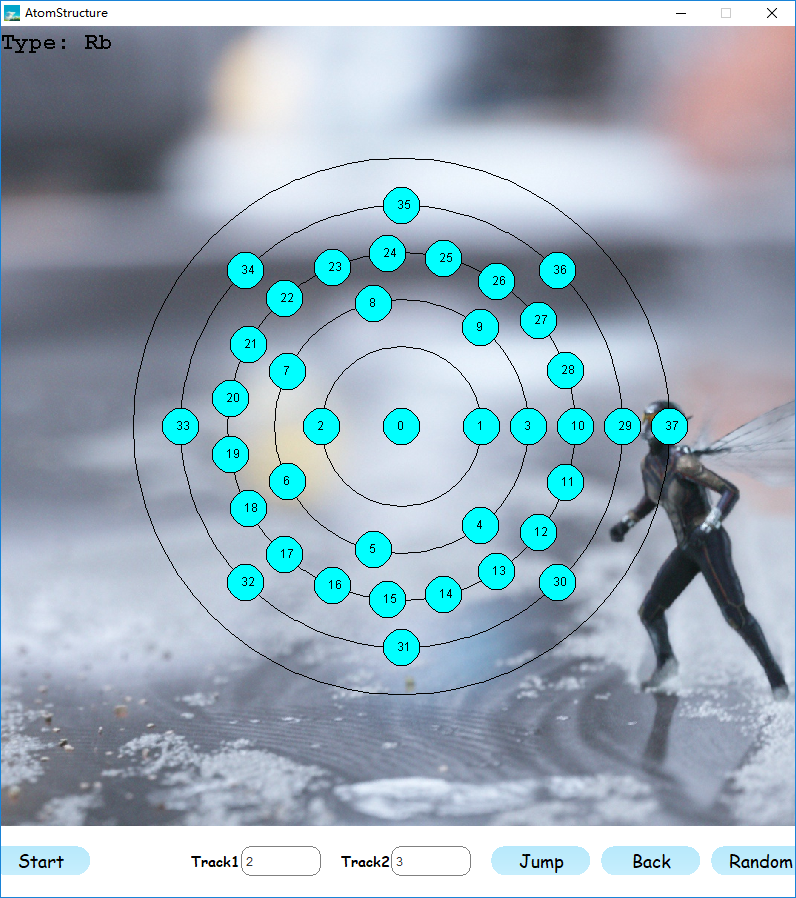
### AtomStructure

这个就更简单了，也是把数据读入后，存入一个数据结构里面，然后每次更改就通过修改这个数据结构，然后初始化重构：



图五 AtomStructure界面一

这里也和上面一样，所有功能都可以在图形化里面实现，可以做电子的跃迁，也可以备忘录模式恢复，当然只能恢复到上一步，顺便说一下这个背景图是蚁人，漫威的英雄，可以缩小到量子级，想到就刚好来当做原子结构的背景图啦哈哈哈。然后下面是跃迁的测试图：

