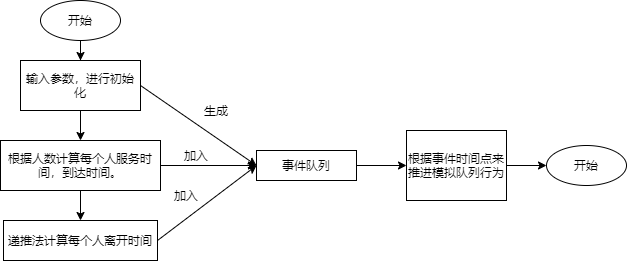
**队列模型实验报告**

1. **实验目的**

应用M/M/1队列和M/M/N编程思想，模拟单窗口排队和多窗口排队的过程，熟悉离散事件推进方式、队列建立和提取方式。

1. **实验需求**
   1. 应用M/M/N 队列编程思想，采用离散事件推进方式
   2. 能输入多个参数（平均到达时间，平均服务时间，顾客数目，服务窗口数目，最大顾客人数）
   3. 有动态输出UI展示（当前时间，队列中平均等待客户数，服务器利用率）
2. **思路概述  
   因为选做了M/M/N，而M/M/1则是M/M/N中N = 1的特殊情况，故只分析M/M/N**
   1. **整体数学建模思路**：  
       M/M/N模型即多窗口单队列排队模型，新来的成员会选择当前排队时间最少的队列加入队列，开始排队。同时，采用下一事件时间推进算法，即预先计算出事件发生的时间（如到达时间，服务时间等），根据事件发生的顺序向后推进，来模拟整个队列过程。  
       流程图如下图：  
      

* 1. **计算关键信息思路**：
     1. 平均逗留时间 = 所有顾客的逗留时间之和 / 顾客数。

逗留时间 = 顾客离开时间 – 顾客到达时间

* + 1. 队列中的平均顾客数：队列面积 / 当前时间  
       队列面积 += 排队人数 \* 排队间隔时间(time\_since\_last\_event)
    2. 服务器利用率 = 服务器状态面积 / 当前时间(current\_time)  
       服务器状态面积 += 服务器状态 ( 0 或1 ) \* 服务间隔时间

**4．**顾客离开时间 = 前一位顾客离开了：顾客到达时间+顾客服务时间  
前一位顾客未离开：前一位顾客离开时间 + 顾客服务时间

* 1. **建模细节**：
     1. 基本假设
        1. 不考虑窗口状态切换的时间。即该窗口没有排队人数的时候，可以立刻切换为空闲
        2. 接受完服务的顾客会直接离开，不会停留在队列中
        3. 新来的顾客会立刻选择队伍开始排队
     2. 时间的生成**：**

顾客到达时间间隔和顾客服务时间均采用指数分布生成。指数分布的重要特征是无记忆性，是一种连续概率分布，经常用来表示独立随机事件发生的时间间隔。比如电话的通话时间、电子元件的寿命等，所以很适合本题中生成服务时间以及到达时间间隔。

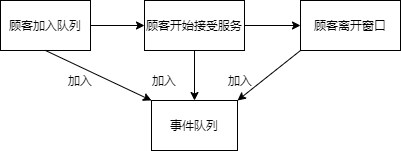
对于指数分布，随机变量x服从参数λ>0的指数分布，其概率密度函数为;

对于其无记忆属性的特点，有以下概率分布规律：

* + 1. 显示事件推进节点：

本模型采用下一事件时间推进法。根据事件的时间来推进仿真的过程，节省了计算机的运行时间。而且，在事件推进时，各模型异步执行，而不是同步执行。

推进的节点有：顾客加入队列，顾客开始服务，顾客服务结束离开队列。如图所示



* + 1. 显示简述事件调度法、活动扫描法和进程交互法的异同：
       1. 事件调度法：

将事件例程作为仿真模型的基本模型单元，按照事件发生得先后顺序不断执行相应的事件例程。每一个事先可预知的事件和确定事件都带有一个事件例程，用于处理事件发生后对实体状态所产生的影响。

