School of Computing
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

Crowd Simulation for Museums

Xuchen He

Submitted in accordance with the requirements for the degree of
High-Performance Graphics and Games Engineering MSc

2020/2021

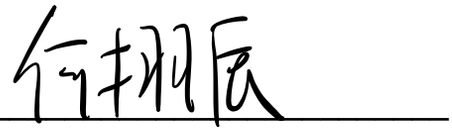
The candidate confirms that the following have been submitted*:*

Items	**Format**	**Recipient(s) and Date**
Project Report	*Report*	*SSO (13/08/2021)*
Software Source Code	*Software codes*	*Supervisor, assessor (13/09/2021)*

Type of Project: Exploratory Software

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) 

Summary

By the impact of COVID-19, after the lockdown for nearly a year, museums are reopening with a sort of COVID-19 safety guidance. Such as social distancing and limited capacity. Therefore museums are facing some problems with some reopening measurements.

In order to helping museums manage the visitors' flow, rooms' layout and exhibits' layout. This project aiming to visualize the movement of a large number of visitors in a museum, which allows the user to create an environment of the museum, with the exhibits and customized visitors using the software delivered. The users are able to see the result real time.

The crowd simulation is the core technic of this project. The crowd simulation was been built with the help of Menge crowd simulation framework. Also, the Unity game engine was being used as the graphical interface tool.

Acknowledgements

I would like to thanks to my supervisor Dr. He Wang for his guidance and advice throughout the project.

Thanks to my family, I could not study here at the University of Leeds without their support.

Table of Contents

Chapter 1
Introduction

Crowd simulation imitate the movement and interaction of a large number of people or any individuals. This technic has been used widely not only in movie and video game industry, but also used to evaluate the crowd management (e.g., simulate the movement of people leaving a large stadium or visiting a large exhibition) (Thalmann and Musse, 2012).

This project focused on visualized the behaviour of people while they visiting a museum, helps the management of the museum's exhibit layout and maximum capacity.

1.1 Problem statement

All of the indoor venues have a maximum capacity limit. It sounds like a simple problem, but whether a unhealthy flow of moving or the guests feeling uncomfortable can cause serious problems. In additional, as the impact of Covid-19, the social distancing is becoming a norm in our daily life. The museums are reopening but with some Covid-19 safety measures, such as the social distancing and the one way system rules, which leads to a limited capacity. Also, these safety rules and capacity needs to be re-considered more frequently as the changing of government's COVID-19 rules.

This is a very flexible project, there is no specific algorithms or tools that must be used for implementation. Therefore, two completely different target of implementing this project has been considered. The first one is focusing on build my own crowd simulation framework that can be used for museum's crowd simulation, in order to get a best performance on simulating the movement and behaviour of agents. The other direction of implementing is to compare the exist tools online, merge them in to a single museum crowd simulation software, so that the software delivered can be more functional.

One of the challenging problem is that the software delivered should be easy to use and comes with a low learning cost, so that it can be used by any kinds of users, whether professional or not.

1.2 Project aims

This project aiming on create a crowd simulation tool just for the museums, focusing more on implement more features than the algorithms of crowd simulation. Which means that no innovative crowd simulation framework is create through this project, more online resources is going to use for implementing. Also, due to the Covid-19, many safety measurements are being applied during the reopening of museums. This project aiming at helping the museums to:

- Reconsider the maximum capacity limit.
- Generate a more efficient layout of placing the exhibits.
- See the visual effect of visitor's moving flow.

Since the COVID-19 safety rules are changing regularly, the scenes of this project should be more customizable and easy to update in the future, which means that users are able to control more properties of exhibits and visitors.

1.3 Project objective

- Build a tool using Unity Game Engine, visualize the museum rooms and the movements of a large group of visitors (1 to 300).
- Be able to customize the museum model.
- Customized exhibits and visitors' properties.
- Simulate people with independent behaviours, such as different personality and different rules (visitors and museum staffs).
- Real-time simulation. At least 30 frames per second.

1.4 Project deliverables

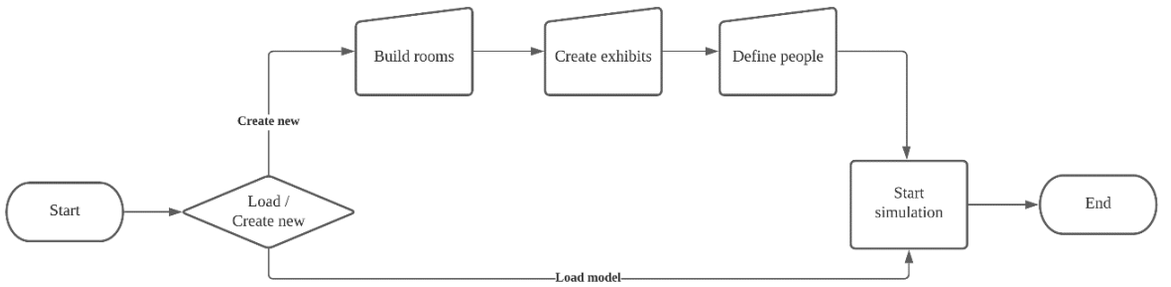


Figure 1.1 software flow chart

A PC software based on Unity 3D is planning to be created with the following features:

User interface:

- Show a 3D environment.
- Customize the number of visitors, see different effect based on the number of visitors.
- Simulate the people's moving flow.
- Visualize the crowdedness.
- Give an overall score of current state.
- Be able to load 3D single floor model of museums with customized file format.

- Be able to create a simple single floor model.
- Has different room types, such aa the ticket centre, toilet and the gift shop.

People:

- Simulate different behaviour, such as walking, stop and focusing on one single exhibit and going to the exit.
- Different rules, staffs and visitors.
- Independent personality.
- Different visitor types, families, Groups and individuals.

Exhibits:

- Visualize the crowdedness of each exhibits.
- Be able to move/add/remove exhibits.
- Exhibit properties.
- Inaccessible space.

1.5 Methodology

This project followed the agile software development methodology, in order to keeping the development of this project heathy. The steps of agile software development methodology defines as follow:

1. *Requirements*
2. *Design*
3. *Development*
4. *Testing*
5. *Deployment*
6. *Review*

1.6 Development plan and timeline

To apply the agile software development methodology into this project, the implementation has been split to several tasks:

1. Background research
    1) Learn the Unity Game Engine skills
    2) Search for the current covid-19 rules of museums.
    3) Search for the museums' room layout in real-life.

4) The measurement of crowdedness.

5) Search the existing frameworks for crowd simulation, have a good understand of each algorithm, see the examples, compare the performance and find the most suitable one to use.