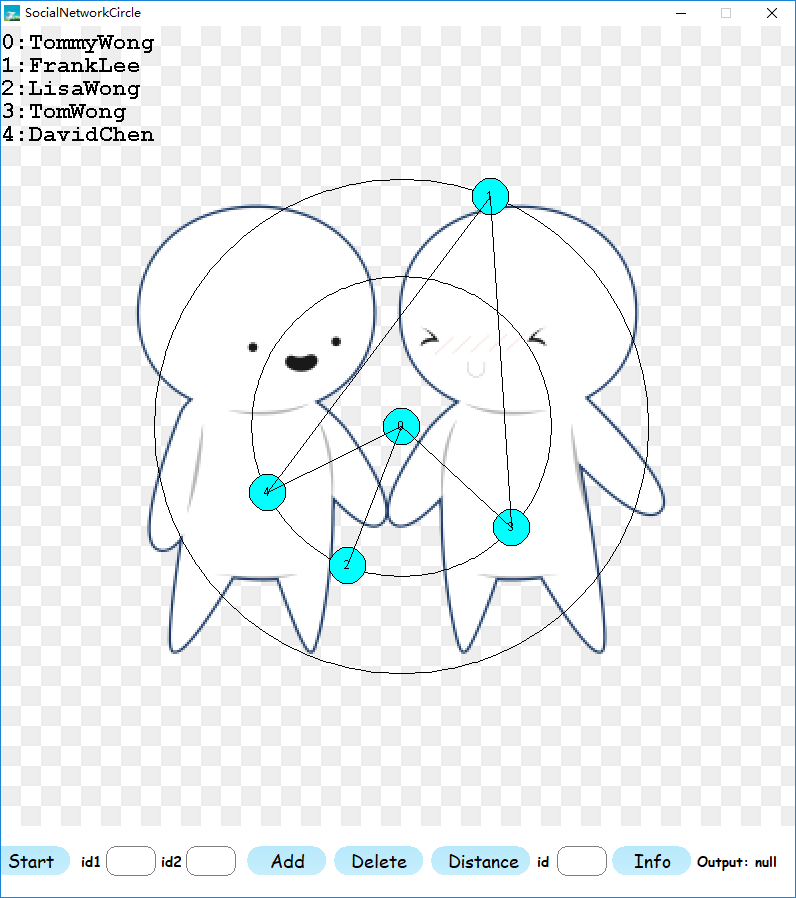


图六 AtomStructure界面二

### PersonalAppEcosystem

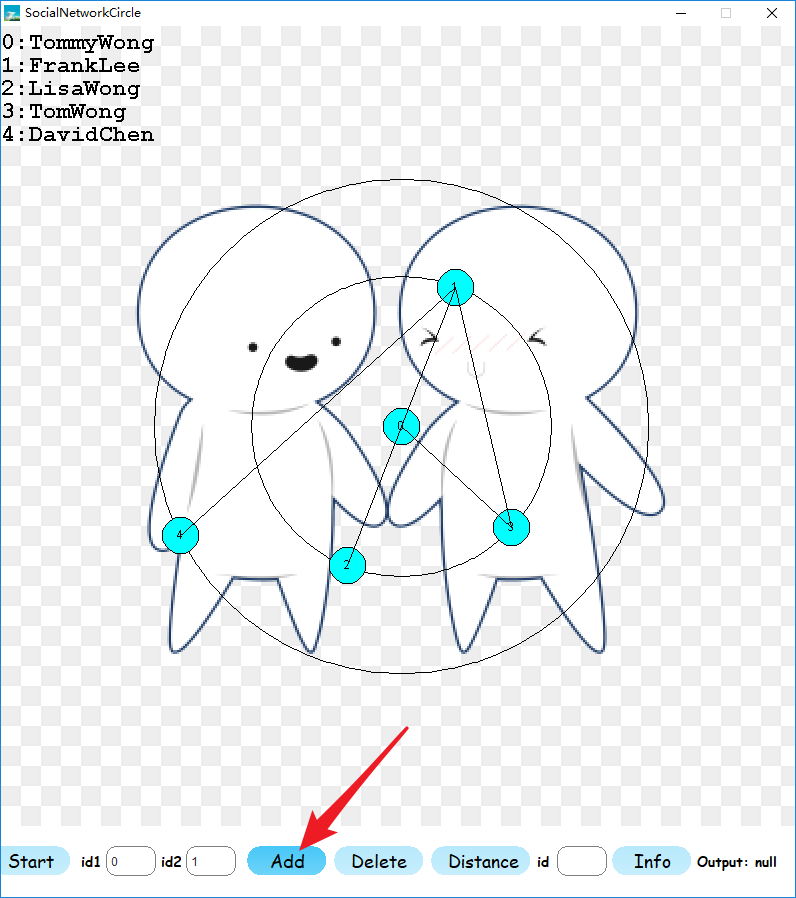
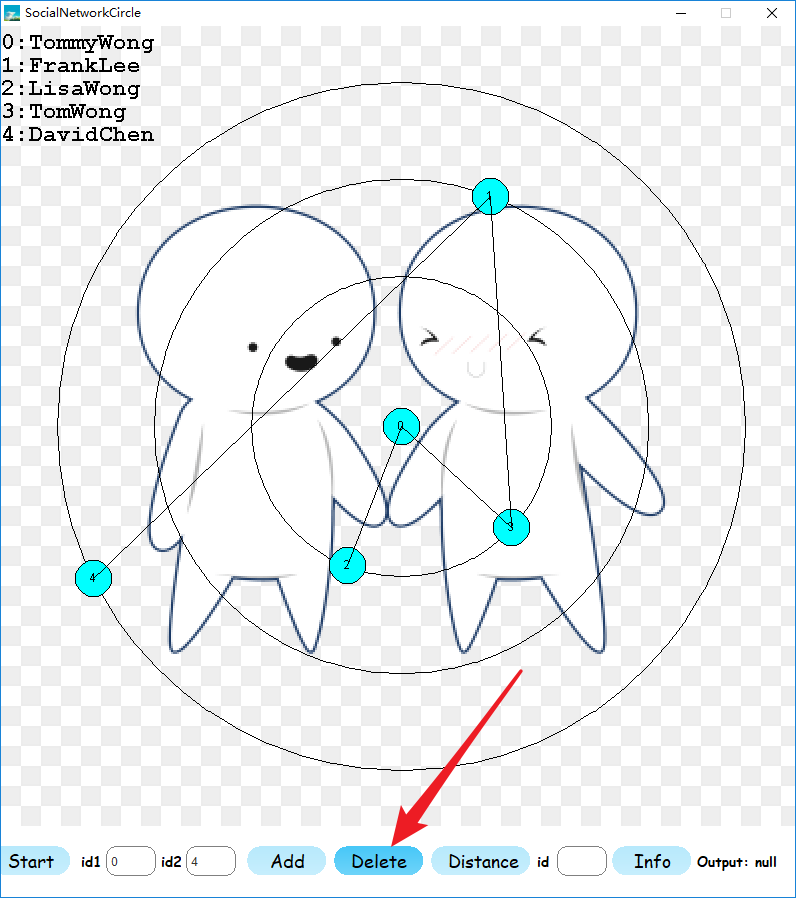
### SocialNetworkCircle

这里是最难的一个应用，因为这个里面的数据结构相当复杂，需要各个人之间的关系等等，这里我是借用了实验二的图结构，将每个人的名字存入了图中，然后根据名字来获取到每个人的关系来构建图，其他的就和上面的产差不多了，后期的修改以及重构都是在数据结构上面修改，然后初始化，然后重新构建所有的点，存入轨道中。



