Balelec 2018

Simulation and optimization project

Victor Delafontaine
Guillaume Gavillet

Teacher: Yoo Min-Jung

Simulation and Optimization of Industrial Applications
Spring 2018

June 1, 2018

Contents

1 Introduction

1.1 Project

Since 1981 the beginning of the month of May is animated by the Balelec festival. This is one of the biggest music event organized by student in Europe. It consist of about 30 live concerts spread on five different stages, a budget of half a million and roughly 300 volunteers. On a Friday night each year, about 15'000 people invade the EPFL campus to enjoy live music, party and drinks.

Figure 1: View on the Balelec festival

How to handle such a big crowd? Where to put the emergency exits in case of an incident? What would happen if the headliner on the main stage is canceled or a small concert attract a huge amount of people?
These questions are major factors that can influence the thinking on the mapping of the concert.

In this work, we used the software `AnyLogic` in order to simulate the people flow inside the Festival. We started by identifying the festival goers needs and actions and used the Balelec map and schedule of 2018 in order to run our simulation. This project interest is the people flow inside the festival and to try to answers the above questions.

1.2 Simulation objectives

The model would consist of the simulation of the people flow in the festival during its whole duration. In order to design the model, we used those following guidelines.

We want our model to be representative of the reality. The movement of people in the festival is really random in reality. Some people come with friends, some comes alone. Some people would go to attend one concert and then just leave. For some other the festival would be

an occasion to meet people and enjoy some music. Hence it is really hard to obtain a realistic scenario as this last may itself vary a lot and depends on a lot of different factors. Hence we are going to implement a model that allow us to simulate extreme scenarios in realistic ways, as seen in the introduction, like an emergency exit.
Our goal is to compare the simulation with a scenario with the simulation without the scenario. As such, we won't need to directly compare our project with the reality, as long as is seems appropriate in terms of flow.

Hence for such scenarios, we would start by simplifying the model by modeling the festival goer with a finite number of actions that would imply movement. We also will not take into consideration the interaction between festival goers and hence the case of people moving or waiting in groups. This would have been possible in `AnyLogic` with the *PedGroup* blocks, but we estimated that it wouldn't add much value to our simulation.
From our point of view, these simplification would allow us to implement realistic scenarios.

1.3 Optimization objectives

The output we are trying to get from this model are as follows. Firstly, we want to see the general flow of people inside the festival. This will not be a perfect simulation, but we want to reach a stage where it fits more or less precisely the reality.
Once we have this base simulation, we want to add scenarios and see their influence on the flow. The scenarios are chosen as things that could happen during the festival. We decided on three possible scenarios.

1. *emergency*: that can be a fire, or any other thing that would require the total evacuation of the festival. While this is not desirable, the organizer needs to be prepared if the issue were to come. The output we can obtain from this is the time until complete evacuation, that we could optimize by changing different parameters (number and size of the exits...).

2. *underdog*: a concert on a small stage that attracts much more people than estimated. This could be the result of a very good concert, or an outburst of praise on social medias for this particular concert. This would result to the saturation of a small stage, as well as the desertion from other areas.

3. *overhyped*: a concert on the main stage that is canceled, or much worst than planned. Reasons for that could be technical problems, or uncertainties from the band side. The result would be similar to the "underdog" scenario in reverse. The big stage will be emptier than without the scenario, which would result in more people in the rest of the festival.

The *emergency* scenario can be triggered at any time, and the worst would be around midnight, when the festival is at its maximum. The two others will be linked to a specific concert for this simulation. Namely, the *underdog* scenario is paired with the *Venus on Fyre* concert on the Squatt stage. and the *overhyped* is linked with the *Orchestre tout puissant Marcel Duchamp XXL* concert.

2 Modelisation

To create this model, we used the pedestrian library of `AnyLogic`. Indeed, as the main point of our simulation is circulation and flows, it was essential. We used the University version and its 60 days trial to override the one hour limitation of the PLE version.
With this, we are able to simulate the entirety of the festival inside one simulation.

2.1 Starting point

The input of our model are as follows:

- the number of people entering the concert, obtained from press articles: 14'000 people for the 2018 edition
- the concerts schedule, obtained from the Balelec Facebook page (see figure 2 below)
- the map, with the location of the food/drink stands, stages, and other interesting points, also from the Facebook page (see figure 3 below)
- an approximation of arrival times, obtained from discussion with the entrance staff and by personal observations
- the personal experience gained from previous editions, as well as a small survey with some festival goers

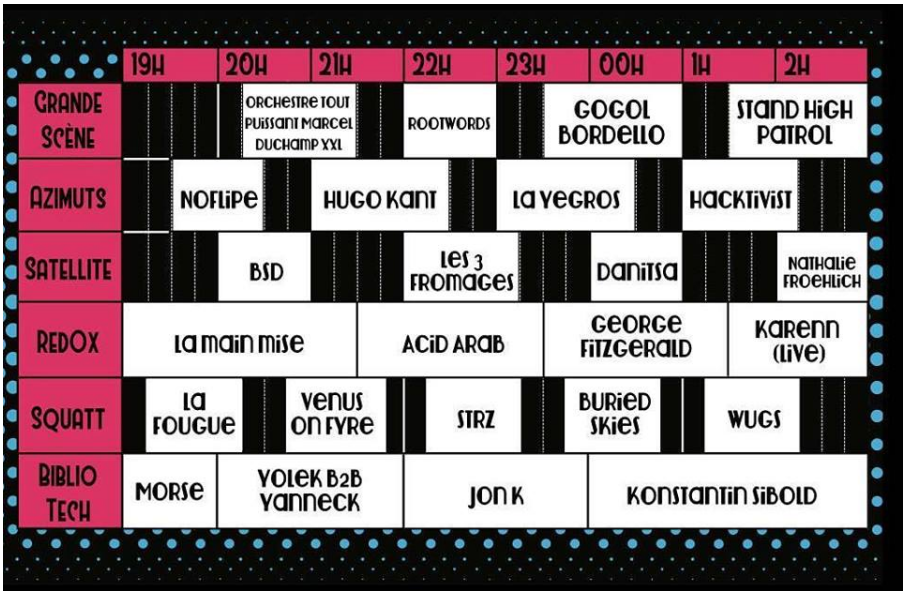


Figure 2: Schedule of the festival

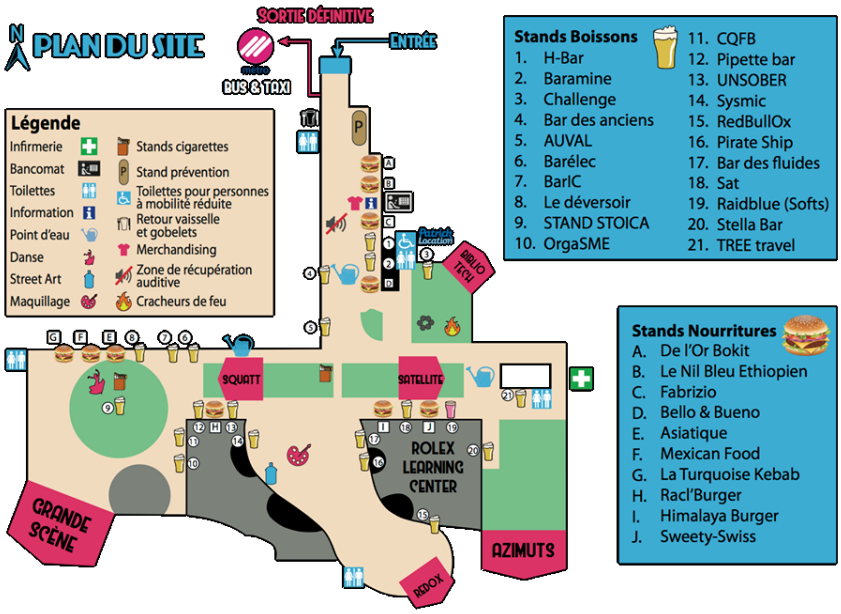


Figure 3: Map of the festival

We inputted the estimation of the arrival times inside an Excel sheet. The important point is that the peak entries are between 21:30 and 22:30. In addition to that, a very small percentage of the total entries are before 20:30, and after midnight. We divided the evening (from 19:00 to 03:00) by 30 minutes slots. The plot obtained follows an approximate Gaussian, and is shown in figure 4 below.