2. Design

1) Decide what features is essentially needed.

2) Decide the algorithms and frameworks needed.

3. Development

1) Build a Unity scene, with a simple museum model.

2) Merge the chosen crowd simulation framework.

3) Implement more functions based on the software built.

4. Testing

1) Build several museum scenes with different properties for testing.

2) Test the performance, correct the measurement of crowdedness score.

3) Optimize the code and algorithm.

5. Documentation

1) Merge the documentation for each function.

2) Finish the final report

The figure 1.2 below shows the timeline of this project. August

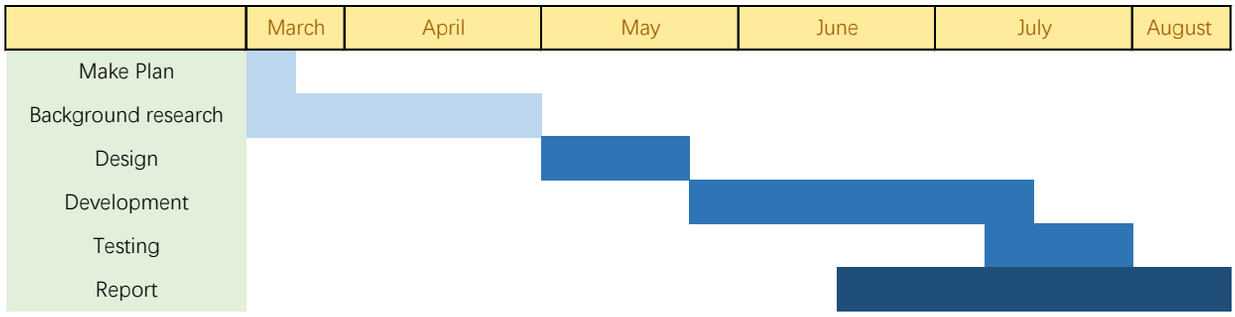


Figure 1.2 Project timeline Gantt chart

1.7 Project completeness

During the background research state, I noticed that the features needed for the designed target is overloaded for this project. Therefore, in order to complete this project on-time, some features were dropped or moved to the future work. The following table shows the completeness of the designed targets.

Feature	Completeness	Note
3D user interface	finished	/
customized number of visitors	finished	/
different rules of people	finished	/
people's moving flow	finished	/
visualize the crowdedness	finished	/
simulate different behaviour	finished	/
people different rules	finished	/
crowdedness of each exhibits	finished	/
load 3D single floor model	Partially finished	See chapter 3
create model	Partially finished	See chapter 3
move/add/remove exhibits	Partially finished	See chapter 3
exhibit properties	Partially finished	See chapter 3
inaccessible space	Partially finished	See chapter 4
different room types	negative	/
independent personality	negative	Hard to implemented
visitor types	negative	/

Table 1.1 Project completeness

1.8 Report structure

This report has been divided into 6 chapters.

Chapter 1 is the introduction, gives an overview of this project. The problems of this project has been discussed, and also states the project's aim, objective and deliverables. The planning and completeness has also been discussed.

In chapter 2, the background knowledge of all the tools, frameworks and museums has been introduced. It first covers the impact of COVID-19 to the museums and how it impact to this project, followed by the research for visitors' movement flow while visiting museums. Then chapter 2 briefly the understanding of algorithms and tools (i.e. crowd simulation, Menge and Unity) that been used for implementing the crowd simulation for museums.

Chapter 3 and 4 focus on the design and implementation of the software. Chapter 3 outlines the design of the software, as well as the implementation through Unity editor. Following by the development of Menge crowd simulation framework introduced in Chapter 4.

The testing configurations and result shown in Chapter 5. Compared the different performance between distinct scenes.

Chapter 6 gives a conclusion of this project, as well as what features are dropped and what is the future planning.

Chapter 2
Background

2.1 The impact of COVID-19

From March 2020, the COVID-19 pandemic impacted the museums all around the Europe. The Network of European Museum Organisations (NEMO) did a survey on the impact of COVID-19 situation on museums around Europe. From their final report, nearly 1000 museums in 48 countries responded their survey. The result of this survey provides following information:

- The museums are planning for reopening to public (Most of them reopened now in 2021).
- 3 out of 5 museums lost a large amount of money every weeks.
- Museums are providing digital services online, but this could not solve their capital issue.

Museums need the income for maintain their operation, and also more importantly, people want the museums back. By the government's COVID-19 rules, museums are allowed to reopening after the lockdown. However, the rules of visiting museum are changing significantly during the Covid-19 pandemic to ensure the safety of visitors.

The UK government provided a guidance for reopening of museums, galleries and the heritage sector. This guidance points out:

- Reduce the number of visitors in an exhibition at same time.
- Plan the one way route to manage the flow of visitors.

Although this guidance published in 2020, the rules above are still being used in most of museums around UK in 2021.

The software delivered through this project is able to helping museums reopening under this guidance, measuring the capacity and manage the flow of visitors.

2.2 Crowd simulation

Any collections of individual agents, whether a group of humans, animals or vehicle flows, is crowds. The behaviour of agents in crowd is always different from they should act as a individual only. The crowd simulation in computer graphics area was been introduced in 1980s, and it has been used in many research fields today.

In the filming industry, the crowd simulation has already been used widely in order to reduce the cost of hiring actors and capturing the shots that needs a large group of people. In 2001, the famous film Lord of the Rings directed by Peter Jackson brought the crowd simulation technology into the filming industry.

Figure 2.1 Lord of the Rings

Nowadays, some software has been implemented that provides an user friendly interface for artists to creating their scenes. Golaem Crowd is a plugin released in 2011, for Autodesk Maya, a CGI software. This plugin allows the user to customizing the activities of agents, such as walking, waving, sitting etc.

Figure 2.2 Golaem Crowd application

The crowd simulation has also already being used widely for measuring the flow of people. Several existing software can be found online.

The Oasys MassMotion is a crowd simulation software developed by Oasys. It allows users to measuring the interaction between pedestrians using the set of behaviours and analyses provided. The social distancing can be easily seen on the software's user interface (figure 2.3), people that closed to each other are being separated by the people that keeping the

social distance with others using different colour. The queueing system can also be built easily using this software.