* + - 1. 活动扫描法：

使用固定时间间隔和基于规则的方法来决定是否开始一个活动。在每一个单位时间步长，检查条件，如果是真，则触发相应活动。

* + - 1. 进程交互法：

调度执行两类时序的实体事件列表:未来事件列表和当前事件列表。具体方式为：扫描未来事件列表，找到移动记录的实体，将当前时间表中的实体向前推进直到进程被锁起。

* + - 1. 区别：

**事件调度法**：面向事件，采用异步推动。以时间为考虑因素，在处理事件时候需要考虑附加影响。

**活动扫描法：**面向活动，采用同步推动。以相应规则为考虑因素。

**进程交互法：**面向进程，采用异步推动。关注实体所经历的事件。

* + - 1. 相同之处：

均按照时间顺序发生，仿真时钟按照时间顺序向前推进。均需要对事件进行提前的判断。

1. **实验验证与对比实验猜想**
   1. **数据分析：验证实验**

这里进行对比实验来对模型进行检验，分别给出10次在本地模型上跑的结果和理论结果，来进行对比，验证模型准确性

* + 1. **实验一：**

平均到达时间为5，平均服务时间为2.5，顾客数目为200，服务器数目为1的情况。

理论值：

|  |  |  |  |
| --- | --- | --- | --- |
| 累计耗时 | 平均等待客户数 | 平均等待时间 | 窗口利用率% |
| 502.50 | 0.00 | 2.50 | 50 |

实验值：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 累计耗时 | 平均等待客户数 | 平均等待时间 | 窗口利用率% |
| 1 | 556.96 | 1.13 | 4.29 | 44 |
| 2 | 520.57 | 2.04 | 8.84 | 60 |
| 3 | 443.94 | 1.63 | 6.56 | 61 |
| 4 | 491.62 | 0.34 | 3.02 | 52 |
| 5 | 475.35 | 0.85 | 5.47 | 0.60 |
| 6 | 473.27 | 0.96 | 5.05 | 0.53 |
| 7 | 482.64 | 0.81 | 4.90 | 0.55 |
| 8 | 497.71 | 0.50 | 3.93 | 0.50 |
| 9 | 488.06 | 1.26 | 7.06 | 0.60 |
| 10 | 503.91 | 0.94 | 5.77 | 0.58 |

结论：基本符合理论值，在适当的范围内略有波动

* + 1. **实验二：**

平均到达时间为5，平均服务时间为10，顾客数目为100，服务器数目为1的情况。

理论值：

|  |  |  |  |
| --- | --- | --- | --- |
| 累计耗时 | 平均等待客户数 | 平均等待时间 | 窗口利用率% |
| 1005.00 | 25.13 | 257.50 | 100 |

实验值：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 累计耗时 | 平均等待客户数 | 平均等待时间 | 窗口利用率% |
| 1 | 919.86 | 18.63 | 174.06 | 99 |
| 2 | 925.18 | 22.05 | 205.32 | 99 |
| 3 | 1125.08 | 21.44 | 250.31 | 99 |
| 4 | 838.73 | 26.69 | 231.18 | 99 |
| 5 | 943.60 | 26.19 | 243.14 | 100 |
| 6 | 972.03 | 26.61 | 258.19 | 100 |
| 7 | 854.47 | 31.34 | 266.57 | 100 |
| 8 | 956.22 | 26.02 | 238.60 | 100 |
| 9 | 947.38 | 25.93 | 245.82 | 99 |
| 10 | 876.87 | 15.75 | 129.02 | 99 |

结论：与理论模型贴近，可以验证M/M/1模型基本正确。

* + 1. **实验三**

平均到达时间为5，平均服务时间为10，顾客数目为100，服务器数目为5的情况（测试M/M/N模型）

理论值：

|  |  |  |  |
| --- | --- | --- | --- |
| 累计耗时 | 平均等待客户数 | 平均等待时间 | 窗口利用率% |
| 510.00 | 0.00 | 10.00 | 40 |