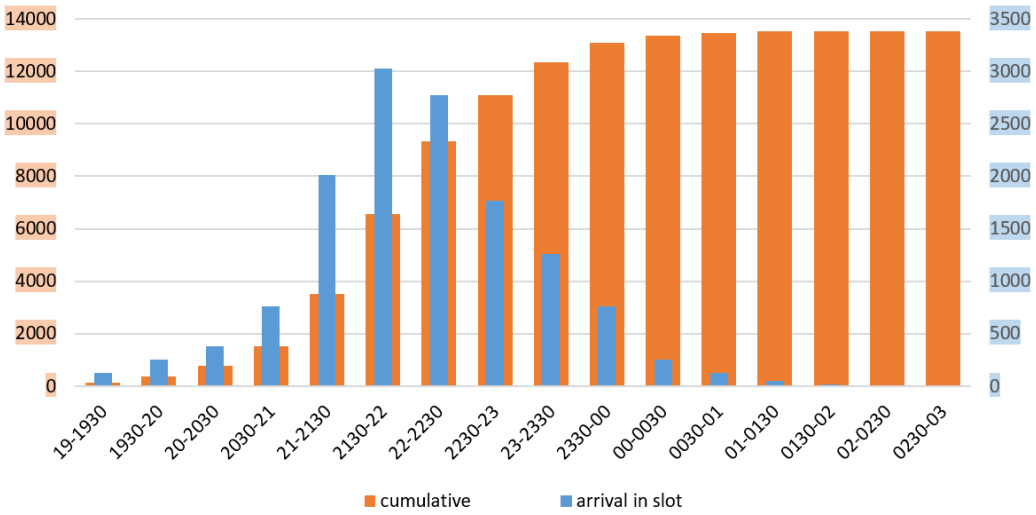


Figure 4: Arrivals estimation

Our personal experience gave us a very important information. The festival goers can be divided in three main categories: people who mainly come for the concerts, people wanting to enjoy the food/drink stands while going at some concerts, and people in between doing a bit

of everything. We will use these categories in our simulation to separate the general drive of the festival goers.
The amount of drinks and food consumed during the evening depends on this category, as well as the time spend in concerts. We will use that categorization to decide on various parameters explained in the following sections.

2.2 Festival map

The 2018 map of the festival was used in order to model our simulation. We separated the map in different zones according to the Balelec layout. We model three different types of zones:

- stages: where the different concerts take place
- food/drink: where the stands are
- random: the majority of the area, include some parts of the stages, plus all the remaining free area

We choose to simplify the stands where they sell drink or food. We decide to model that as three different spots where the sell both food and drink for the sake of simplicity. The final simulation layout can be observed on figure 5, where each zone is labeled for a better comprehension.

We define an area, that is the biggest on the model that represent the part where people randomly move around or stay when they are not taking other actions as it will be explained further.

This manipulation would allow us to create the displacement of the people regarding their action. That would imply that for example a person is watching a concert on the main stage, then he/she get thirsty and decide to go for a drink. Then in our model that person would just go to the bar closest to this stage.

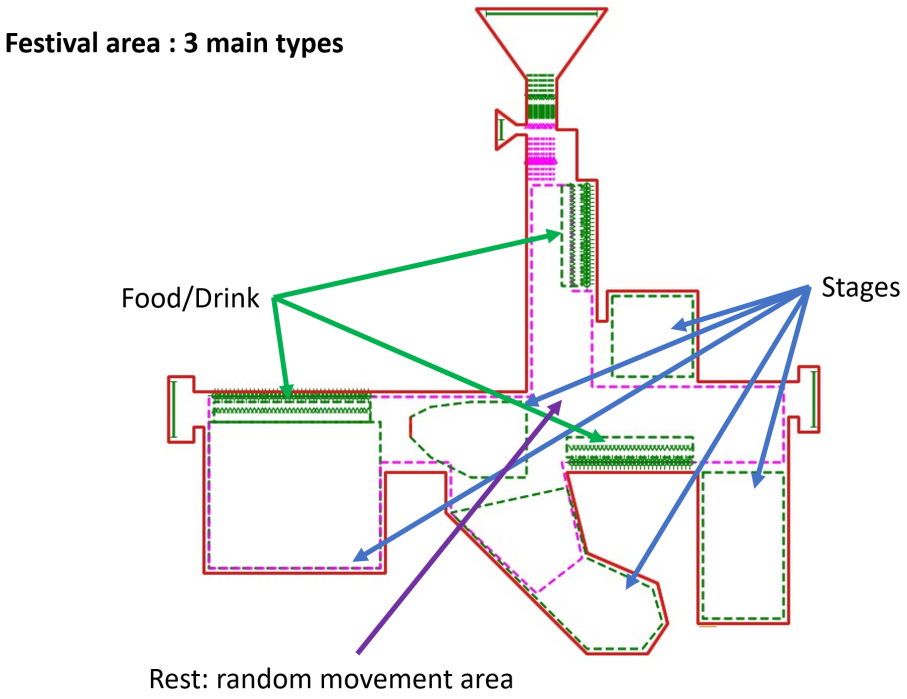


Figure 5: 2D map of the festival with the different zones, as seen from above

2.3 Pedestrian model

The biggest part of our work was to create and model the displacement and actions of the festival goers. To accomplish this task, we created an agent type that we called `Person`. Its role would be to take decisions of what action to do at which time, as well as store the parameters relative to each particular agent.

This agent contains a set of parameters. The major ones are as follows:

- a first set of probabilities for the actions (eat, drink, concert, random and exit)
- a second set for the concerts (grande, azimuts, redox, squatt and biblio)
- the times spend in the concerts

Note that we decided to model only five of the six stages of the festival. This is due to the fact that the *PedSelectOutput* block only has five outputs. Adding a second block after the first one would have been possible, but would have changed our probabilities model. Hence we decided to remove one of the stages from our simulation.

The agent also contains a set of parameters dictating some actions. For example, it contains the variables *cancel_drink* and *cancel_food*, that dictates the maximum number of people in front of them in a queue before they decide to do something else. This information was collected via a small survey.

The two variables *number_food* and *number_drink* are also present here. These two variables limit the number of drinks and food items taken in the evening, depending on the agent category. For example, a *concerter* agent will have a drink limit of between 2 and 5 (randomly decided), while a *eater* will be limited between 3 and 8.

