Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт

по лабораторной работе № 6

Название: Моделирование парка аттракционов

Дисциплина: Моделирование

Студент ИУ7-75Б _____ Д.В. Сусликов

(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ И.В. Рудаков

(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Содержание

Задание

Реализовать программу для моделирования следующей системы: люди подходят к кассам парка аттракционов с заданным интервалом времени - 1-5. У каждой кассы формируется своя очередь. Посетитель выбирает очередь с минимальной длинной. Кассиры обслуживают клиентов за заданный интервал времени. После того, как посетитель купил билет, он идёт к аттракционам. У каждого аттракциона формируется своя очередь. Посетитель выбирает аттракцион с очередью наименьшей длины. У аттракционов посетителей обслуживают за фиксированный интервал времени. Также посетитель может испугаться аттракциона с вероятностью 25% и вернуть купленный билет. Количество посетителей задается.

Производительность кассиров:

- Кассир №1 - 2-5;
- Кассир №2 - 4-8;
- Кассир №1 - 10-15.

Производительность аттракционов:

- Аттракционов №1 - 5-9;
- Аттракционов №2 - 10-25.

Ниже на Рисунке 1 структурная схема парка аттракционов.

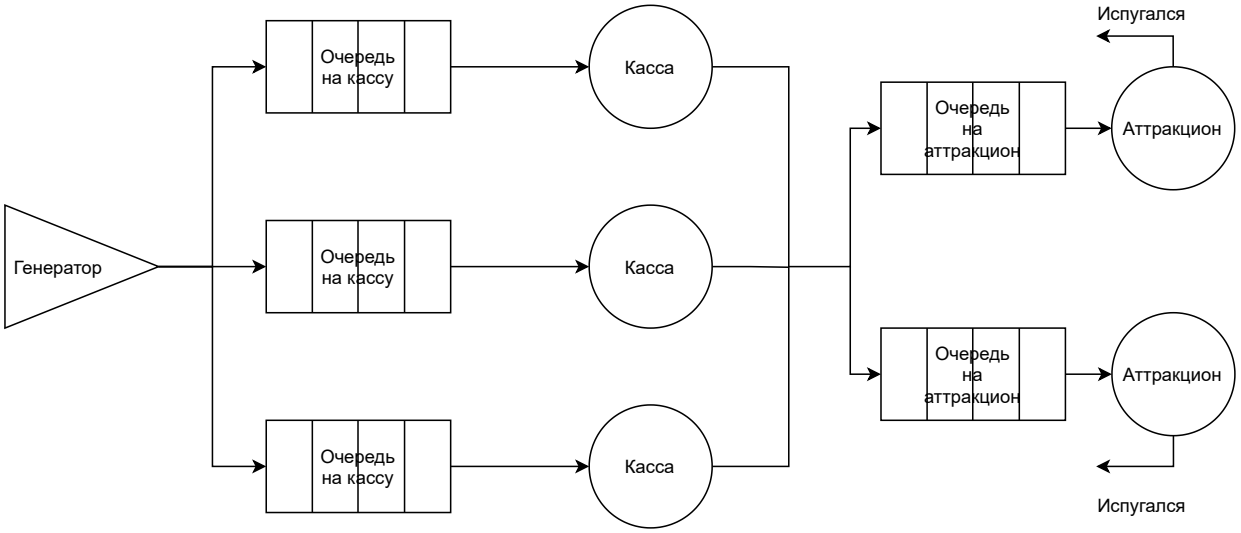

Рисунок 1 – Структурная схема

Аналитическая часть

В процессе взаимодействия посетителей с аттракционами возможно:

1) Режим нормального обслуживания, т.е. посетитель посещает понравившийся ему аттракцион.

2) Режим отказа от обслуживания, когда посетитель испугался аттракциона и решил вернуть билет.

Формула расчёта вероятности отказа:

$$P_{\text{отк}} = \frac{C_{\text{отк}}}{C_{\text{отк}} + C_{\text{обсл}}}$$

Примеры

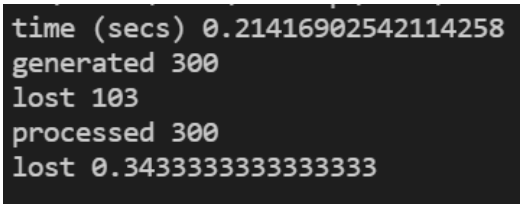

Рисунок 2 – 300 заявок

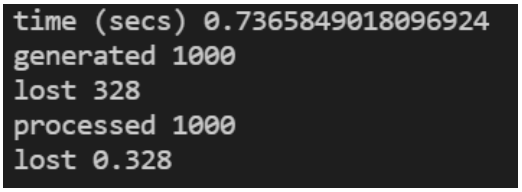

Рисунок 3 – 1000 заявок

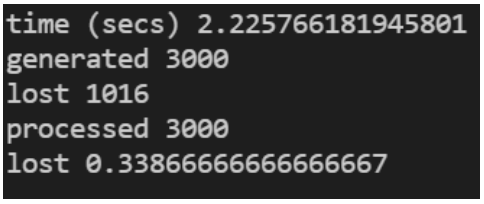

Рисунок 4 – 3000 заявок

Вывод

Основываясь на примерах работы системы при 300, 1000 и 3000 посетителях, можно сделать вывод, что процент потерь примерно равен 33%. Это значит, что следует сделать аттракционы менее страшными, чтобы избежать потерь.