实验值：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 累计耗时 | 平均等待客户数 | 平均等待时间 | 窗口利用率% |
| 1 | 509.2 | 2.15 | 11.62 | 37.6 |
| 1 | 545.08 | 0.55 | 12.41 | 40.2 |
| 2 | 573.20 | 0.19 | 9.36 | 41.2 |
| 3 | 586.96 | 1.24 | 15.43 | 43.1 |
| 4 | 536.94 | 1.31 | 15.79 | 39.8 |
| 5 | 503.55 | 1.20 | 15.51 | 38.1 |
| 6 | 598.43 | 0.43 | 11.45 | 43.2 |
| 7 | 484.64 | 1.47 | 14.59 | 37.5 |
| 8 | 473.22 | 0.69 | 11.43 | 37.3 |
| 9 | 441.25 | 2.84 | 20.47 | 36.2 |

结论：基本符合理论模型，证明M/M/N队列基本正确

。

* 1. **数据分析：对比试验**

因为服务时间和到达时间均与指数相关的随机生成函数有关，所以在人数数量多的时候较为稳定。因此，我才用顾客人数总为200。

1. **实验一**

当我们控制住平均服务时间不变时取平均服务时间为30，测试n=5的结果，测试平均到达时间对其他变量的影响：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 平均到达时间 | 平均服务时间 | 利用率 | 平均等待时间 | 平均等待人数 |
| 5 | 30 | 99 | 222 | 33 |
| 10 | 30 | 60 | 31 | 3 |
| 15 | 30 | 44 | 30 | 2 |
| 20 | 30 | 39 | 31 | 1.5 |
| 25 | 30 | 43 | 30 | 1.5 |
| 30 | 30 | 39 | 25 | 1 |
| 窗口数为5，人数为200时 | | | | |

可以看出来，当平均到达时间与平均服务时间接近时，利用率、平均等待时间、平均等待人数变化不大。当平均到达时间成倍小于平均服务时间时，利用率、平均等待时间，平均等待人数也会成倍增长。

所以说，当平均到达时间接近或这大于平均服务时间时，排队效率会有显著的提升。

1. **实验二**

由实验一的反应，我不禁好奇窗口数n与其他变量的关系，我们控制平均到达时间和平均服务时间分别为 15，30，人数为5，窗口数n从1到5分别测试：

|  |  |  |  |
| --- | --- | --- | --- |
| 窗口数 | 利用率 | 平均等待时间 | 平均等待人数 |
| 1 | 100 | 1572 | 50 |
| 2 | 90 | 115 | 7 |
| 3 | 70 | 45 | 2 |
| 4 | 50 | 30 | 1.5 |
| 5 | 35 | 25 | 1 |
| 平均到达时间：15，平均服务时间30，人数200 | | | | |

由实验结果可知：当n=1时，排队人数过多，结果不具有参考性，当n>=1时，有下图所示结果

利用率与窗口数成线性关系，而平均等待时间、平均等待服务人数与窗口数成不明显的线性关系。

1. **实验三**

最后，我还想探究一下，当人数为200，窗口数为5时，控制平均到达时间为15s，对其余变量的影响。因为人数为200时，准确性可以得到保证，窗口数为5时，利用率不会溢出

|  |  |  |  |
| --- | --- | --- | --- |
| 平均服务时间 | 利用率 | 平均等待时间 | 平均等待人数 |
| 5 | 7 | 5 | 0.8 |
| 10 | 15 | 10 | 1.1 |
| 15 | 22 | 16 | 1.3 |
| 20 | 29 | 20 | 1.5 |
| 25 | 34 | 25 | 1.6 |
| 平均到达时间：15，窗口数为5，人数200 | | | | |

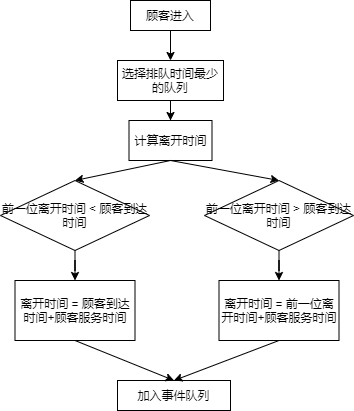
从图中可以看出，当服务时间与到达时间没有巨大差异的时候，平均服务时间与其余变量确实有**明显的线性关系。**