It also contains a set of function that are called periodically by an event repeated every minute:

- *update_proba()*: increases the action probability set according to the agent category, his time already spend in Balelec, the current time, and what he already did. For example, if the time is 2am, his probability to exit will be high. If the "Grande scene" concert just started, its probability to go to the concert will be high. See listing 7 for more details.
- *update_concert_proba()*: changes the probability to go to a concert or another depending on the concert state (on/off). It also takes into consideration the optimization scenarios that will be explained later
- *update_concert_delay()*: changes the time spend in the concerts, depending on the agent category and on the scenarios

```
1 p_hungry += 10/(60*100);
2 p_thirsty += 15/(60*100);
3 p_concert += 10/(60*100);
4 p_exit += 0.8/(60*100);
5 p_random = 1-(p_hungry+p_thirsty+p_concert+p_exit);
```

Listing 1: Extract of the *update_proba* function

In listing 7 above, we can see the probability update for a eater type. For example, the probability to go get something to eat (*p_hungry*) will be increased of 10% in one hour.

A last function can be called from the main. It is the *reset_proba* function, called when the agent enters a block. For example, it he goes to a drink stand, his *p_thirsty* will be reseted. By experience,

In addition to that, a small state chart controls the time spend in the concerts. It controls that the time spend in the concert is the same as dictated by the *update_concert_delay* function. This chart is also used to exit the concerts in the case of an emergency scenario.

In the *main*, the agent evolves inside a flowchart. At its source. it creates the agents based on the schedule of figure 4. On creation, it stores the `Person` agents inside a collection.
The goal of the flowchart is to route the agents based on the probabilities. This is done using the *SelectOutput* block, with the five path being the five actions, or the five concerts.

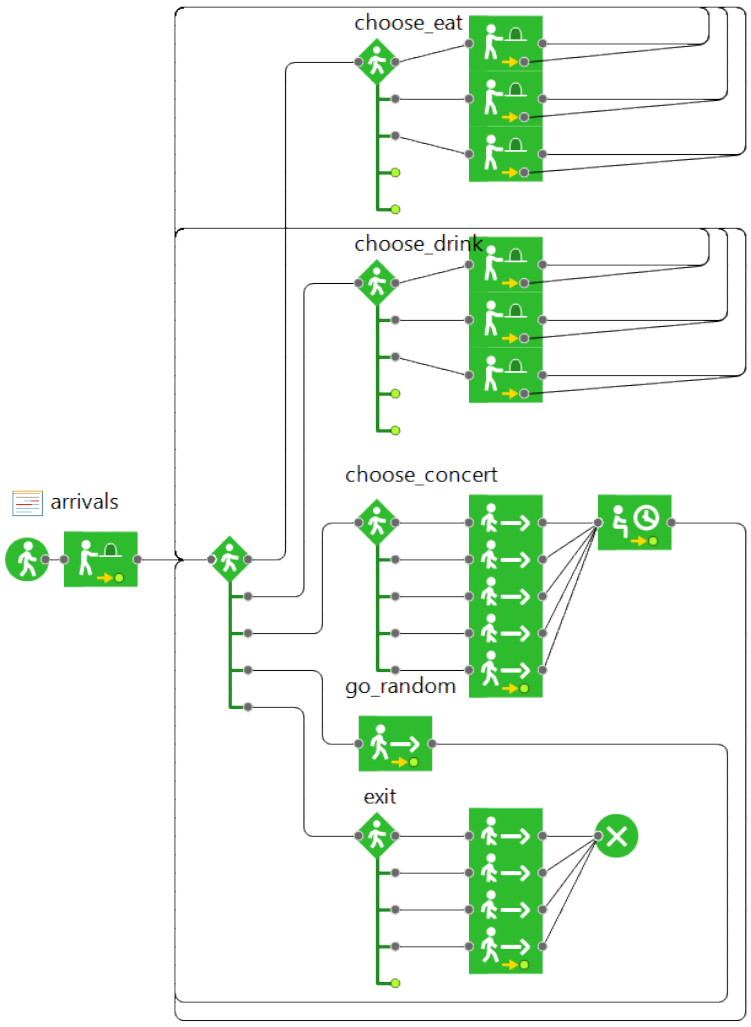


Figure 6: Flowchart for the `Person` agent

To sum up, this agent is created from the main as a `Person` type. It has a default set of probabilities based on a category chosen at random. This set of probabilities is what will make it move between the different possible actions. During its time in the festival, the probabilities will change depending on parameters.

3 Results

The first goal of this simulation is to obtain festival goers' movements and action close to the reality. This was done by a lot of fine tuning to the parameters (probabilities growth, base levels...). In the end, we obtained a model that is close enough to the reality.
Some points are not perfectly represented, for example the group modeling, which we don't consider. However, we estimated that the individual actions are sufficient for this model.

With this base model working, we will then be able to move on to our optimization scenarios.

3.1 Design of Experiment

For our optimization, our input are whether our three scenarios are turned on or off. In addition to that, we could check the effect depending on some parameters. For the overhyped and underdog scenarios, it could be the stage of influence, or the time. For emergency, we decided to change the size of the exit. Due to some technical restrictions, it is not possible to change the position of the exits in the real festival, so we didn't judge it necessary to move the exits.

Factor	**Values**
Emergency	true/false
Underdog	true/false
Overhyped	true/false
Size of exit	5 to 50m
Concert/stage of influence	Following schedule

Table 1: Controllable factors

The purpose of our scenarios is to see the density across the different points of the festival, which will be our main output. We will also check the time until complete evacuation for our emergency scenario.

3.2 Emergency

The *emergency* scenario, when active happens at a fixed time. When it is triggered, all the people inside Balelec immediately evacuate. From a simulation perspective, all the pedestrian in all blocks are canceled.
The pedestrians go to the exit depending on their position according to figure 7. We decided to add two emergency exits in addition to the main exit at the entrance of the festival, as they were during the festival. These two exits are placed at strategic points. Indeed, they go straight onto the main road (Route des Noyerettes). We thought about adding other exits at other points, for example near the RedOx stage (under the Rolex Learning Center), but the configuration of the stages doesn't allow it.
One parameter we could change is the width of the emergency exits, but this is limited by the

Securitas capacity. Indeed, it is not possible to have a huge emergency exit as it would need to be monitored by *Securitas* during the rest of the evening.

The way we modeled the emergency scenario inside our simulation was by adding a checkbox that we could manually trigged. Afterward, we also added an event that could trigger it automatically at a fixed time. Both have the same effect: they set a boolean (*emergency_bool*) to *true*. It also launches a time counter to get the time until complete evacuation, and changes the walking speed of the pedestrians, now running for their lives in an orderly manner. In addition to that, it stops the festival goers to enter the festival.
When this boolean is true, all festival goers immediately evacuate, using an exit based on the figure 7 below. Instead of using the "normal" exit for during the festival, they use the entrance in reverse way. This routing is done using the *getX()* and *getY()* functions. It is not based on the closest exit, that would direct the people on the RedOx stage at the right exit.