Листинги

Листинг 1 – main.py

```
from time import time
from UniformDistribution import UniformDistribution
from Generator import Generator
from Operator import Operator
from Attraction import Attraction

timeStep = 0.01



def DoStep(generator, operators, Attractions, visitorInfo,
    newGenerate=True):
  if newGenerate:
    res = generator.UpdateTime(timeStep)
    if res:
      visitorInfo['generated'] += 1

  for curOperator in operators:
    curOperator.UpdateTime(timeStep)

  for curAttraction in Attractions:
    res = curAttraction.UpdateTime(timeStep)
    if res == 0:
      visitorInfo['processed'] += 1
    elif res == -1:
      visitorInfo['lost'] += 1

def modeling(generator, operators, Attractions,
    incomingVisitorAmount):
  visitorInfo = {'generated': 0, 'lost': 0, 'processed': 0}

  while visitorInfo['generated'] < incomingVisitorAmount:
```

```
30          DoStep(generator, operators, Attractions, visitorInfo)
31
32      while visitorInfo['processed'] < incomingVisitorAmount:
33          DoStep(generator, operators, Attractions, visitorInfo, False
                )
34
35      return visitorInfo
36
37
38  def main():
39      firstQueueGroup = [[], [], []]
40      secondQueueGroup = [[], []]
41
42      clientGenerator = Generator(UniformDistribution(1, 5),
            firstQueueGroup)
43
44      operators = [
45          Operator(firstQueueGroup[0], secondQueueGroup,
                UniformDistribution(2, 5)),
46          Operator(firstQueueGroup[1], secondQueueGroup,
                UniformDistribution(4, 8)),
47          Operator(firstQueueGroup[2], secondQueueGroup,
                UniformDistribution(10, 15))
48      ]
49
50      Attractions = [
51          Attraction(secondQueueGroup[0], UniformDistribution(5, 9)),
52          Attraction(secondQueueGroup[1], UniformDistribution(10, 25))
53      ]
54
55      totalVisitors = 300
56
57      tStart = time()
58      res = modeling(clientGenerator, operators, Attractions,
            totalVisitors)
```

```
59
60        print('time (secs)', time() - tStart)
61        for key in res.keys():
62          print(key, res[key])
63
64        print('lost', res['lost'] / totalVisitors)
65
66
67      if __name__ == '__main__':
68        main()
```

Листинг 2 – Generator.py

```
1   class Generator:
2     def __init__(self, distribution):
3       self.timeDistribution = distribution
4       self.queues = queueGroup
5       self.finishTime = 0
6       self.visitorId = -1
7
8     def UpdateTime(self, dt):
9       self.finishTime -= dt
10
11      if self.finishTime <= 1e-5:
12        self.finishTime = self.timeDistribution.generate()
13        self.visitorId += 1
14        self.AddToQueue()
15        return True
16
17      return False
18
19    def AddToQueue(self):
20      minLen = len(self.queues[0])
21      minQueueIndex = 0
22      for i in range(1, len(self.queues)):
23        if len(self.queues[i]) < minLen:
```

```
            minLen = len(self.queues[i])
            minQueueIndex = i
        self.queues[minQueueIndex].append(self.visitorId)
```

Листинг 3 – Attraction.py

```
class Attraction:
    def __init__(self, visitorsQueue, distribution):
        self.timeDistribution = distribution
        self.busy = False
        self.visitorsQueue = visitorsQueue
        self.finishTime = 0

    def UpdateTime(self, dt):
        self.finishTime -= dt
        if self.busy and self.finishTime <= 1e-5:
            self.busy = False
            return 0

        if not self.busy and len(self.visitorsQueue) != 0:
            if random() < 0.75:
            self.visitorsQueue.pop(0)
            self.finishTime = self.timeDistribution.generate()
            self.busy = True
            return 1
        else:
            return -1

        return 2
```

Листинг 4 – Operator.py

```
class Operator:
    def __init__(self, queue, queueGroup, distribution):
        self.timeDistribution = distribution
        self.busy = False
        self.queue = queue
```

```
6         self.queues = queueGroup
7         self.curVisitor = None
8         self.finishTime = 0
9
10      def acceptVisitor(self):
11        self.busy = True
12        self.curVisitor = self.queue.pop(0)
13        self.finishTime = self.timeDistribution.generate()
14
15      def finishCurVisitor(self):
16        self.AddToQueue()
17        self.busy = False
18        self.curVisitor = None
19
20      def AddToQueue(self):
21      minLen = len(self.queues[0])
22        minQueueIndex = 0
23        for i in range(1, len(self.queues)):
24          if len(self.queues[i]) < minLen:
25            minLen = len(self.queues[i])
26            minQueueIndex = i
27        self.queues[minQueueIndex].append(self.curVisitor)
28
29      def UpdateTime(self, dt):
30        self.finishTime -= dt
31        if not self.busy and len(self.queue) > 0:
32          self.acceptVisitor()
33
34        if self.busy and self.finishTime <= 1e-5:
35          self.finishCurVisitor()
36          return 0
37      return 2
```