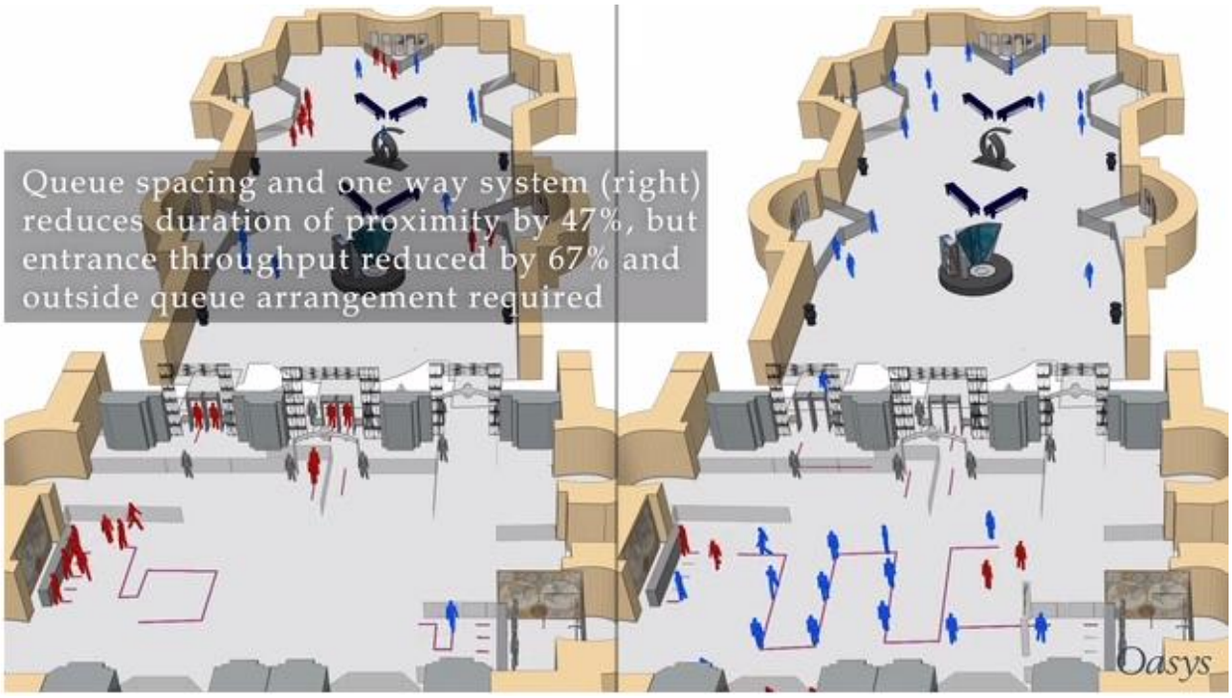


Figure 2.3 Oasys MassMotion

2.3 What is visualization and why

In crowd simulation, the data and result is always represented by visualization.

Visualization is any graphical representation technique of any data information, which aiming for better communication with the data. A visualization must meet the following three minimal criteria:

- The data that been visualized must be a non-visual data. For example, the image processing is not visualization.
- Must produce a graphical representation. Also, the major methods of communication between data and users must be the visual, only the additional information can be showed by other methods.
- The final result delivered must be human readable and understandable. Which means that a more convenient way of learning the data should be provided by the visualization.

The data visualization is now widely used in many different business area through data analysis, helping them to create a graphical environment of data by the most efficient way possible. Taking the raw data, model it and gives user a much more readable and recognizable representation, this is the one of the easiest way to understand and interpret data.

2.4 Unity

Since this project is focusing on develop a software that merged to tools available online, in order to build a graphical user interface, a game engine should be used. Two game engines has been considered, Unity and Unreal, both of them are mature and widely used.

However, since this project does not requires any complex graphical techniques, the only different between these two game engines is the learning cost. Therefore, the Unity game engine with more tutorials and larger community has been used in this project.

Unity is a cross-platform game engine developed by Unity technologies. The Unity course 'Game Design and Development with Unity 2020 Specialization' available on Coursera has been used as the hands-on tutorial.

2.4.1 Unity project architecture

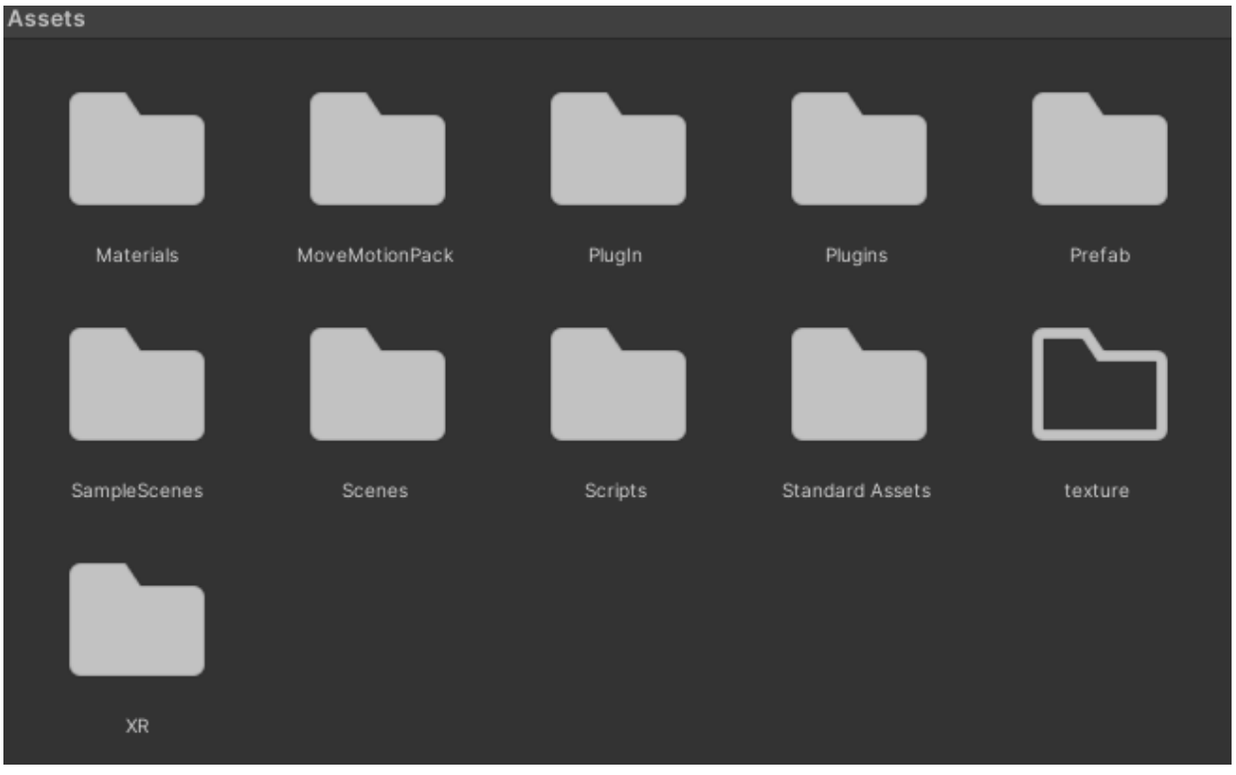


Figure 2.4 Unity file structure for this project

- Materials: Game object materials folder. For example, the green, yellow and red material for the exhibit cylinders.
- Plugins: Plugin tools.
- Prefab: Stores the standard floor, wall and exhibit game object, used for modifying the scene easily.
- Scene: Different scene created, large and small museum model, different exhibit layout.
- Scripts: Script files written in C#.