1. **总结：**
   * + 1. 当控制变量时，平均服务时间与利用率、平均等待时间、平均等待人数都有较为明显的正向线性关系
       2. 当控制变量时，平均到达时间与平均服务时间接近时，其余数据无明显的变化，当平均到达时间成倍数小于平均服务时间时，会有线性倍数的变化。
       3. 当控制变量时，窗口数与利用率是负线性相关，即窗口数越多，利用率越低。与其他两个变量没有明显的线性变化关系。
2. **编程实现与编程思路**
   1. **程序设计思路：**  
       整体上，分为用户、窗口、和模拟程序三个类。分别模拟用户的行为，队列的行为和整体时间线。
      1. 用户：

用户有属性：个人ID、到达时间、服务时间、离开时间、排队时间、等待时间（离开时间-到达时间）、加入的窗口号。

1. **class** Person:
2. **def** \_\_init\_\_(self,id, arrive\_time, serve\_time):
3. self.id = id #个人ID
4. self.arrive\_time = arrive\_time  # 到达时间
5. self.serve\_time = serve\_time  # 服务时间
6. self.leave\_time = 0  # 离开时间
7. self.queue\_time = 0  # 排队时间
8. self.wait\_time = 0  # 等待时间（从来到走）
9. self.table\_number = 0  # 加入的窗口号

用户行为如以下流程图所 **(此处与离开事件图一致)**

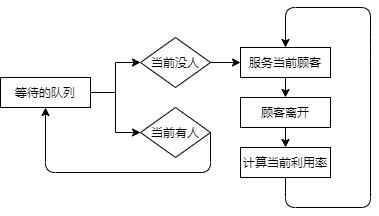


* + 1. 窗口：

窗口属性：结束时间、所有排队过的客户、当前队列的客户、是否正在被使用、利用率、服务总时间。代码如下。

1. **class** Table:  # 窗口
2. **def** \_\_init\_\_(self):
3. self.finish\_time = 0  # 结束时间
4. self.peopleList = []  # 总排队人次
5. self.queue = []  # 当前队列
6. self.is\_use = 0  # 是否正在使用 使用为1，不使用为0
7. self.use\_ratio = 0  # 利用率
8. self.serve\_time = 0  # 服务总时间

窗口行为如以下流程图所 **此处与进入队列配图一致**



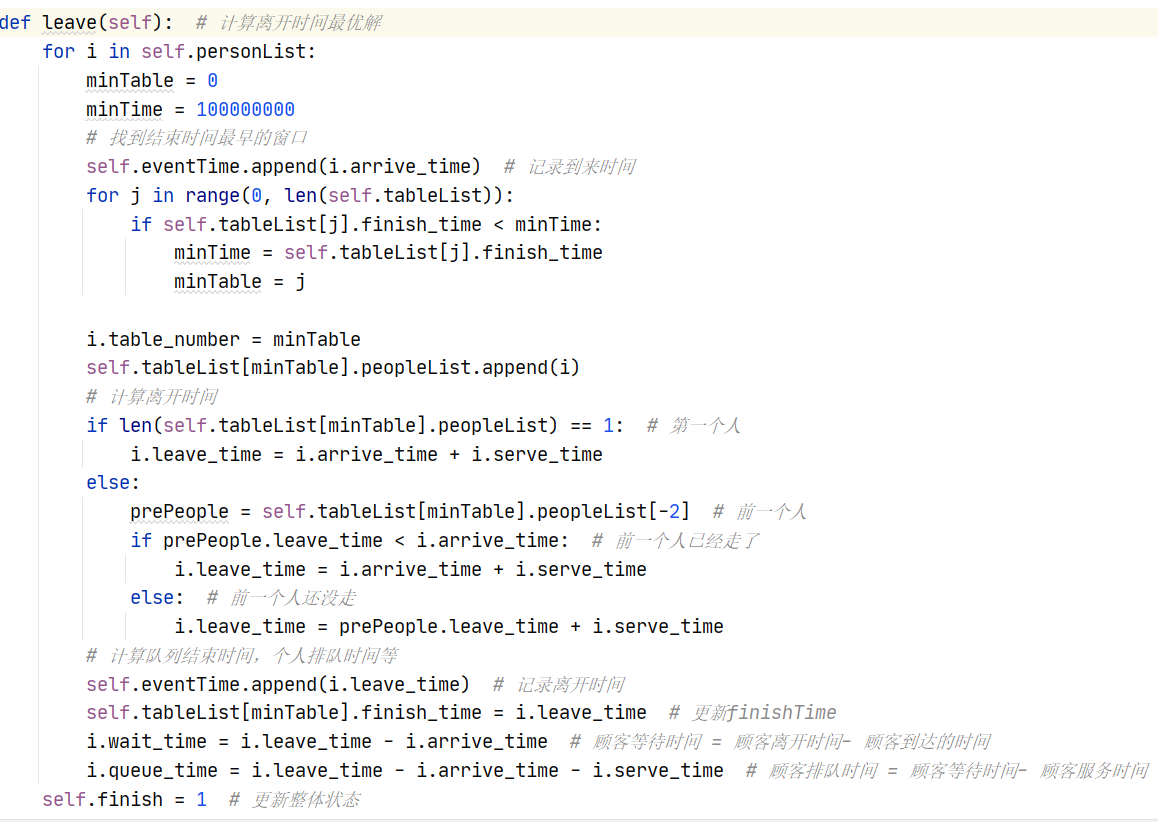
* 1. **后端程序具体实现：**
     1. 生成指数分布的时间间隔:  
        这里负指数分布我们使用python.random中random函数配合maht.log。