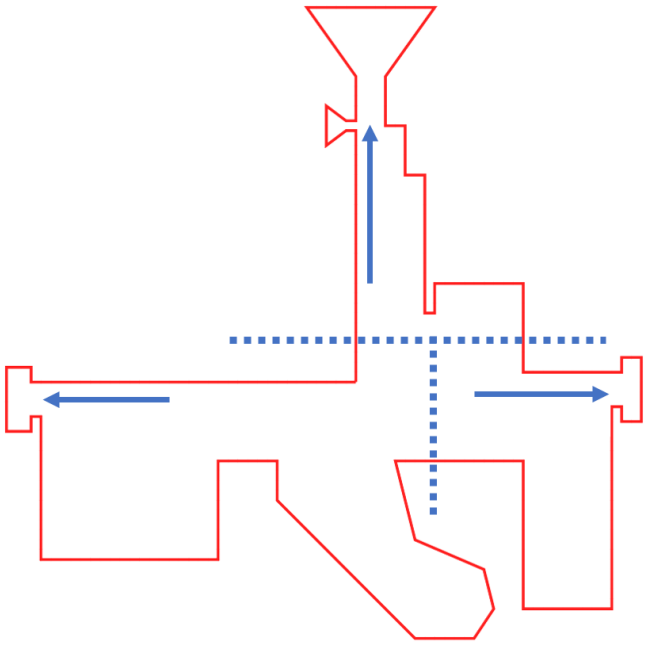
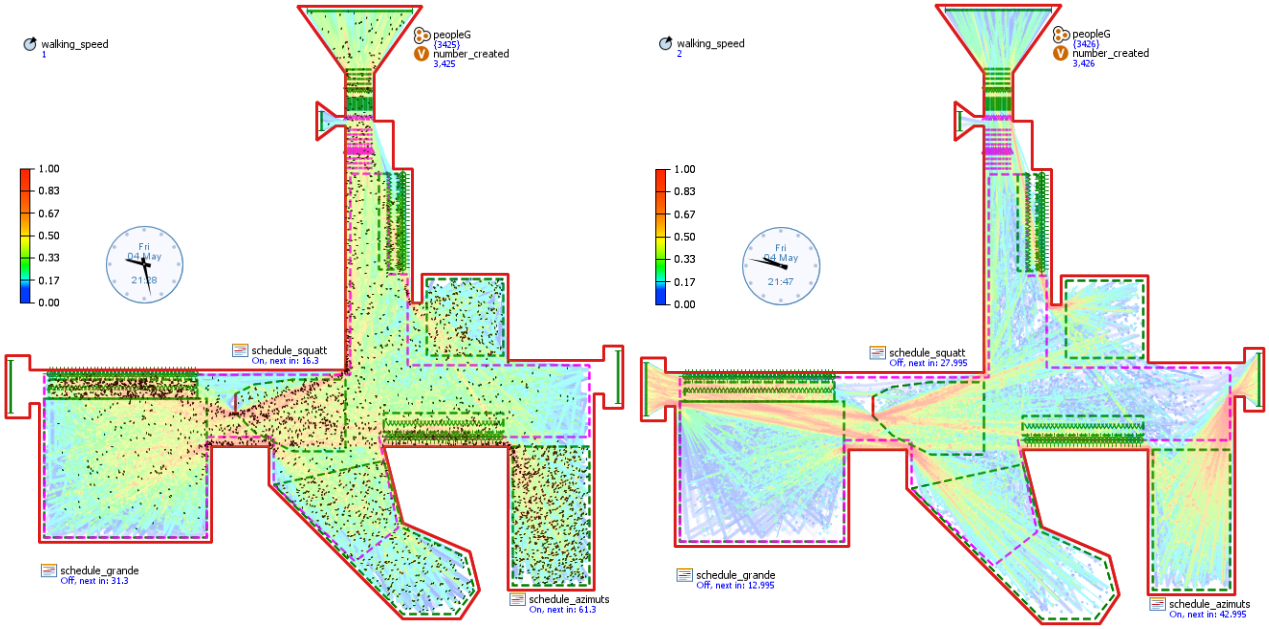


Figure 7: Emergency direction for the people inside the festival

We decided to trigger the scenario at 21:30. This time was chosen because of the problem that will be named in section 4.3. Mainly, it runs very badly when the festival is full. This made it impossible to test our scenario at the moment when it would have been most helpful... This time is still pretty useful, as it is just after the end of a concert on the main stage. At this moment, the pedestrian are moving to different locations. Limitations in the model made it impossible to cancel the movement of a pedestrian, so they always need to go to their destination before evacuating.
In figure 8 below, you can find the result of our scenario. We can first see that the time until complete evacuation with this size of exit is approximately 20 minutes. That's a lot considering that the population inside Balelec is far from its maximum. Indeed, there is "only" approximately 3'500 people inside, a fourth of the total count.



(a) Before the emergency

(b) After the emergency

Figure 8: Result of our emergency scenario

To better understand the cause of this long time, we can analyze the following four figures (9 to 12).
We can see in figure 9 that the evacuation begins efficiently the minute after the emergency is declared (at 21:28). However, we can notice in figures 10 and 11 that the pedestrian of the model go to the exit they are assigned to, but taking the direct path. This is an issue that we will address in 4.3. As a result, they disturb each other, and can't go to the exit efficiently.
Another thing can be seen in figure 12. After ten minutes, most people have already evacuated. Only a few remains in the festival (from the model 58 in this case, but most of those are already almost outside).
On a bigger scale, an almost complete evacuation of 3'500 people in only ten minutes is good. In the reality, chaos may occur, which would probably increase this number a lot, but the human comportment under stress is not what we want to simulate here.