2.5 Menge

Similar to the game engine, a crowd simulation framework should be used for this project.

The Menge is an open source modular framework developed by University of North Carolina – Chapel Hill, which aiming at a powerful, cross-platform modular framework designed for simulating the movement of agents in a crowd.

The core crowd simulation part of this project is powered by Menge, the Menge framework and the dynamic link library file (.dll file) provided by Menge has been used in this project. More details of the algorithms of Menge and also the implementation of this project can be found at chapter 4.

2.6 Menge for Unity

Since the Menge crowd simulation framework is written in C++. A dynamic link library has been provided by Menge through their GitHub page in order to implementing using Unity. A Dynamic link library is a type of file that contains the dynamic links to other programs. Which means that several different programs can be called through one single ddl file. The Unity editor allows developers using dynamic link library files as plug-ins.

Chapter 3
Software Design

3.1 Overview

This chapter gives the details of the implementation and deployment of this project. The design of the software, the reason and development of all the essential features delivered through this project, and also the core algorithms of the Menge crowd simulation framework are introduced through this Chapter.

3.2 Software user interface

The user interface of the software implemented is shown by the screenshot below.

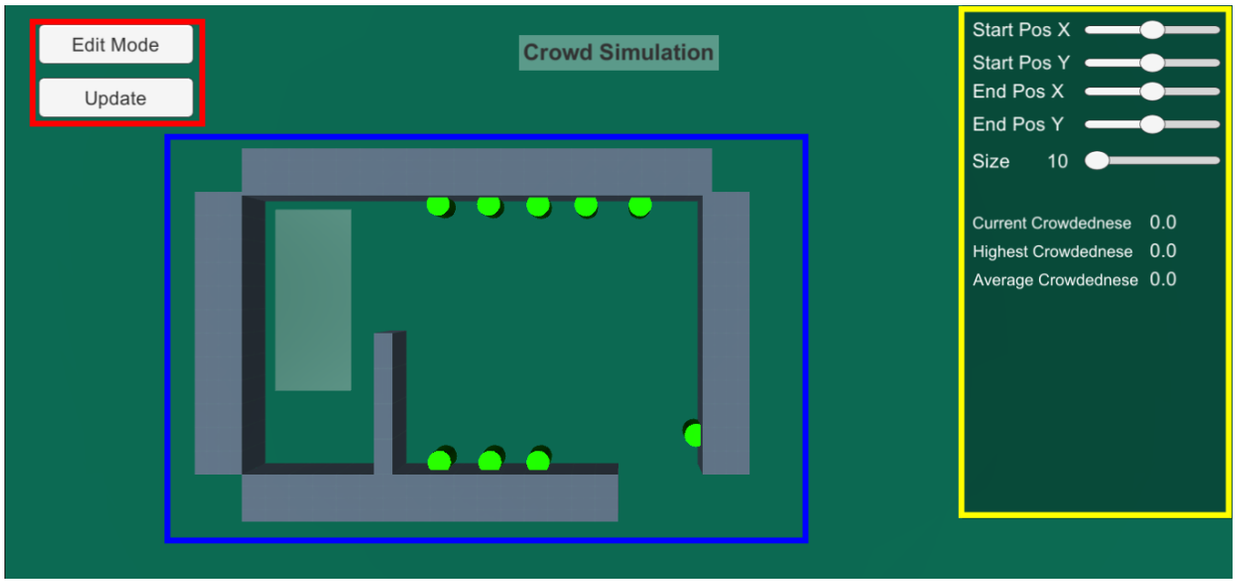


Figure 3.1 Software user interface screenshot

The three major areas of the user interface are introduced in the following three sub-sections.

3.2.1 Blue area: Visualization display

The model of the museum is shown in the main area of this software. Figure 3.1 shows a simple single room floor plan of a museum. The dark blue cuboids are walls, divide the area into several rooms. In the example shown in Figure 3.1, the entrance is on the left hand side, and on the right hand side is the main exhibition room, the exit in on the bottom right corner. The cylinders marked as green in the exhibition room is the exhibits.

During the simulation running, three states are given to the exhibit cylinders, green, yellow and red. Green represents normal, yellow for few visitors are currently visiting the specified exhibit, and red if the exhibit is overcrowding.

Figure 3.2 gives an example of different states of exhibit cylinders. The exhibits with 1 or zero visitors are marked as green, yellow cylinders for two to four visitor and red for five and more than five visitors.

