ROLLERCOASTER QUEUE

Presented by
Jirapit Sripitak

A Simulation Modeling Project

Let's understand a rollercoaster's process first!

OUTSIDE

THE QUEUE

THE RIDE

When rides open people will decide whether to come in or not

OUTSIDE

THE QUEUE

THE RIDE

People decides to come in
so we record the timestamps and enqueue them to the system

OUTSIDE

THE QUEUE

THE RIDE

Ride will not operate if the conditions for vehicle's capacity is not fulfilled
people enqueued, if does not satisfy vehicle's condition, will have to wait for more riders

OUTSIDE

THE QUEUE

THE RIDE

More people comes in the system, enqueued, satisfies ride's condition to start

OUTSIDE

THE QUEUE

THE RIDE

Rides operates, timeout the ride with the actual ride time, wait for new events

OUTSIDE

THE QUEUE

THE RIDE

The process starts to repeat, new people will be enqueued waiting for the next ride

OUTSIDE

THE QUEUE

THE RIDE

Queue may stack if the actual ride's time is greater than the interarrival rate

OUTSIDE

THE QUEUE

THE RIDE

Once ride finishes, current riders departs from the system, records timestamps

OUTSIDE

THE QUEUE

THE RIDE

Enqueued may enter the newly cleared vehicle

OUTSIDE

THE QUEUE

THE RIDE

More people enqueued the sytem, the whole process repeats till ride closes

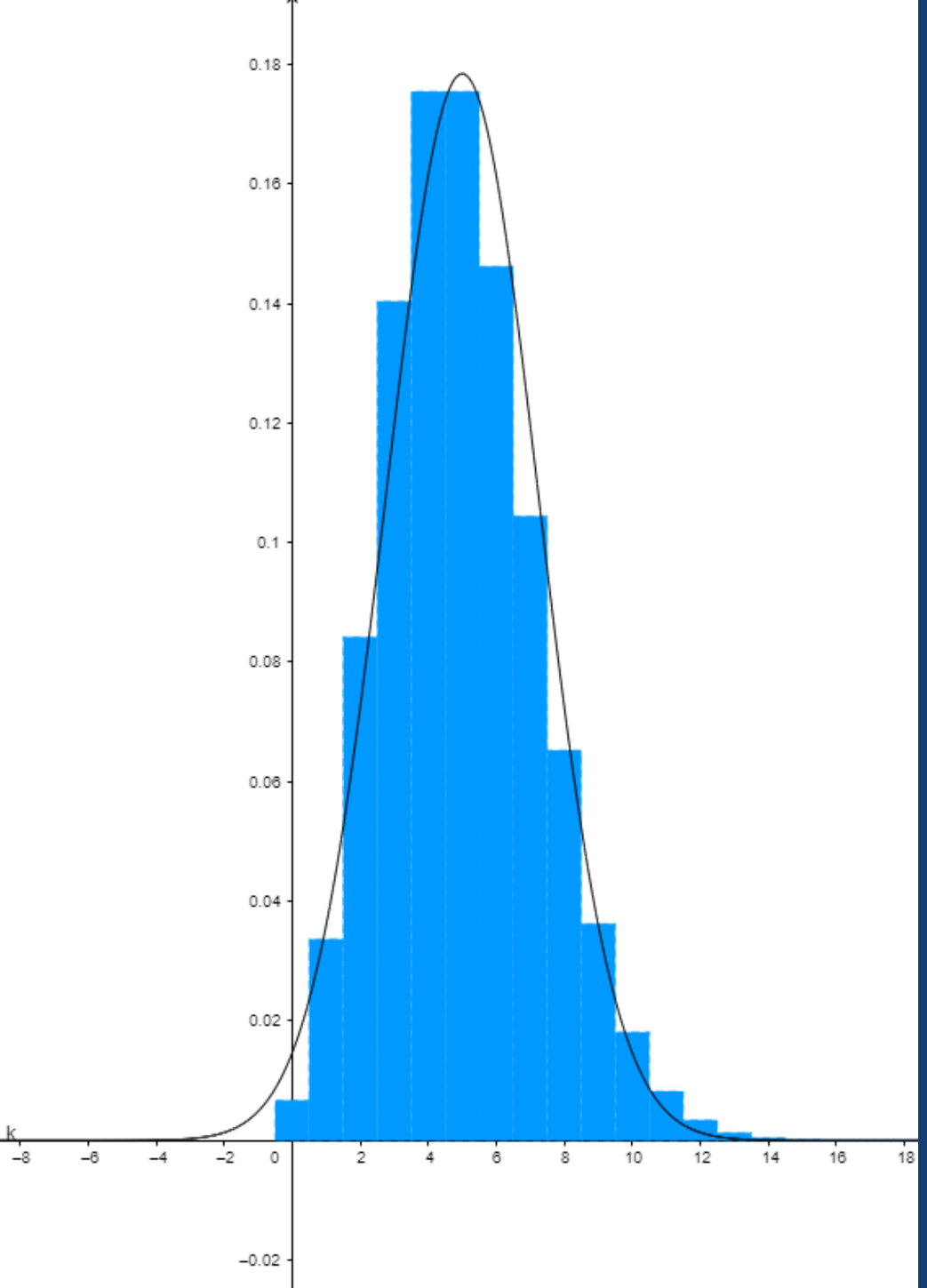