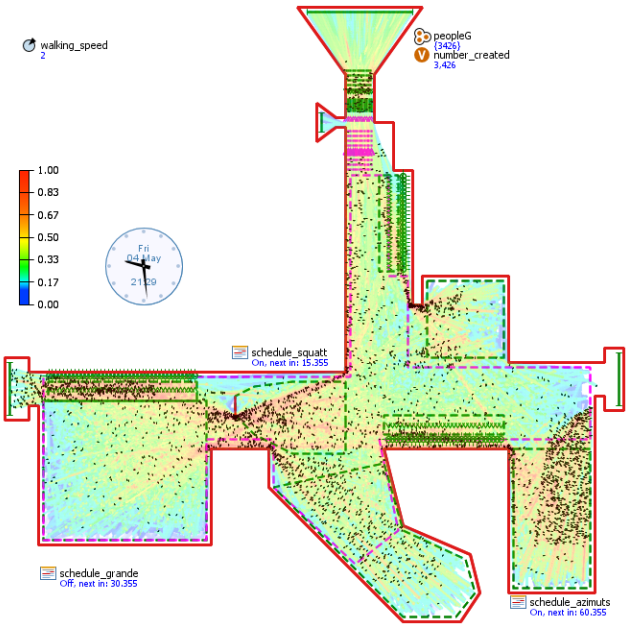


Figure 9: Situation at 21:29

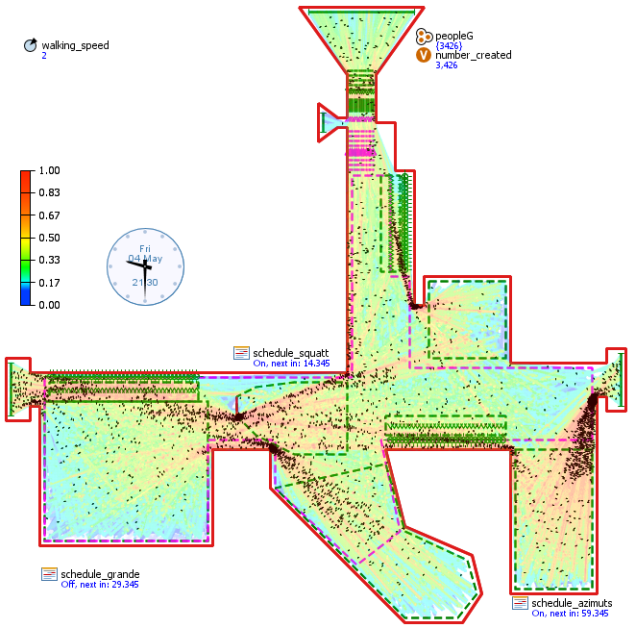


Figure 10: Situation at 21:30

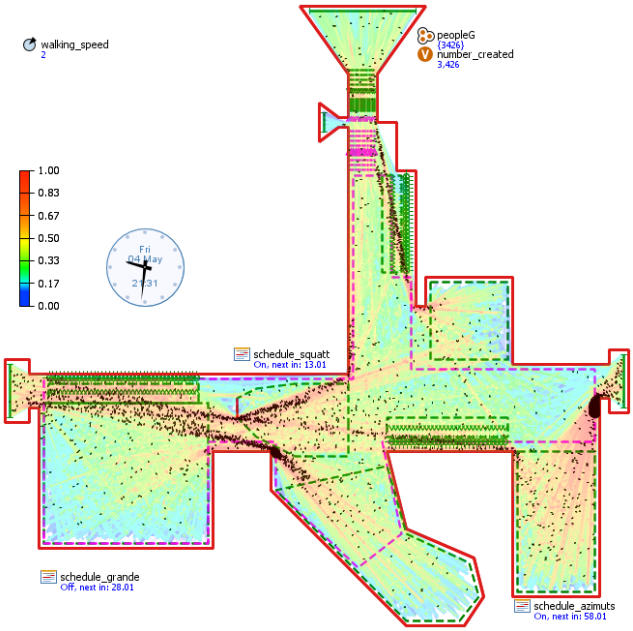


Figure 11: Situation at 21:31

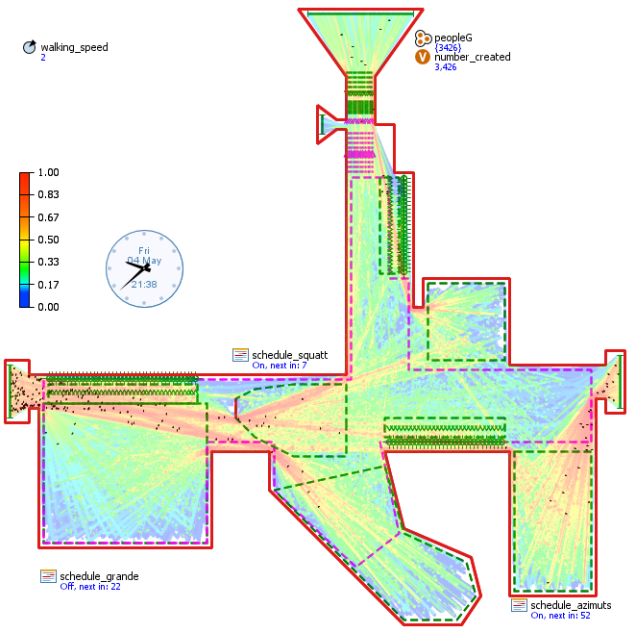


Figure 12: Situation at 21:38

We can now try to change some parameters to see if the time will change a lot. We tried to increase and decrease the size of the exits. As they are now, they are approximately 30 meter large, which is already a lot for the *Securitas* to monitor during the evening.
We tried to change it to 10 and 40 meters. However, it didn't change the result enough. For 10 meters, the time was even smaller (18 minutes), probably due to the location of the pedestrian at the emergency signal. For 40 meters, it was also smaller of three minutes (17 minutes).

The reasons we identified for this is the one we already shown above. In figure 10, for the right-hand emergency exit, we can see that the pedestrian don't take all the exit size, but rather only a corner. As a result, reducing or increasing the size of the exit doesn't make that much of a difference...

One thing we can notice is that in all figures during the evacuation, the main entrance is not crowded as are the two emergency exits. Indeed, our zone division made it so that a minority of the festival goers are in the corresponding zones. This is due to the lack of a major stage in this zone. Something we could do is push a majority of the festival goers to this exit. This would require a small action from the staff to push the pedestrian to this exit in case of emergency. Additionally, the festival goers could be warned to mainly evacuate there in the event. But this is realistically not feasible in an event like that, where most people already drank a bit before coming. In addition, it could be bad for the festival image, as it could show that the festival organizers are "expecting" an emergency.

Further testing with a zone map as shown in figure 13 resulted in more coherent results, closer to what we could expect in a real concert. The resulting density ten minutes after triggering the emergency is shown in figure 14.
It took a bit less time as we triggered it at 21:00 instead of 21:30, but the evacuation resulted in less bottlenecks.

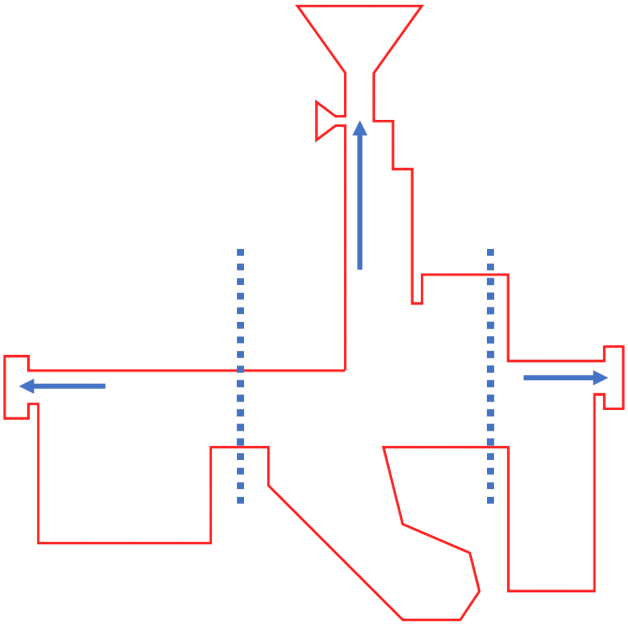


Figure 13: Updated emergency mapping

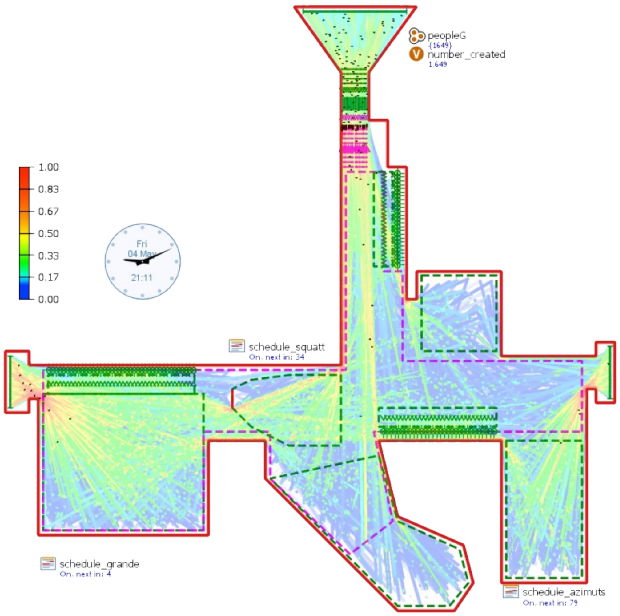


Figure 14: Situation after ten minutes

3.3 Underdog concert

We decided that the *underdog* scenario happens for the concert between 20:45 and 21:45. It is a small concert on a stage placed critically. Indeed, it is between the *grande scene* and the center of the festival. As a result, a lot of people can pass next to it when the *grande scene* is active.