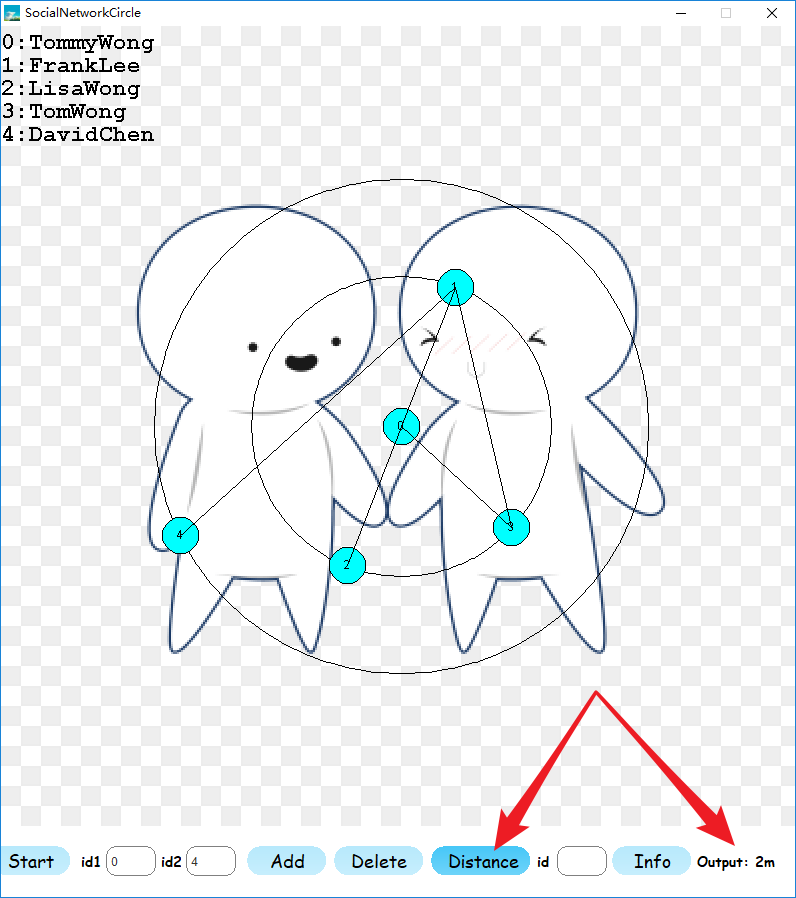
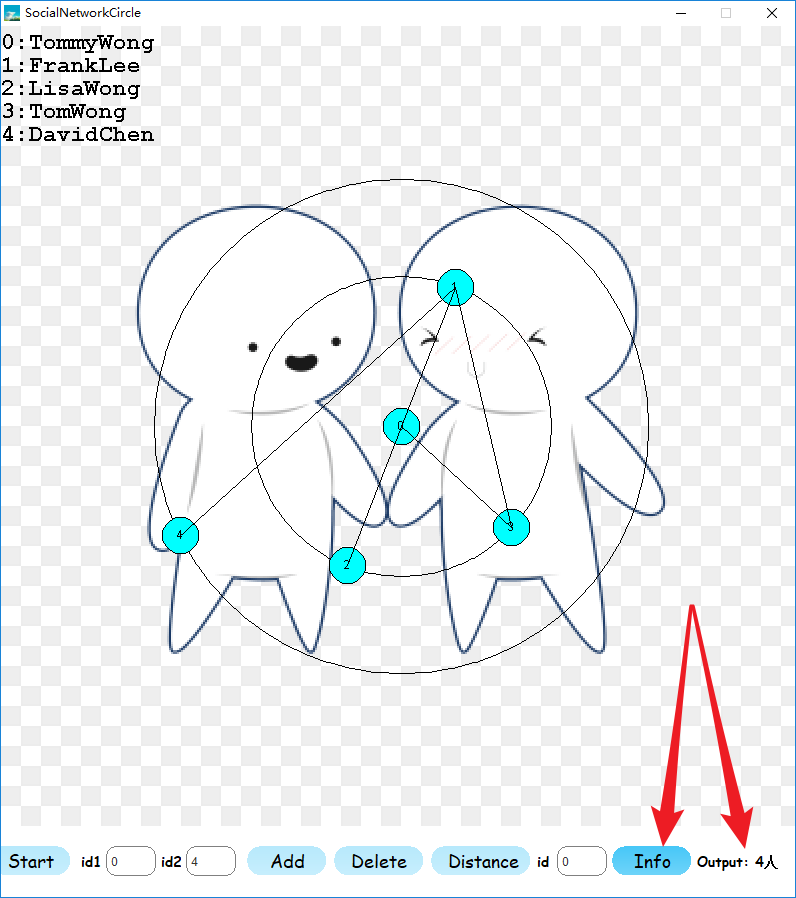
图七 AtomStructure界面一