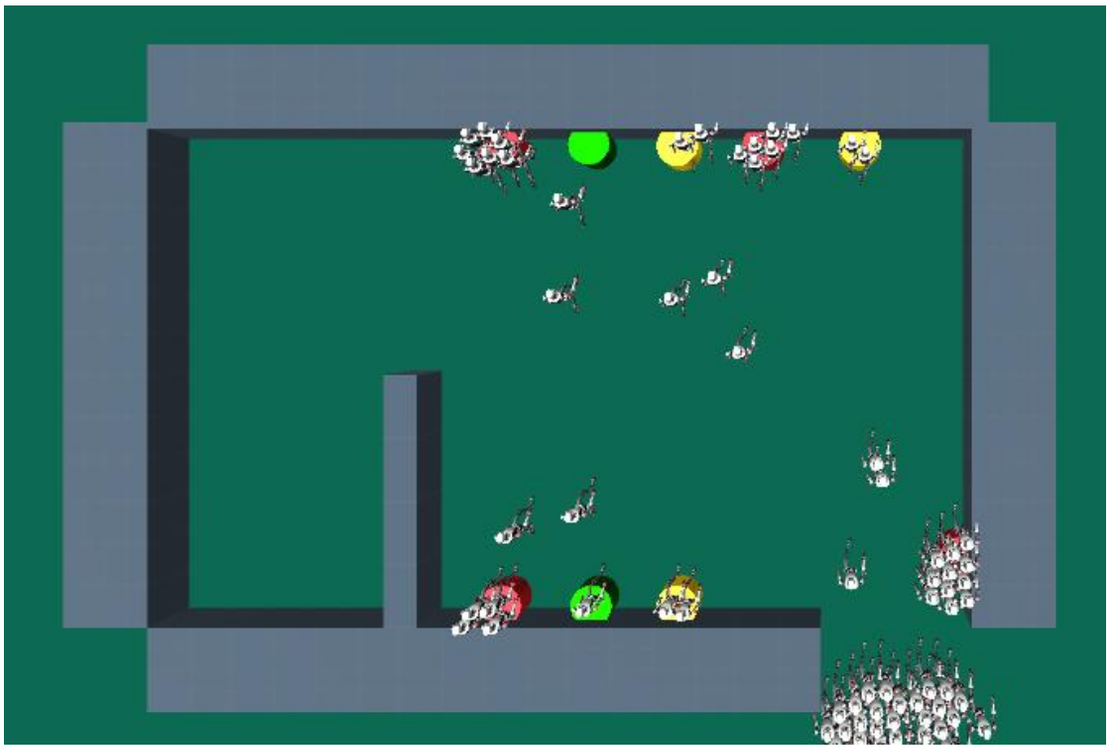


Figure 3.2 On simulation running screenshot

3.2.2 Yellow area: Scene customize and result display panel

The properties of current scene can be modified in the yellow area. Include the start and end position of the visitors and the number of visitors.

More information of current crowd simulation is also shown in the yellow area. Current crowdedness represents how crowded of the entire current simulation is in real-time, updated three times per second. The highest crowdedness records the highest score during the entire simulation, and also the average crowdedness is on the third line. During the simulation is running, as the Figure 3.3 shows, the score of crowdedness has been divided into three group. Green scores means the crowdedness is healthy and under control, yellow scores stands for a little bit crowded in current simulation, while the number of visitor must be decreased if you see a red score.

Section 3.5 provides more details of how the score of crowdedness has been calculated.

Figure 3.3 Crowdedness display screenshot

3.2.3 Red area: Main control panel

There are two major modes designed in software, edit mode and play mode, controlled by the first button in the red area.

Below the mode control button is the update button, designed for updating the xml files linked to the Menge framework.

3.3 Edit mode

As Chapter 1 mentioned, one of the main feature planned for this project is customized museum model. This feature is marked as ‘partially finished’ at this moment. Current version of software does not allows users modify or create a new museum model directly through software’s user interface. However, this does not means that this project delivered a fixed model only. The model can be modified simply through the Unity editor, the data is updated automatically to the Menge framework before the simulation start running (Section 3.6).

3.3.1 Current delivered customized scene

There are two major part of a crowd simulation scene, the museum model and visitors.

Users can edit the museum model through the Unity editor before running the current scene. Prefabs has been prepared with independent tags, such as floor cuboid, wall cuboid and exhibit cylinders. Only single floor model is supported for current version, which means that all of the prefabs must be placed on the same floor plane.

For visitors, which is the crowd simulation target of current project, there are also several features can be customized. While the software is running, on edit mode, users are allowed to modify the start and end position of the visitor's flow using the scene customize and result display panel (Section 3.2.2) . The start position displayed as a transparent grey area (on the left of Figure 3.1), the size of grey area is changed by the number of visitors.

3.3.2 The future plan of customizing museum model

Of course, it is definitely inconvenient that all the models can only be modified through the Unity editor, not every user have Unity installed, and also it does not meet the project original plan. The target is that all the models can be create, save and load through the software delivered only.

3.4 Play mode

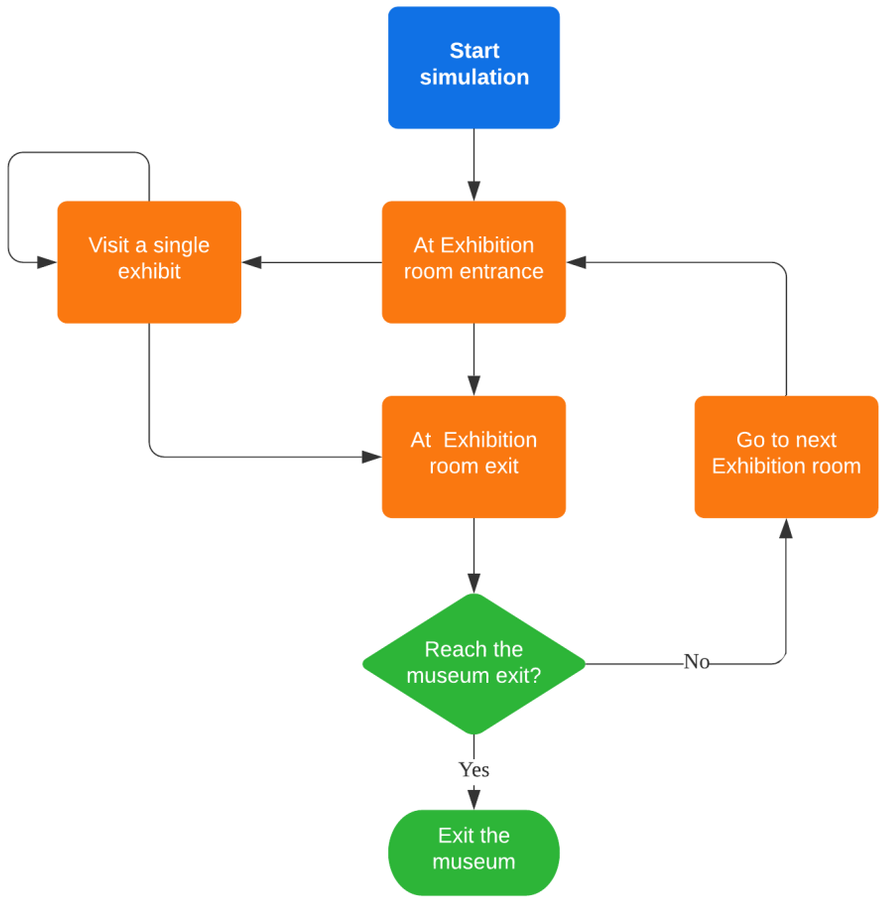


Figure 3.4 Agent's visit flowchart

Once the model and visitor has been decided, the software is now ready for simulating the visitor's movement in the museum, by simply press the update and play buttons.

Figure 3.4 introduced how a single visitor is designed for visiting the museum.

- Start simulation: The initial state of every single agent, start simulation at the start position user specified during edit mode.
- At exhibition room entrance: The state for a agent that just reached the entrance of a exhibition room. Each agent are now going to decide to visit a exhibit in this room or just heading to the room exit straight away.
- Visit a single exhibit: While the agent approached a single exhibit picked on previous state, the agent will stop moving, entre this state and a randomly stop time from 3 to 10 second. After that, the agent can be whether exit the current room or continue to next exhibit in this room.
- At exhibition room exit: An agent at this state means that he is exiting an exhibition room. Go to the next exhibition room if not reaching the end of the tour.