In order to implement this simulation, we introduce an option underdog in our model, which represent is a boolean variable. This variable is initially set as `false` however, between 20.45 and 21.30, we can activate it. When we do so, the probabilities in every agent of type `Person` are modified as following :

- the probability of the person to go to a concert increases faster for the "concerter" type, in addition to the usual modifications done

```
if(underdog)
  p_concert += 10/(60*100);
```

Listing 2: Extract of the *update_proba* function

- In the function where the festival goer choose which stage to go to, the probability to go to the *scène squatt* increases by 25%, for any type

```
if (underdog)
  p_squatt += 25;
```

Listing 3: Extract of the *update_concert_proba* function

Hence this would affect every people regardless of their type.

- Finally, we increase the time that the person would spend at the concert by 15 minutes, regardless of the type of the person.

```
if(underdog)
  t_squatt += 15;
```

Listing 4: Extract of the *update_concert_delay* function

That scenario would lead to a bigger number of people attending the concert. As we can observe on figures 15 and 16 the crowd start to gather in front of the stage. We see that the other stages are a little less crowdy, hence the people left them to come to the underdog concert. In the standard scenario (figures 17 and 18), the most part of the people only pass by this stage to go the main stage or the one under the Rolex, the people are more spread. This simulation, was predictable as we increased both the probability to go to this concert and the time spend there.

This scenario is a good illustration of a random event that may happen during a festival that may attract all the people that are around. This could be for example an influencer tweeting about the unexpected quality of this concert. It could also be something like free drinks given at a certain place in the festival that attracts the people around.

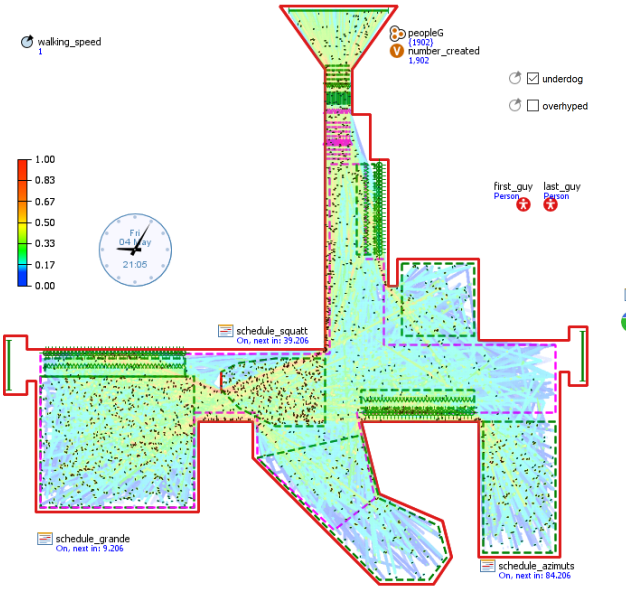


Figure 15: Underdog at 21:05

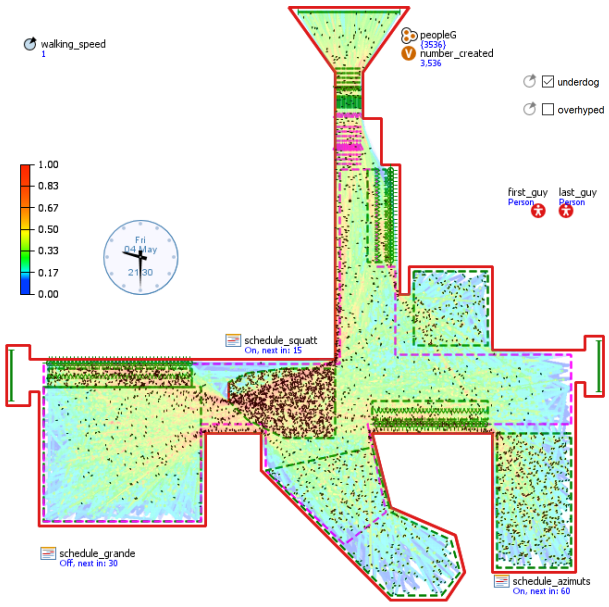


Figure 16: Underdog at 21:30

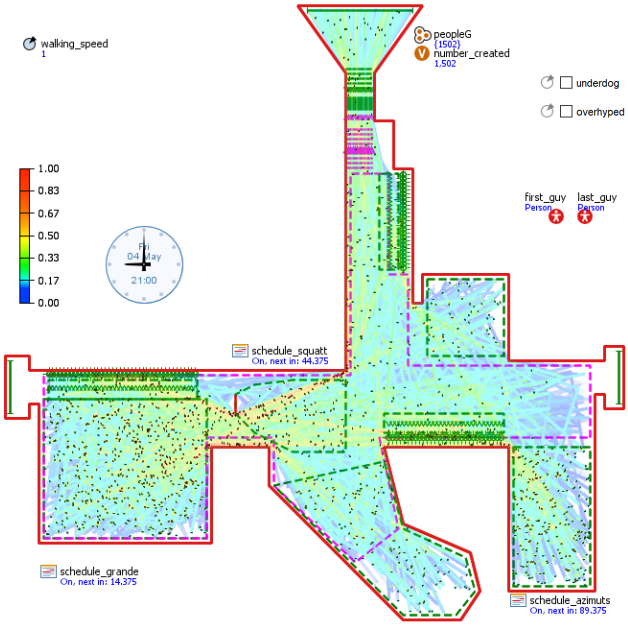


Figure 17: Standard at 21:00

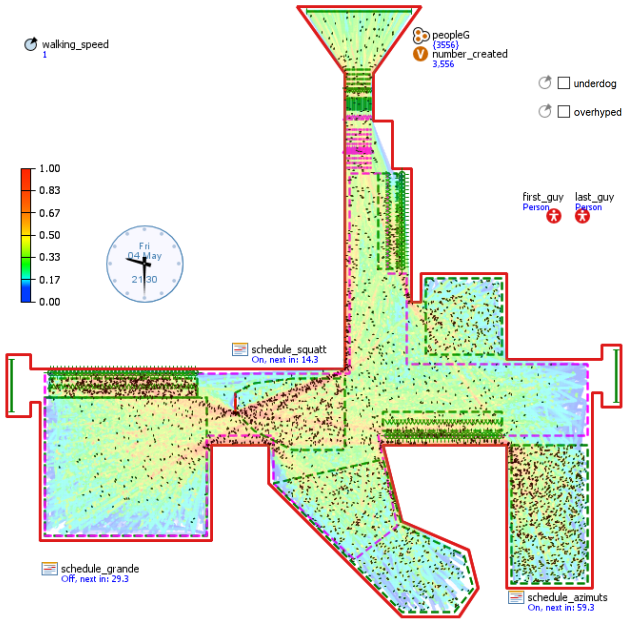


Figure 18: Standard at 21:30

3.4 Overhyped concert

The *overhyped* scenario is paired with the *Orchestre tout puissant Marcel Duchamp XXL* concert. This concert is between 20:15 and 21:15. The reason we chose this concert is that it happens when there is already a good number of people inside Balelec. Moreover, our simulation is quite laggy for the rest of the simulation so we couldn't simulate our scenarios effectively on the later headliners...