1. # 生成指数分布的时间间隔
2. **for** i **in** range(0, self.custom\_num):
3. arrive\_time = -1 \* self.average\_arrive\_time \* math.log(random.uniform(0, 1))
4. self.arrive\_space.append(arrive\_time)
   * 1. 计算到达时间:  
        到达时间 = 上一人到达时间 + 到达时间间隔
5. # 到达时间 = 上一人到达时间 + 到达时间间隔
6. self.arrive\_time.append(self.arrive\_space[0])
7. **for** i **in** range(1, self.custom\_num):
8. self.arrive\_time.append(self.arrive\_time[i - 1] + self.arrive\_space[i])
   * 1. 生成服从指数分布的服务时间

指数分布生成如1。

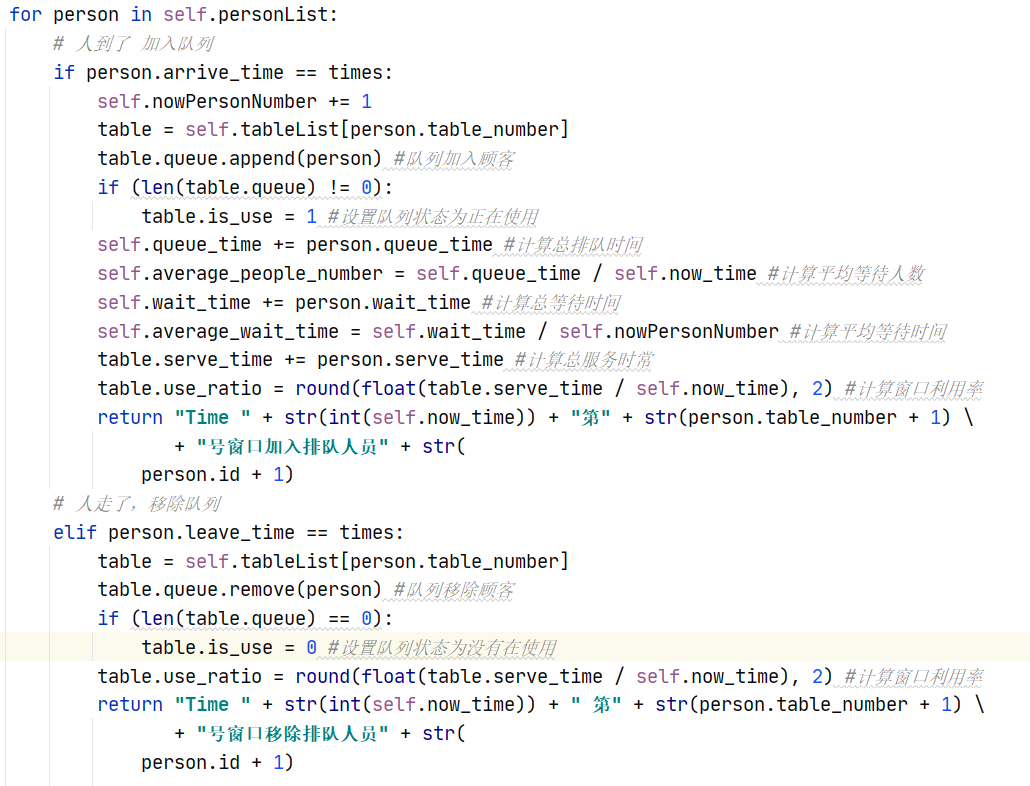
1. # 生成服从指数分布的服务时间
2. **for** i **in** range(0, self.custom\_num):
3. ran = -self.average\_serve\_time \* math.log(random.uniform(0, 1))
4. self.serve\_time.append(ran)
   * 1. 计算离开时间  
        此处流程图与顾客行为流程图一致。