Probability Distribution Tables

Time between arrivals (seconds)	Probability
10	0.35
20	0.25
30	0.15
40	0.15
50	0.075
60	0.025

> Let the number of people that will arrive along the time between arrivals be generated and randomized using **Poisson Distribution** (lambda = 4)

Ride service time (seconds)	Probability
30	0.6
35	0.2
40	0.1
45	0.05
50	0.025
55	0.015
60	0.01

> Note that the **ride service time** excludes the **actual ride time**. It is for the **service time** for when finished riders **exits** and new riders **enters** the vehicle at the station.



For the number of people per arrival time

we will generate it using poison distribution with lambda = 5

OVERVIEW

simpy & numpy

Probability Distribution Tables

Initialize Arrays & Variables

Setup Poisson Generator

Setup Class

Setup Functions for Operations

Placing simpy Events

Creating PDT Generator (numpy)

Placing Generators

Placing Array Manipulators

Performing Simulations

Finalizing Results

```python
"""
Created on 03 November 2022 16:46:56

@author: Jirapit Sripitak
"""

import simpy
import numpy as np

"""
Probability Distribution Tables
"""
IAT = [10,20,30,40,50,60]
IATprob = [0.35,0.25,0.15,0.15,0.075,0.025]
ServiceTime = [30,35,40,45,50,55,60]
ServiceTimeProb = [0.6,0.2,0.1,0.05,0.025,0.015,0.01]