To proceed this simulation, we introduce an option overhyped in our model, which represent is a boolean variable. This variable is initially set as `false` however, between 20.15 and 21.15, we can activate it. When we do so, the probabilities in every `Person` agent are modified as following :

- the probability of the person to go to a concert decreases for the "concerter" type

```
if(overhyped)
  p_concert -= 10/(60*100);
```

Listing 5: Extract of the *update_proba* function

- In the function where the festival goer choose which stage to go to, the probability to go to the *Grande scène* decreases as following

```
if (overhyped)
  p_grande -= 35; // same here
```

Listing 6: Extract of the *update_concert_proba* function

Hence this would affect every people regardless of their type.

- Finally, we decrease the time that the person would spend at the concert by 15 minutes, regardless of the type of the person.

```
if(overhyped)
  t_grande /= 4;
```

Listing 7: Extract of the *update_concert_delay* function

In that case the scenario is harder to predict than the one of the underdog concert. In fact for the underdog concert we expected the crowd to gather in front of the stage as in the overhyped scenario, we only expect the majority of the crowd to leave the main stage. Looking from the code point of view, as the probabilities to choose an action are normalized. The people would tend to be less in concert and go more for food or a drink, or for a random walk. Then for those who would still go to attend to a concert, the popularity of the other stages would be way increased.

We can observe on figure 19, that the main stage is less crowded than the one of the standard situation 21. We deduce that if the concert is really bad on the main stage, for different reasons, bad sound quality and so on. The most part of the people would go to an other concert or do an other action. At the end of the concert we can see that in the overhyped scenarios on figure 20 the stalls that sell food and drink by the main stage are far less crowded than in the standard situation 22. We think this to be quiet reflective of reality as when the concert is good, the most part of the people attend it till the end and during that time they get more thirsty and hungry. On the other hand when the concert is not great, the most part of the people would tend to go somewhere else or to buy drink or food without waiting.

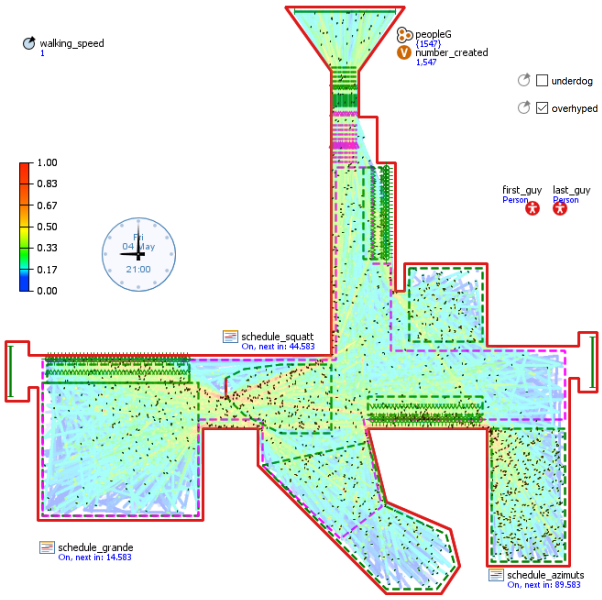


Figure 19: Overhyped at 21:00

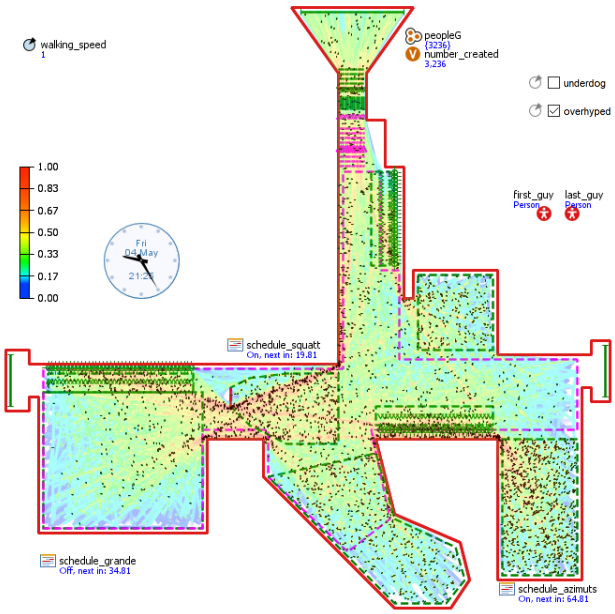


Figure 20: Overhyped at 21:25

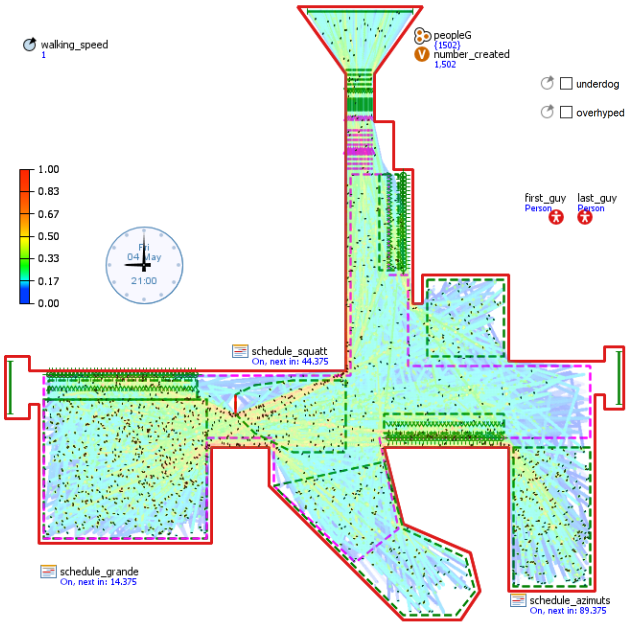


Figure 21: Standard at 21:00

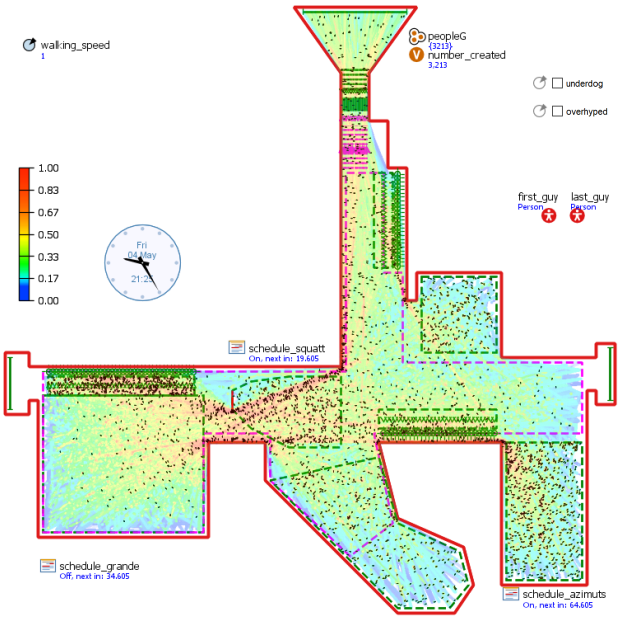


Figure 22: Standard at 21:25

We can observe with our simulation that the effect of this simulation is totally the opposite of the underdog. In that case the people leave a precise place and spread randomly. As in the underdog scenario, the people just gather at a certain place.