不难看出，这个里面的功能相当复杂相当的多，add和delete的作用是删除和增加关系，然后distance的计算逻辑距离，输出就在右边的标签可以读取，然后右边的info就是可以认识多少个人，在右边输入，然后点击按钮，也可以从右边的输出可以看到相关人数（可以看得出来，空间有点小不允许两个输出，就写到一起了）下面是相应的测试界面：



图八 AtomStructure界面二

这里就是删除和增加一条关系，可以看出关系的联系不见了，然后图也进行了重构。

图九 AtomStructure界面三

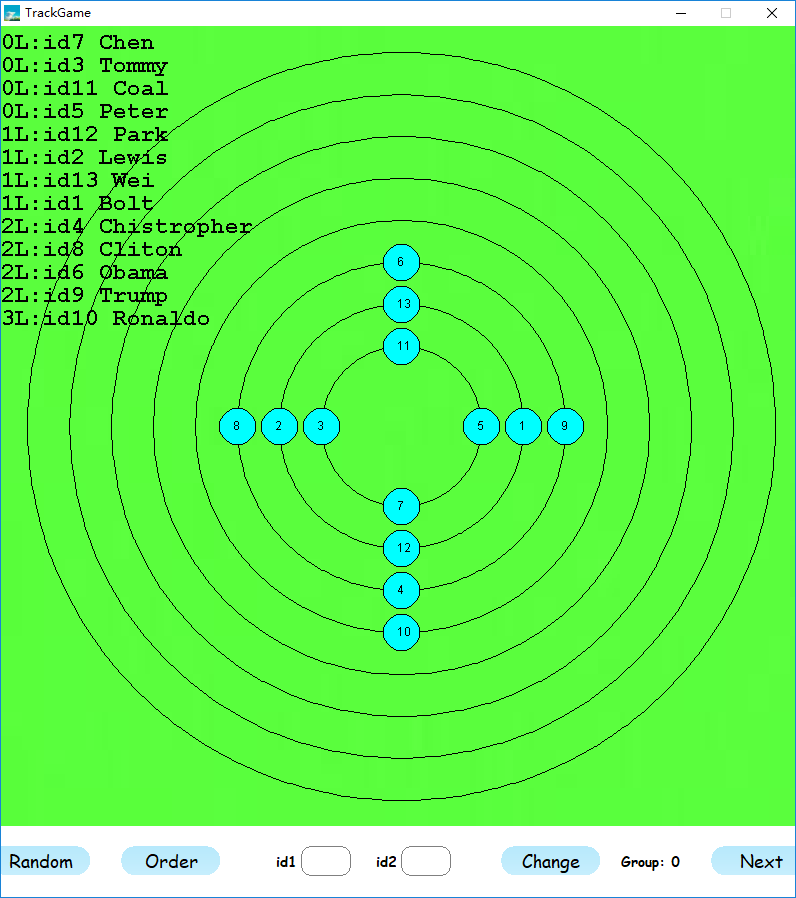
可以看出这里可以计算距离和信息扩散度。

## 应对应用面临的新变化

以下各小节，只需保留和完成你所选定的应用即可。

### TrackGame

这里就简单地增加了一个构造函数，本来的构造函数是通过seq每个轨道一个人放置到上面，现在是放置四个人，这个由于可能会和排序冲突就没有加到可视化里面，不过可以测试一下，下面的是测试画面。