```

```
"""
Global Settings
"""
queued = []
timestampsA = []
timestampsB = []
total_arrival = 0
total_depart = 0
total_system = 0
docking_time = 10

"""
Global Poisson Generator
"""
def generate_poisson():
    return np.random.poisson(lam=5)
```

```
"""
Rollercoaster Informations
"""
class RollerCoaster:
    def __init__(self, env, name, ridetime_seconds, ridecapactiy, vehiclecount):
        self.env = env
        self.name = name
        self.ridetime = int(ridetime_seconds/60)
        self.ridetime_seconds = ridetime_seconds
        self.ridecapacity = ridecapactiy
        self.vehiclecount = vehiclecount

    def __str__(self):
        return f"The ride,'{self.name}', has {self.ridetime} minutes riding time, {self.ridecapacity} seats,\
        and {self.vehiclecount} vehicles."
```

```
"""
Operations
"""

# current_queue = 0

def enqueue(ride: RollerCoaster, env: simpy.Environment, servers: simpy.Resource):
    while True:
        yield env.timeout(np.random.choice(IAT, 1, IATprob))
        people_count = 0
        people_count = generate_poisson()
        print('%d rider(s) arrives at %d' % (people_count,env.now))
        timestampsA.append(env.now[0])
        print("!!! Arrival Time Log: ",timestampsA)
        global total_arrival, total_system, queued
        # current_queue += people_count
        queued.append(people_count)
        print("!!! Queue Log:",queued)
        total_arrival = total_arrival + people_count
        print("- total arrivals:     ",total_arrival)
        total_system = total_system + people_count
        print("- total in the system:",total_system)
        # if current_queue > ride.ridecapacity:
        env.process(rideoperate(ride, env, servers))
            # current_queue -= ride.ridecapacity
```

```
77 def rideoperate(ride: RollerCoaster, env: simpy.Environment, servers: simpy.Resource):
78     with servers.request() as req:
79         yield req
80         global total_depart, total_system
81         out = get_total_seat(ride.ridecapacity)
82         total_depart += out
83         print("+ Ride starts and is released from the station")
84 
85         print("- total depart: ",total_depart)
86         total_system -= out
87         print("- total in the system:",total_system)
88         yield env.timeout(ride.ridetime_seconds + np.random.choice(ServiceTime, 1, ServiceTimeProb))
89         timestampsB.append(env.now[0])
90         print("!!! Departure Time Log: ",timestampsB)
91         print("+ Ride ends and docks at the station at %d" % env.now)
92         print("...Current riders departs from the vehicle...")
```

```python
def get_total_seat(ride_capacity: int):
    global queued
    total_seats = 0
    idx = 0
    for i in range(len(queued)):
        idx += 1
        if total_seats + queued[i] > ride_capacity:
            queued[i] -= ride_capacity - total_seats
            total_seats = ride_capacity
            break
        total_seats += queued[i]
    queued = queued[idx:]
    return total_seats

def get_time_sys(arr, dep):
    for a, d in zip(arr, dep):
        yield d - a
```

```
"""
Simulation
"""
env = simpy.Environment()

ride1 = RollerCoaster(env, "Mark1", 90, 12, 2)

servers = simpy.Resource(env, capacity=ride1.vehiclecount)

env.process(enqueue(ride1, env, servers))

env.run(until=6000)

sys_time = list(get_time_sys(timestampsA, timestampsB))
expected_sys_time = sum(sys_time) / len(sys_time)

print("----------------------------------------")
print(ride1)
print("The expected system time: ", expected_sys_time,"seconds")
print("----------------------------------------")
```

```
PS C:\Users\User\Desktop\Mark\Programming\simulation modeling>
3 rider(s) arrives at 40
!!! Arrival Time Log:  [40]
!!! Queue Log: [3]
- total arrivals:      3
- total in the system: 3
+ Ride starts and is released from the station
- total depart:  3
- total in the system: 0
3 rider(s) arrives at 100
!!! Arrival Time Log:  [40, 100]
!!! Queue Log: [3]
- total arrivals:      6
- total in the system: 3
+ Ride starts and is released from the station
- total depart:  6
- total in the system: 0
6 rider(s) arrives at 150
!!! Arrival Time Log:  [40, 100, 150]
!!! Queue Log: [6]
- total arrivals:      12
- total in the system: 6
!!! Departure Time Log:  [165]
+ Ride ends and docks at the station at 165
...Current riders departs from the vehicle...
+ Ride starts and is released from the station
- total depart:  12
- total in the system: 0
4 rider(s) arrives at 170
```

```
4 rider(s) arrives at 510
!!! Arrival Time Log:  [40, 100, 150, 170, 180, 190, 200, 260, 320, 340, 360, 420, 450, 500, 510]
!!! Queue Log: [6, 4]
- total arrivals:      62
- total in the system: 15
4 rider(s) arrives at 520
!!! Arrival Time Log:  [40, 100, 150, 170, 180, 190, 200, 260, 320, 340, 360, 420, 450, 500, 510, 520]
!!! Queue Log: [6, 4, 4]
- total arrivals:      66
- total in the system: 19
9 rider(s) arrives at 570
!!! Arrival Time Log:  [40, 100, 150, 170, 180, 190, 200, 260, 320, 340, 360, 420, 450, 500, 510, 520, 570]
!!! Queue Log: [6, 4, 4, 9]
- total arrivals:      75
- total in the system: 28
!!! Departure Time Log:  [165, 230, 285, 350, 430, 480, 575]
+ Ride ends and docks at the station at 575
...Current riders departs from the vehicle...
+ Ride starts and is released from the station
- total depart:  59
- total in the system: 16
---------------------------------------
The ride,'Mark1', has 1 minutes riding time,12 seats, and 2 vehicles.
The expected system time:  212.14285714285714 seconds
---------------------------------------
```

Output

Thank you!