先找到结束时间最早的窗口，再根据当前窗口排队情况计算出顾客离开时间，更新当前窗口结束时间。  
代码如下：



* + 1. 仿真

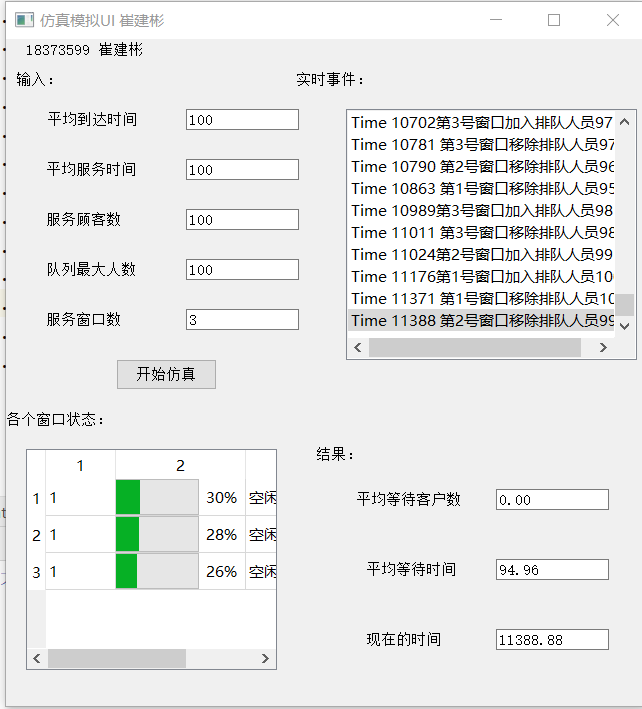
为了和前端代码配合，这里一步一步进行仿真，即运行一次只进行到下一个事件，而不是一致执行。且为了更好的UI效果，这里加了sleep函数来增加延迟。



* 1. **前端UI代码：**

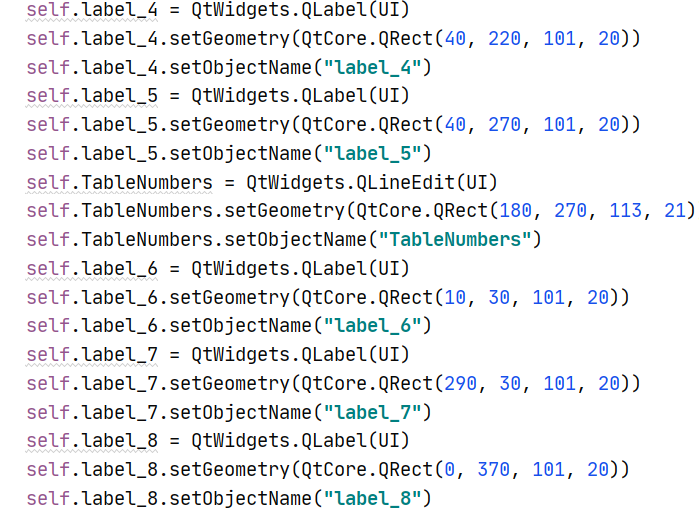
整体采用了PYQT来设计前端，参考了老师课堂上展示的优秀作业的UI设计，加入了动态显示效果。整体UI如下：（具体参考附件中的动图）