图十 TrackGame312change测试

### StellarSystem

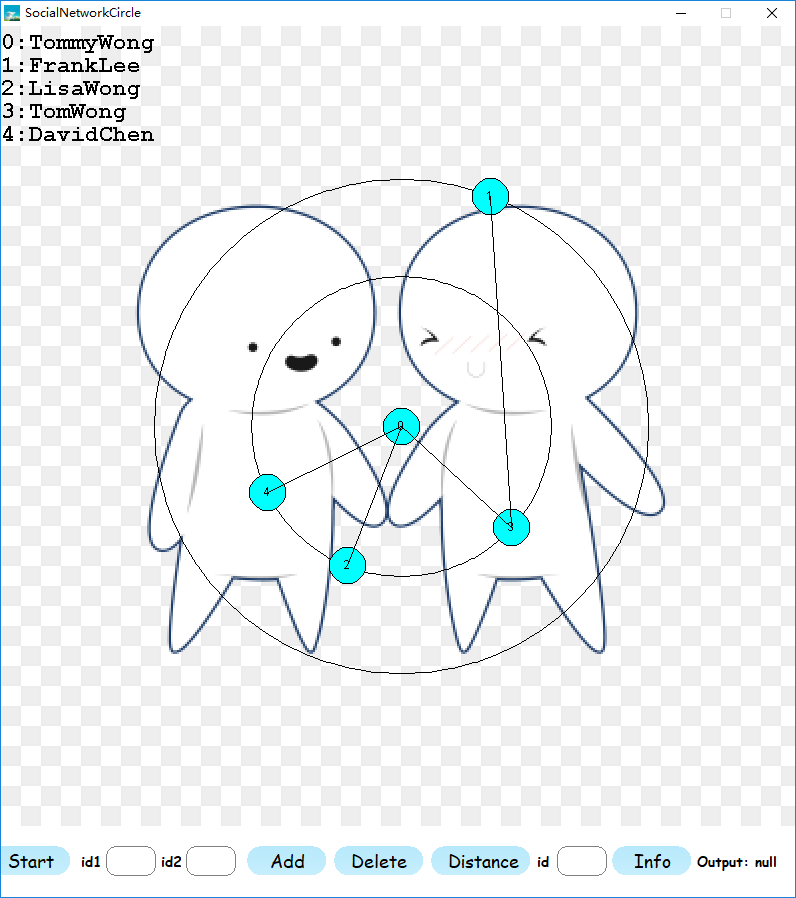
### AtomStructure

这个就更简单了，只需要修改一下中心物体的rep，一开始是只有一个kinds表示含有的原子种类，现在我把这个kind改成了kinds，这是一个Set里面可以存放许多种类，就实现了手册所需要的效果。

### PersonalAppEcosystem

### SocialNetworkCircle

这个一开始我也晕了一下，手册里面描述的只包含中心点向外的外交关系，我以为只保存第一层轨道，后来发现不是，是只保存向外的关系，这样的话，我可以先把图存下来，有了数据结构以后，我再根据原始输入文件，判断哪些不符合条件，利用已经实现的Delete函数，将不符合条件的关系删掉后，在初始化后重构，就达到了相应的效果。下面是通过我增加的Trim函数构造的只有向外关系图等等应用：