The simulation ended automatically after all the visitors reached the museum exit.

3.5 How the crowdedness score been calculated

As the Section 2.4 mentioned, the result of visualization must be understandable and readable. The crowdedness of the simulation is represented by a score shown in the software user interface. This section focusing on measures that used to quantify the level of crowdedness for this project.

3.5.1 Grid-based measures

In this method the space is being split into orderly grid. In order to measure the crowdedness around agent A at time T, the number of agents with in the cell C that is counted, where cell C is the cell that agent A stays at time T. The detailed definition can be mathematically be formulated as follows:

$$\overrightarrow{x_p}(t) \in X_c \Rightarrow \rho(c,t) \qquad (1)$$

$$\rho(c,t) = \frac{\sum_q n_q(t)}{A_c} \qquad (2)$$

$$n_q(t) = \begin{cases} 1, & if\ \overrightarrow{x_q}(t) \in X_c \\ 0, & otherwise \end{cases} \qquad (3)$$

Where p is the agent that we want to compute the density, $\overrightarrow{x_p}(t)$ and $\overrightarrow{x_q}(t)$ stands for the coordinates of the specified agent p and any agent q at time t. X_c represents the set of coordinate with in the cell c. $\rho(c,t)$ is the density of cell c at time t. A_c represents the area of cell c.

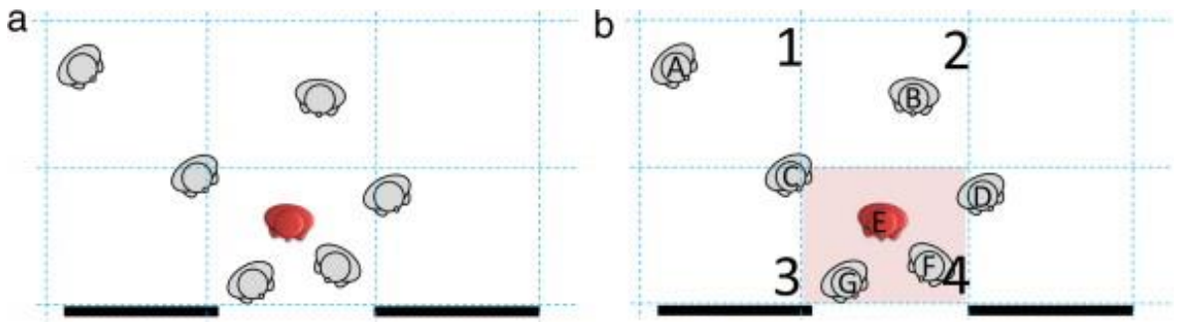


Figure 3.5 Grid-base calculation method (Dorine et al., 2015)

Figure 3.5a gives an graphical overview of grid-base method, and Figure 3.5b is an example of measuring crowdedness around agent E. Assume the size of each cell is 1 metre by 1 metre. The crowdedness around agent E can be estimate as $\frac{0+0+1+1+1+1+1}{1} = 5$.

3.5.2 Apply the grid-based measures into museum crowdedness measurement

Measuring crowdedness of a museum is different from other scenes. Unlike a usual crowd simulation scene that agents are keep moving all the time during the simulation, the agents in a museum scene are usually stopping at a exhibition. For example, figure 3.6 is a heat map that provides a real-world total number of agent's interaction with exhibitions. Where an agent state changing (i.e. from moving to stopping or from stopping to moving) is counted as a interaction. As the heat map shows, all the state changes happens around exhibits.

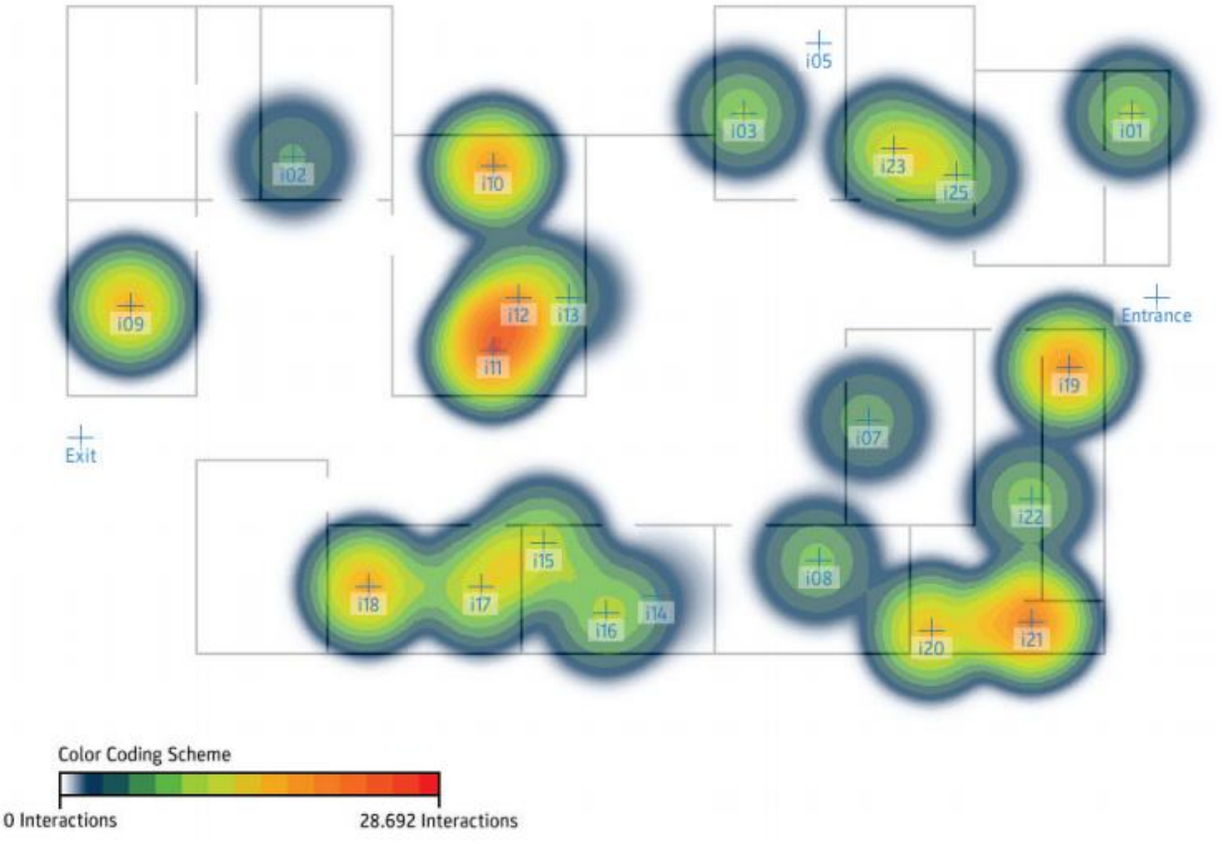


Figure 3.6 Number of interactions at each computer-based exhibit (Strohmaier et al., 2015)

Back to the grid-based measurement, since all the agents are moving between the exhibits, the crowdedness around the exhibits would be much more important than other areas in the exhibition room. Therefore, only the crowdedness around each exhibit is considered to the overall crowdedness score of this project .