部分代码截图如下:

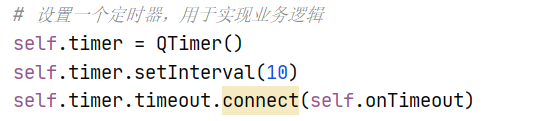
UI.py:



UiClass.py中 刷新业面的代码如下:

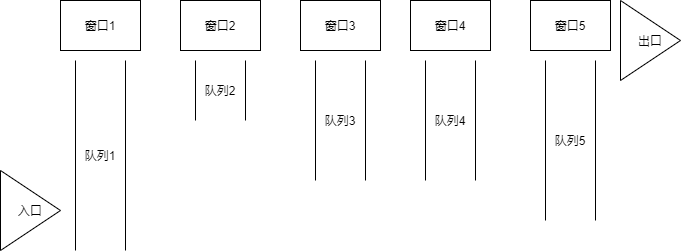


刷新利用了PYQT中Timer函数,从而做到单独开一个进程来刷新程序界面。代码截图如下：



1. **附加题:二食堂的窗口优化策略：**
   1. **该场景的拓扑结构**

学二食堂有非常多的打饭窗口，每个窗口可以供学生排队打饭，每个窗口只能排一个队伍，比较符合M/M/N模型。

但是这个与M/M/N模型也有差距。因为食堂每个窗口提供的饭菜种类不同，每个学生爱好不同，导致他们不会严格按照M/M/N模型中估计得那样，选取当前队列时间最短的来排队。图如下:

* 1. **人流分析:**

学二食堂位于教学区与宿舍区之间，是地理位置相对较好的食堂。常常人流量密集。

因为是大学食堂，所以人流数量避免不了会有高峰期和低谷期。课程安排为上午第四节课下课时间：11:25, 第五节课下课为: 12:15，考虑到同学们平均用餐时间为30min，而且上午有第五节课的同学数量相对较少，所以可以预测中午有两波高峰期。第一波高峰期为周一到周五的11:25-11:55，第二波高峰期为周一到周五的12:15-12:45。

而在下午，下午第四节课的下课时间为17:25，第五节课下课时间为18:15，所以同样的，有两拨高峰期。第一波高峰期为周一到周五的17:25-17:55，第二波高峰期为周一到周五的18:15-18:45。

到了周末，同学们用餐的选择更广，但是整体上用餐时间仍集中在上午11:30-12:30，下午17:30-18:30之间，所以周六周日将有两波高峰期。

* 1. **模拟仿真**
     1. 用M/M/N程序对食堂高峰期进行模拟仿真有如下结果：

假设：食堂有5个窗口，每个窗口只能进行单队列排队，任意一个高峰期人数大约为350人，平均服务时间为1min（打饭的时间），平均到达时间为0.1min，有如下结果：