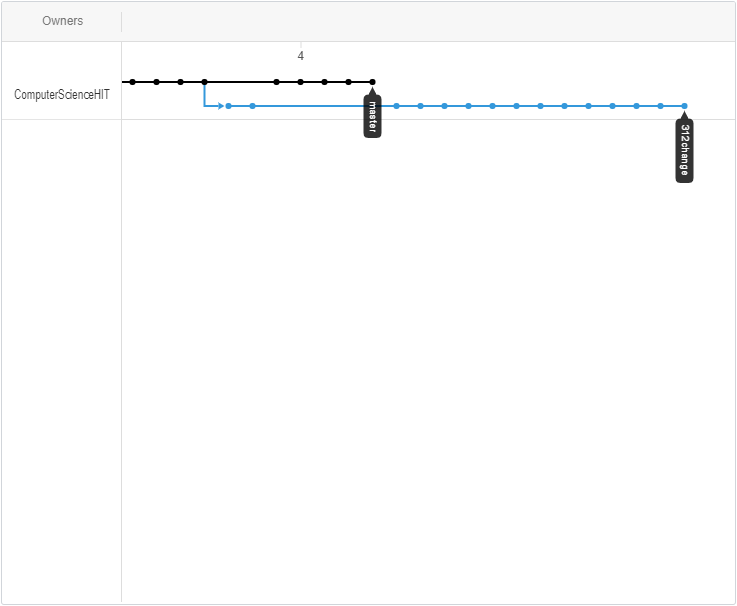


图十一 SocialNetwork312change测试

通过输入文件不难看出该图修改正确。

## Git仓库结构

请在完成全部实验要求之后，利用Git log指令或Git图形化客户端或GitHub上项目仓库的Insight页面，给出你的仓库到目前为止的Object Graph，尤其是区分清楚312change分支和master分支所指向的位置。



图十二 Object Graph

# 实验进度记录

请使用表格方式记录你的进度情况，以超过半小时的连续编程时间为一行。

每次结束编程时，请向该表格中增加一行。不要事后胡乱填写。

不要嫌烦，该表格可帮助你汇总你在每个任务上付出的时间和精力，发现自己不擅长的任务，后续有意识的弥补。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 2019/4/8 | 19:00-22:00 | 完成文件读取的函数 | 按计划完成 |
| 2019/4/12 | 19:00-23:00 | 完成GUI的选择和设计 | 未按计划完成 |
| 2019/4/13 | 18:00-23:00 | 决定GUI库并开始设计 | 按计划完成 |
| 2019/4/22 | 19:00-23:00 | 接口设计以及SPEC撰写 | 按计划完成 |
| 2019/4/23 | 19:00-23:00 | 尝试第一个APP | 计划进行中 |
| 2019/4/26 | 19:00-23:00 | 完成第一个APP | 未按计划完成 |
| 2019/4/29 | 19:00-23:00 | 完成第二个APP | 按计划完成 |
| 2019/4/30 | 19:00-23:00 | 完成第三个APP | 未按计划完成 |
| 2019/5/1 | 13:00-23:30 | 完成第三个APP | 未按计划完成 |
| 2019/5/2 | 9:00-23:30 | 完成第三个APP | 按计划完成 |
| 2019/5/3 | 9:00-23:30 | 代码注释以及修改 | 按计划完成 |
| 2019/5/4 | 9:00-23:30 | 312change以及报告撰写 | 按计划完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 选择第三方绘图库,在StackOverflow上面看了很多,一开始使用javafx,发现要做到函数外置很困难 | 在一番尝试以后,发现基础的Swing可以实现相关功能,于是决定使用Swing来一点一点来实现绘图的功能,代码量还是不小的,但是在绘图库里面用了许多的设计模式,也算练习了. |
| app3里面的数据结构有点复杂,构造比较容易,但是在删除关系的时候总是出现一些奇怪的问题,目前还是不知道怎么去解决. | 后来发现是数据结构里面有一些rep出现了问题，后来使用checkrep检查错误以及调试过程中慢慢修改，终于解决了这个问题，最后程序可以正常运行了。 |
|  |  |

# 实验过程中收获的经验、教训、感想

## 实验过程中收获的经验和教训

1.看完实验手册再写实验，不然后面会改改改，特别难受。

2.还有的时候，会get不到想要做的点是什么，然后就会走偏。

3.然后就是，一定要尽早做，不然太后面有点点难受。

4.有的时候，会特别的崩溃，觉得自己做不下去了，这个时候一定不要放弃，当然可以休息一会做做其他的事情，但是不可以太久，等放松一点，就回来做，虽然不敢说一定会有结果，但是基本上都会有好的解决方案。不放弃。

## 针对以下方面的感受

1. **重新思考Lab2中的问题：面向ADT的编程和直接面向应用场景编程，你体会到二者有何差异？本实验设计的ADT在五个不同的应用场景下使用，你是否体会到复用的好处？**