The grid-based structure makes it more easier to find the agents that are less one metre to an exhibition. The crowdedness is exponential growth by the increasing number of visitors.

The size of each single cells in the grid structure has been set to 1 metre by 1 metre. Therefore, for every single exhibit, the cell that contains the exhibit and also the 8 cells around are searched for the agents around it. The pseudo code of calculating the crowdedness shown as follow:

1. **for** each exhibit in the exhibit list **do**:
2. ---- find the cell contains current exhibit
3. ---- **for** the cells around the exhibit **do**:
4. ---- ---- **for** the agents in current cell **do**:
5. ---- ---- ---- **if** the current agent stays less than 1 metre to the exhibit **do**:
6. ---- ---- ---- ---- $\text{current exhibit pressure} \mathrel{+}= 1$
7. ---- ---- ---- **end** if
8. ---- ---- **end** for
9. ---- **end** for
10. ---- $\text{crowdedness of current exhibit} = \mathit{current\ exhibit\ pressure}^3$
11. **end** for
12. $\text{total crowdedness} = \frac{\mathit{sum\ of\ every\ exhibit's\ crowdedness}}{\mathit{number\ of\ exhibits}}$

After the simulation ends, a highest score and a average score is given to the user. The score is been shown by three colours, green, yellow and red, represents normal, slightly crowded and overcrowded respectively. The measurement of the crowdedness's green, yellow and red discussed in chapter 5.

Figure 3.6 Crowdedness around the exhibits

3.5.3 Grid-based measures data structure

Figure 3.7 Grid-based measures classes

Agent class:

- agent: stores the Unity game object agent, used to get the position of agents.
- next: holds a pointer to an Agent that in the same cell.
- inGrid: a boolean that true if the agent is in the room boundary, false if out of the room boundary.

Cell class:

- firstAgent: holds a pointer to the first Agent in the current cell.

Exhibit class:

- exhibit: stores the Unity game object exhibit, used to get the position and change the colour.

RoomBoundary class:

- Holds the position of four corners of the rooms' boundary.

A linked list data structure is being used for the grid. Each cell holds a pointer to an agent in that cell, and that agent also holds a pointer to another agent in the same cell.

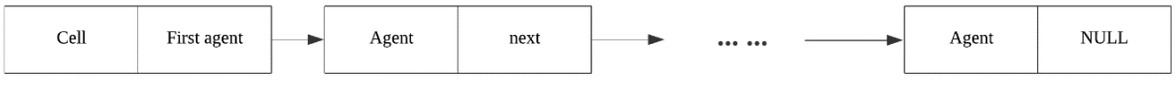


Figure 3.8 Linked list data structure

A 2-dimentional array of Cell is been created in order to store the cells. The update function is called three times per second in order to balancing between performance and the result. Below is the pseudo code of update function :

1. *// first update the agents outside the room boundary, see if it went in to the room boundary.*
2. **for** each agent in the agent list **do**:
3. ---- **if** agent is in the room boundary **then** continue.
4. ---- check the new position of current agent
5. ---- **if** agent till out of the boundary **then** continue.
6. ---- store current agent into the grid.
7. **end** for
8. // update the agents in the room boundary.
9. **for** each cell in the 2-dimentional array **do**:
10. ---- current agent = the first agent in current cell.
11. ---- parent agent = null.
12. ---- **while** current agent != null **do**:
13. ---- ---- **if** the current agent still in current cell **do**:
14. ---- ---- ---- parent agent = current agent.
15. ---- ---- ---- current agent = currentAgent.next.
16. ---- ---- ---- continue.
17. ---- ---- **end** if
18. ---- ---- delete agent from current cell, parentAgent.next = currentAgent.next.
19. ---- ---- **if** current agent is out of the room boundary **do**:
20. ---- ---- ---- currentAgent.inGrid = false.
21. ---- ---- ---- parent agent = current agent.
22. ---- ---- ---- current agent = currentAgent.next.
23. ---- ---- ---- continue.
24. --- ---- **end** if
25. --- ---- assign current agent to the end of the linked list of new cell.
26. --- ---- parent agent = current agent.
27. ---- ---- current agent = currentAgent.next.
28. ---- **end** while

29. ***end*** for
30. update the crowdedness score

Chapter 4
Menge crowd simulation framework and Implementation

In this project, the core functions of crowd simulation is powered by the Menge crowd simulation framework.

4.1 Menge's architecture

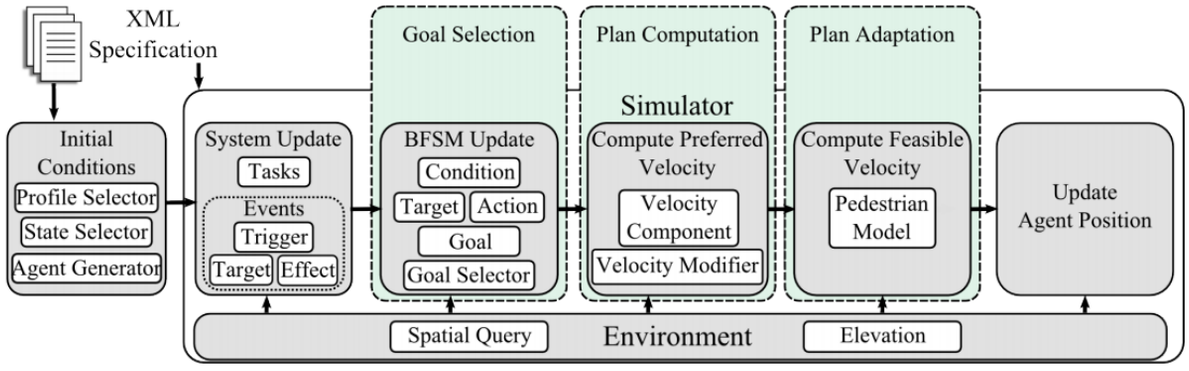


Figure 4.1 Menge's computation pipeline (Curtis et al., 2016)

Menge's modular architecture is based on the concept of elements (Curtis et al., 2016), which represented by white boxes in Figure 4.1. Each type of element stands for a particular aspect of a subproblem, and also a particular solution is being implemented for the interface that element type defined. XML files are bring used to instantiated the implemented elements.

