

Luna Park Simulation in Anylogic

Alice Schiavone

Project for the course “Simulation”, held by Prof. Alberto Ceselli at Università degli Studi di Milano Statale

## Contents:

1. Introduction
2. Overview
3. Key System Parameters and Key Performance Indicators
4. What-If Analysis
5. Conclusions

## 1. Introduction

“It’s a sunny Sunday morning, and the gates to your Luna Park have opened at 9:00. As more visitors come to go on rides, eat candies and rest on the benches scattered across the park, you begin to wonder: how much money will I make today?”

Our Luna Park has two main sources of income: the entrance to the park is free, but each ride (like the Carousel or the Drop Tower) has a price, but a hungry visitor will also have to pay for snacks. If the visitor is feeling tired they can use a bench, or stop at a bathroom. At the end of the day (from 9:00 to 21:00, 12 hours) we are interested in knowing our profits, with respect to some other factors, like the number of visitors coming in (which could be influenced by advertising) or the number of facilities that satisfy visitors' needs. To simulate the day, we implemented a simple version of the Luna Park in AnyLogic (AnyLogic 8 Personal Learning Edition 8.7.12). The goal is to study the relationship between the Key System Parameters and the Key Performance Indicators: the first indicate the parameters that build and make the system work in a certain way, while the second show the performance of the system, which for us could be the money made that day.

## 2. Overview

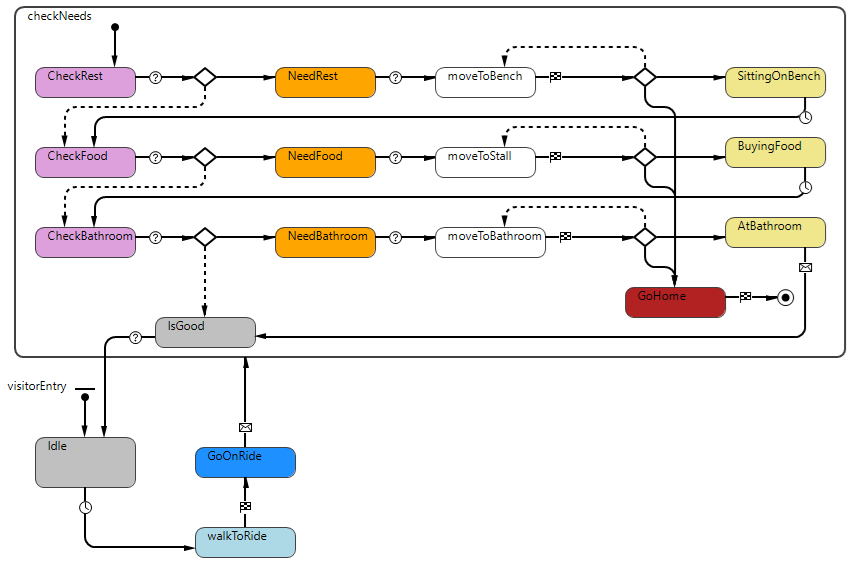
Our LunaPark simulation will stop after 12 hours (or 720 minutes). In the meantime, visitors will enter the park and their main interest will be that of choosing a ride to go on. They can choose from the following, each with their parameters:

| Name | X | Y | Price | Riding Time | Excitement | Intensity | MaxLineLength | Capacity |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Carousel | 200 | 200 | € 2,00 | 10 | 1 | 1 | 30 | 20 |
| RollerCoaster | 300 | 200 | € 8,00 | 2 | 10 | 10 | 100 | 10 |
| Haunted House | 400 | 200 | € 3,00 | 20 | 4 | 6 | 30 | 10 |
| Swing Boat | 200 | 300 | € 5,00 | 10 | 8 | 6 | 40 | 30 |
| Drop Tower | 300 | 300 | € 5,00 | 5 | 9 | 9 | 40 | 10 |
| Log Flume | 400 | 300 | € 6,00 | 15 | 7 | 2 | 40 | 20 |
| River Rapids | 200 | 400 | € 4,00 | 15 | 6 | 4 | 40 | 20 |
| Bumper Cars | 300 | 400 | € 3,00 | 10 | 5 | 2 | 30 | 20 |
| Ferris Wheel | 400 | 400 | € 2,00 | 20 | 3 | 3 | 70 | 50 |