利用率为98% 平均等待客户数为80 平均等待时间为 20min

利用率为99% 平均等待客户数为73 平均等待时间为 15min

利用率为99% 平均等待客户数为77 平均等待时间为 16min

由此可见，整体排队人数较多，等待时间较长，也比较符合高峰期的情况。

* + 1. 当我们增加到10个窗口时：有如下结果：

利用率97%，平均等待客户数为5，平均等待时间为1.3min

利用率96%，平均等待客户数为13，平均等待时间为1.6min

利用率98%，平均等待客户数为7，平均等待时间为1.4min

**3．**当我们缩短服务时间到30s时，有如下结果

利用率89%，平均等待客户数为9，平均等待时间为1.5min

利用率89%，平均等待客户数为8，平均等待时间为2.1min

利用率90%，平均等待客户数为8，平均等待时间为1.7min

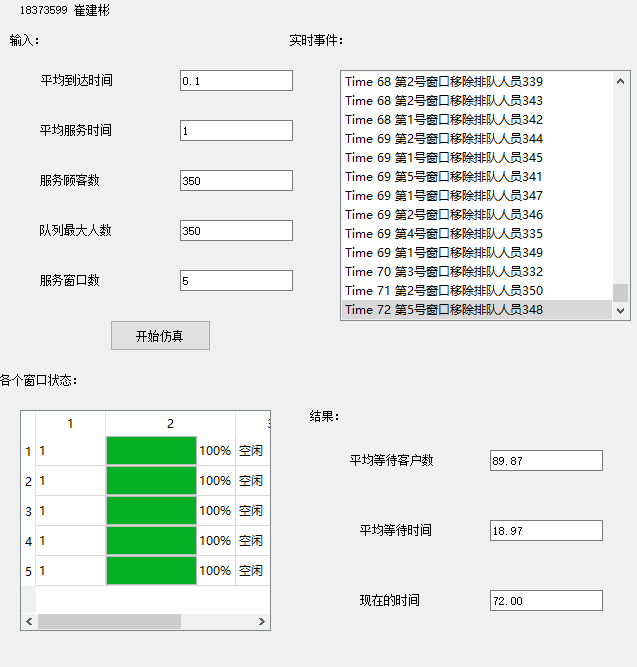
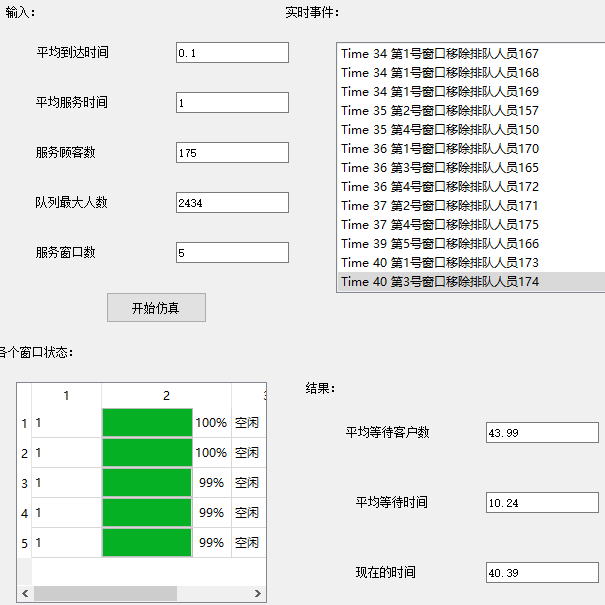
**4**. 当我们减少人流量到175时，有如下结果：

利用率90%，平均等待客人数45，平均等待时间10.2min

利用率88%，平均等待客人数44，平均等待时间11.3min

利用率91%，平均等待客人数48, 平均等待时间9.9min

部分实验截图：



* 1. **更改建议：**
     1. 由上述模拟仿真可知，提高高峰期窗口数量可以大幅度缩减平均等待时间和平均等待客户数，对与缓解高分期压力有较大的帮助。
     2. 由上述仿真结果可知，提高服务速度也就是食堂打菜人员打菜速度，同样可以较大幅度的缩减平均等待时间和平均等待人数。可以较大的程度的环节高峰期压力。
     3. 由上述仿真结果可知，多设置几个下课时间点，或者多开食堂，或者避开人数高分期，即减少人流量少的情况，都可以有效的缓解排队的情况，使排队的效率成指数的提高。但是第三点效果不如前两点显著，只能是线性的减少。
  2. **综上：**

由上述仿真模拟结果可知，更建议食堂采用培训员工，来增加打饭/做饭速度和增加窗口的方法来减少高分期的堵塞。因为这两点将分别增加排队窗口数量和减少服务时间，将极大的减少排队造成的堵塞。

如果条件允许的情况，也可以建议学生避开高峰期就餐，课程安排上，设置更多的课堂分布，来避开就餐高峰期。