面向ADT的编程会考虑的少一点，就是一个抽象结构的构造，出错的概率还是比较少的，然而面向应用编程的话就会出现一系列不同的问题，然后应用的结构也比较复杂，编写的时候需要多注意一下复用和健壮性可以大大降低工作量和错误发生的可能性。

1. **重新思考Lab2中的问题：为ADT撰写复杂的specification, invariants, RI, AF，时刻注意ADT是否有rep exposure，这些工作的意义是什么？你是否愿意在以后的编程中坚持这么做？**

这些工作的意义是为了防止在程序运行的时候出现一些奇怪的问题，在编写的时候就尽量发现这些问题，不要等到交付到用户手上的时候再出现不变量泄露这些问题可就真的麻烦啦，不过话又说回来，大型协作项目使用这个是极好的，我个人觉得小项目自己写的话，写注释会更加方便自己懂一点，然后就是我觉得无会在以后的编程中使用这些的。

1. **之前你将别人提供的API用于自己的程序开发中，本次实验你尝试着开发给别人使用的API，是否能够体会到其中的难处和乐趣？**

难啊，相当难，要考虑到别人可能会用到什么方法，要在保证数据的可用性，要防止出现各种奇奇怪怪的问题，尽量为客户端考虑，说乐趣也是有的，就是这个东西写出来以后别人就可以使用了，作为一名不算程序员的一个小小的大学生来说，能有自己的产品也是很快乐的。

1. **在编程中使用设计模式，增加了很多类，但在复用和可维护性方面带来了收益。你如何看待设计模式？**

设计模式在构造大型的应用里面起到了举足轻重的作用，这些设计模式可以解决许多的在开发中的遇到的问题，并不是简单地在复用和可维护性方面带来了收益，而是有一些不用设计模式根本达不到想要的功能，我觉得设计模式相当有用。

1. **你之前在使用其他软件时，应该体会过输入各种命令向系统发出指令。本次实验你开发了一个解析器，使用语法和正则表达式去解析输入文件并据此构造对象。你对语法驱动编程有何感受？**

我觉得挺好的，可以直接从配置文件读入数据，而不用每次手动输入数据，方便的多，我猜一些软件的配置也是从文件读入然后进行配置的，这些设置可能会相当复杂但是用一个文件再让软件自动读入的话就会方便很多很多。

1. **Lab1和Lab2的大部分工作都不是从0开始，而是基于他人给出的设计方案和初始代码。本次实验是你完全从0开始进行ADT的设计并用OOP实现，经过三周之后，你感觉“设计ADT”的难度主要体现在哪些地方？你是如何克服的？**

我感觉难度就在于，如何设计数据结构，而抽象数据型（ADT）是一个相当尴尬的东西，如果直接是应用的话，我可以直接根据我需要什么就去造什么，很容易想，但ADT不是，你要去猜用户需要什么，然后再去设计，实现，这中间的断带有点大，就是不好想。我克服的方法就是我自己先写应用，然后，看我自己需要什么，然后再往里面加，可能本实验我量比较少，但是，我确信在大量的迭代下，我设计的ADT一定会越来越完美的。

1. **你在完成本实验时，是否有参考Lab4和Lab5的实验手册？若有，你如何在本次实验中同时去考虑后续两个实验的要求的？**

讲真，没咋参考，一开始一头雾水，最终的产品都是一次次迭代出来的，就没参考，不过我在写的过程中有考虑到健壮性和效率的问题，后面应该问题不是特别大。

1. **关于本实验的工作量、难度、deadline。**

工作量，大的过分了。这一个月是期中考试和一些考试的集中周，非要把这么大头放到这个时间，真的是太难受了，就不能量少一点细嚼慢咽呢，或者说时间长一点也行呢。难度嘛，中上吧。

1. **到目前为止你对《软件构造》课程的评价。**

很有用的一门课，但是这个实验的量的设置，有点过于多了，可以考虑考虑细嚼慢咽学更多知识。