The parameters describe:

| X, Y | Location on the main frame. |
| --- | --- |
| Price | Price of the ride for one visitor. |
| Riding Time | Time duration for riding the attraction. |
| Excitement | How exciting it is to go on this ride. |
| Intensity | How intense it is to go on this ride. |
| MaxLineLength | Maximum queue length for the ride. |
| Capacity | Maximum number of people allowed together on the ride. |

After going on a ride, each visitor will check their needs. If some need has to be satisfied, the visitor will look for a Bench (if tired), a Stall (if hungry) or a Bathroom. If any of these needs cannot be satisfied, the visitor will go home (with their unspent money!). We want our visitors to stay as long as possible doing rides or buying food. The visitor logic is described by its flowchart:



To avoid cluttering this report, details about the logic of the Visitor agents, like parameters and thresholds, can be inspected by looking at the Anylogic model code. It is worth mentioning that the patience and the money parameters are decided at agent creation, following (respectively) a truncated normal and an exponential distribution.

| Money | exponential(0.05, 10) | [lambda, min] |
| --- | --- | --- |
| Patience | (int) normal(30, 120, 50, 30) | [min, max, shift = mean, stretch = std deviation] |

The facility or ride are selected uniformly at random.

The visitor needs to have their needs above a certain threshold (20) for that need to be satisfied. Needs are:

* Energy
* Hunger
* Bathroom Need

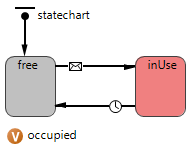
When any of these are less than 20, the visitor will try to raise its need points. If they can’t, they will leave the park. At agent generation each need has points equal to 100. The function lowerneeds decreases point based on elapsed time.

The logic of the Visitor relies on 4 other agents, namely:

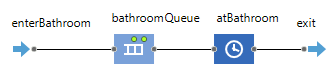
* Rides, as previously described. When selecting a ride, a visitor will firstly check if the line is full, if not they will join the queue. Visitors in line will go on the ride if the capacity of the ride is not reached. The ride will start only when the capacity limit is met, or otherwise after some time is passed. The events WaitForResume and ResumeIfFull handle this process. The visitors are put on hold if the capacity is met, and the delay (riding time) is started by the two events. If a visitor stays in line for too long (exceeding the patience parameter), it will exit the ride queue.

## 

* Benches, which have a capacity of one visitor, and will light up in red when in use. Visitors will sit on a bench for 10 minutes and restore 50 energy points.



* Stall, which are generated based on a table with attributes “Name” and “Price”. The hunger value restored by each stall is based on the price, so a pricier food will restore more hunger points. We assume that a visitor is served as soon as he orders food at a stall.
* Bathroom, which follows a simple queue-delay logic, with a capacity of 10 bathroom stalls per bathroom.



If the visitor that is trying to satisfy their needs and meets a full line, or doesn’t have enough money, will try another facility by lowering its needs further. If the need points reach zero or lower, the visitor will go home.

The visitor Excitement and Intensity are not actually useful to the analysis or to the simulation, but they still give insights about the level of satisfaction of a visitor, and could be considered secondary KPIs.

## 

## 3. Key System Parameters and Key Performance Indicators

We have many parameters for this system, but we will call KSPs (Key System Parameters) only those parameters that we can imagine to be actually modified in a real life scenario. For example, the threshold for which a visitor will look for a bench is not a KSP, because us as Luna Park owners cannot have an impact on that. Instead, we will focus particularly on the following:

| Number of Benches | How many branches we will have at start-up. |
| --- | --- |
| Number of Bathrooms | How many bathrooms we will have at start-up. |
| Number of Visitors  per Minute | Based on some advertising strategy, we will have a certain number of visitors entering the park per minute. |

Other parameters like the price of the rides or the food influence the system, but we will treat them as immutable and agnostic to the system. We will not actually define a price for our advertising strategies, because that is behind the scope of this project. Instead, we will ask ourselves: “If I can reach a certain number of visitors per minute, how much profit can I make?”.

To evaluate the performance of the model, we need to identify on which specific index we want to focus our attention. For us, the KPIs (Key Performance Indicators), will be:

| Total Profit | aa |
| --- | --- |
| Profit per visitor | aa |
| Unspent money  (Total, mean per visitor) | aa |
| Impatience Visitors | Number of visitors that will leave the ride queue because they have waited too long. |

## 4. What-If Analysis

Alaska

## 5. Conclusions

Alaska