4 Discussion

We will address here some of the issue that arose during this project, as well as how we would like to enhance it if we would have more time.

4.1 Scenarios adjustment

Our scenarios do not correspond perfectly what will happen in real life. Indeed, some people don't care of the mass influencer that could say that a concert is good or bad, and stay at the concerts nevertheless. Some people want to go to certain concerts, and will go no matter what.

For the emergency scenario, we would have to consider the stage of chaos that might happen, as well as the fact that some people will stay in the festival even when given the order to evacuate. These people don't represent a large part of the festival goers, but they exist nevertheless and need to be considered.
This could be done by adding a delay specific to the people they wait before beginning evacuating.

4.2 Errors in simulation and difficulties

We run into quite a lot of errors and difficulties during the semester. Fortunately, we were able to fix them. You can find most of them below.

At first, we were limited by the one our simulation time offered by the PLE version of the software. Due to this, we lost a lot of time trying to thing about a solution to counter it. One of our solution was to compress time by a factor of 8, so that one minute in the simulation counted for 8 in real time. However this would have meant more computing, which would be a big obstacle for us.

Another issue was to address the contents of our agents. Indeed, our first implementation was to separate the creation inside the flowchart and in a collection of agent. With this, the simulation was not able to link them together. We didn't see this issue in the beginning as we tested only with one agent to check the flowchart model. When we added more people, a lot of issues arose. Fortunately, we were able to fix this with the help of our teacher.
The new solution consist of creating the agent in the collection directly from the flowchart.

4.3 Simulation problems

Our simulation has three main issues that we couldn't solve.

The first one is the major limiting factor for most of our project. Due to the nature of the festival (14'000 people in 8 hours), our simulation is very heavy in term of computation power. This means that to run the simulation in full, we had to change the presentation:simulation

ratio to 1:16, which means that the computer computing power is focused on the simulation and the presentation (what we see) is very limited. And even with that, as we don't have NASA-level computing power, our laptops couldn't run the simulation well...

The second issue was already shown for the emergency scenario. The pedestrians in `AnyLogic` always take the shortest path from point A to point B. This resulted in many cases where some strategic points were crowded even if the way was clear two meters further. In reality, the festival goers would take this second path, but this is not possible here...

Finally, we had an issue to cancel the agents going in a block. Namely, we wanted to cancel the pedestrian going to a food/drink stand if the number of people in the queue was superior to a number set for each person (between 4 and 15). The code for this is shown in 8 below. The **10* is due to the number of distinct stands in the service block. Unfortunately, this didn't work as we intended for a reason we still don't understand. Some people evacuate, but at some moments in the festival, the blocks contain more people that the limit.

```
1 if(current_service=="eat" && ((Main)getEngine().getRoot()).eat.size()>
    cancel_food*10)
2   ((Main)getEngine().getRoot()).eat.cancel(this);
```

Listing 8: Cancelling code

For example, in figure 23 below, more than 400 people are present in the food/drink stands on the left, which should trigger the cancel for most of these. However, this does not happen...

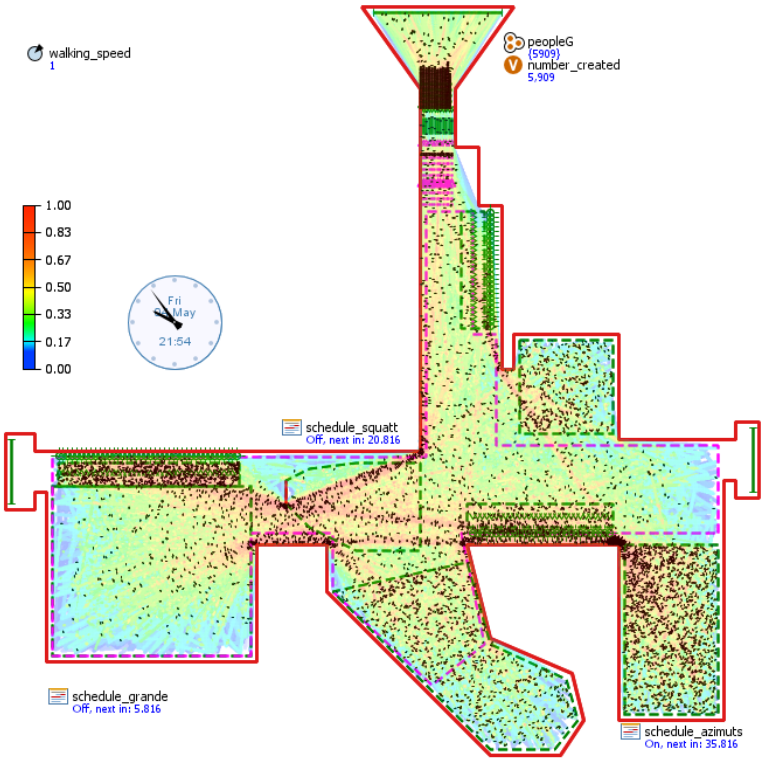


Figure 23: Issue at the food/drink stands

4.4 Future steps

Our simulation is not perfect. In addition to fixing the issues aforementioned, to improve it we could do two main things. The first one is to add grouping mechanism. This could be done relatively easily using the grouping blocks of the pedestrian library. At first we thought about adding it if time allowed it, but due to some other issues, we didn't in the end...
Grouping would add a new precision, as most of the people go to a festival with friends, or meet other people with which they will spend the evening.
An other point to improve, would be to do a survey on festival goer and as them their habits as well as preferences in order to code a better decision maker according to the probabilities of this survey.

Another thing on which to improve is the data we have. All the data used in this project can be obtained on internet. We tried communicating with the Balelec staff, but we decided to do that too late, and all were busy with the final preparation of the festival. Were we to do that earlier, we could have obtained more precise information, for example on the emergency exit placement.

5 Conclusion

Our aim was to have a model reflective of the reality in order to study the mapping of the festival from the point of view of the crowd handling.

We create a model, made of a lot of agent of the same type (`Person`). The moment where it gets interesting is that we define different action that this agent can do and then we code a decision maker, with probability that change dynamically according to some events, the time and the festival goer type. This allow us to create different movements. We notice that it is really hard to change the parameter in an agent dynamically using `Anylogic`.

This agent is the base of our model as it define the movement of the crowd. In order to have something close to the reality, we tried to scope the different action that a festival goer may do. However the movement of the crowd is really random in a festival, as there are no clear rules to define them. Hence we reduce the actions type of a pedestrian to four different actions that for us seems the most important to simulate the flow of people through the festival : eating, drinking, going to a concert and walking around. Hence we don't take into account the social interaction.

Those decisions allows us to simulate the scenarios as the emergency exit and the case of a good or bad concert. These three scenarios were chosen as events that could happen during the festival.
The results of the scenarios allow us to validate our model in itself, by showing that the pedestrian moves like what we wanted to begin with. We can see that the simulation of our scenarios match our expectations. Namely, an emergency will result in a huge concentration of people near the exits (even if this was amplified by the problems shown in section 4.3), a concert better

than expected will attract more people that it have should, which would results in large density where the festival staff didn't expect it, and finally, a bad concert that was planned on a big stage will result in a higher density at the other festival stages.

Overall this project was a very good exercise to the use of `Anylogic`. The software is very powerful in term of the things we can simulate, and this project showed us that it can be applied to a large variety of subject.