4.2 Menge's XML specification

Three XML files are accepted by Menge, scene specification, behaviour specification and view specification. In particular the Scene and behaviour specification must be provided in order to start a crowd simulation. The following three sub-sections are going to introduce these three XML files.

4.3 Scene specification

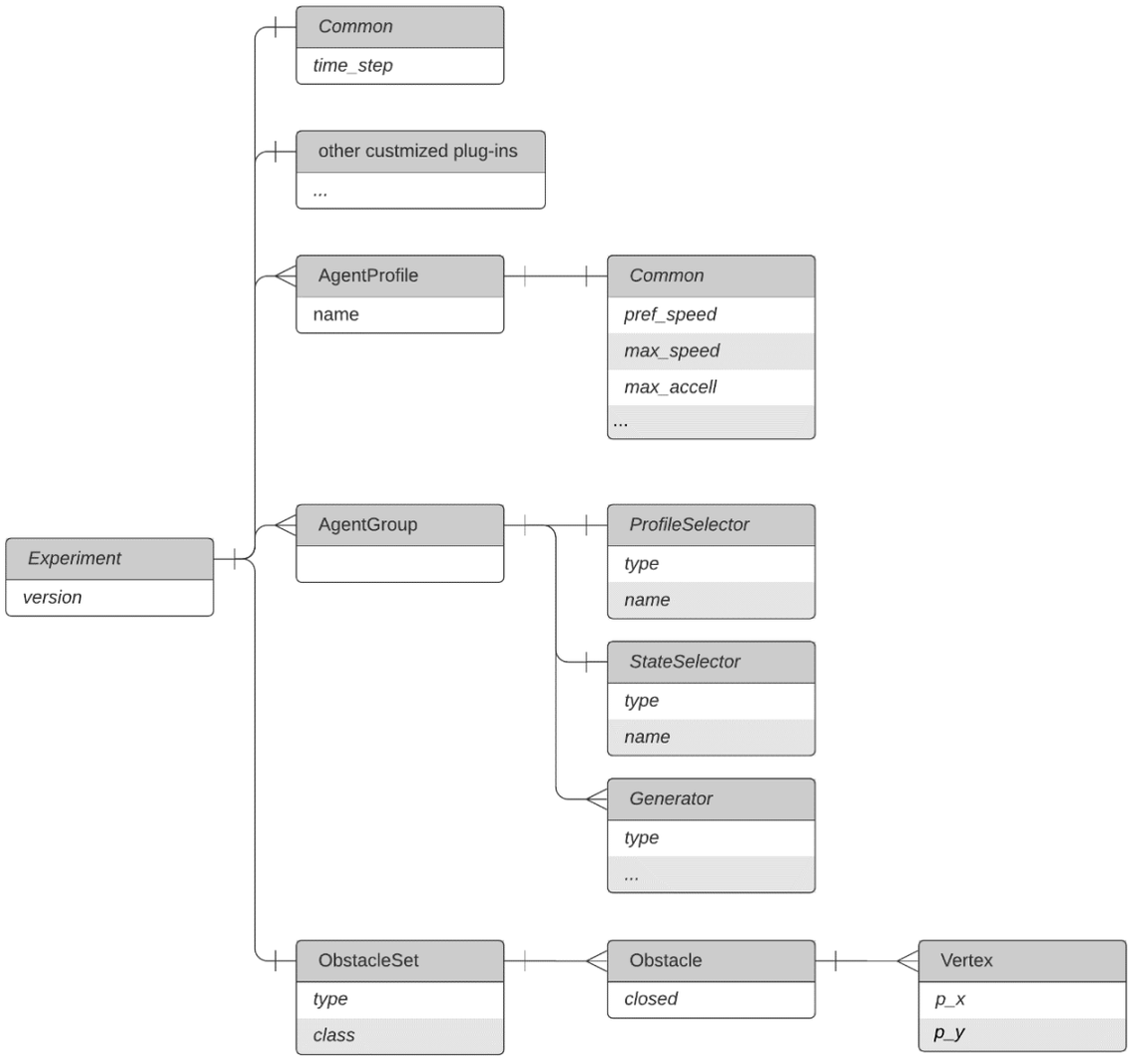


Figure 4.2 Scene xml file structure

In the scene specification XML file, four essential types of element must be declared, the simulation properties (common and other customized plug-ins in the figure above), agent profile, agent group and obstacle set.

- Simulation properties: This element defines all the details of agents and also the simulation. Such as the body force, reaction time, wall distance and obstacle. There are two properties that must be defined before the simulation start. The time step of common element must be declared in scene XML file, which basically means how fast the simulation running. Another element needed is the spatial query. It provides an interface to perform visibility and proximity queries. In this project, kd-tree-centric class has been used.

- Agent profile: One scene XML file must have one or more than one agent profile. An agent profile defines a collection of additional properties for agents in more details, such as radius of disk, response time, etc. In Menge it is called behavioural state or b-state. The changing of behavioural state will affect the calculation of the trajectory of an agent, which leads to different behaviours. For example, in the museum crowd simulation, a preferred speed, maximum speed and maximum acceleration is given to every single agent in order to control their movement speed in the scene.
- Agent group: Same as the agent file, one or more than one agent group element must be defined in the scene XML file. An agent group is where a collection of agents been defined. In this project, three sub-elements are added into each agent group, which is profile selector, state selector and agent generator. The profile selector assigns an agent profile defined in previous element to each agent who belongs to current agent group. To be noticed that each agent profile should have a unique name assigned, so that the profile selector can be able to find a specific profile by the unique name. State selector is similar to the profile selector, but assign the initial state for agents from the states defined in the behaviour XML file. More details of state element will be introduced in Section 4.4. The agent generator used to generating a group of agents and assign them more initial properties, such as the start position. In current project, only a single group has been defined, a two-dimensional array of agents is been applied to the start group of agents, which is a rectangle grid.
- Obstacle Set: Different from the two elements introduced above, Obstacle can be defined zero or more times. A single obstacle set specify a group of impassable walls into the simulation, which is the boundaries of the rooms. For a single obstacle in a obstacle set, it allows for the explicit instantiation of obstacle by a vertex array, or even a navigation mesh for more complex obstacles. In this project obstacles are declared by vertex array.

4.4 Behaviour specification

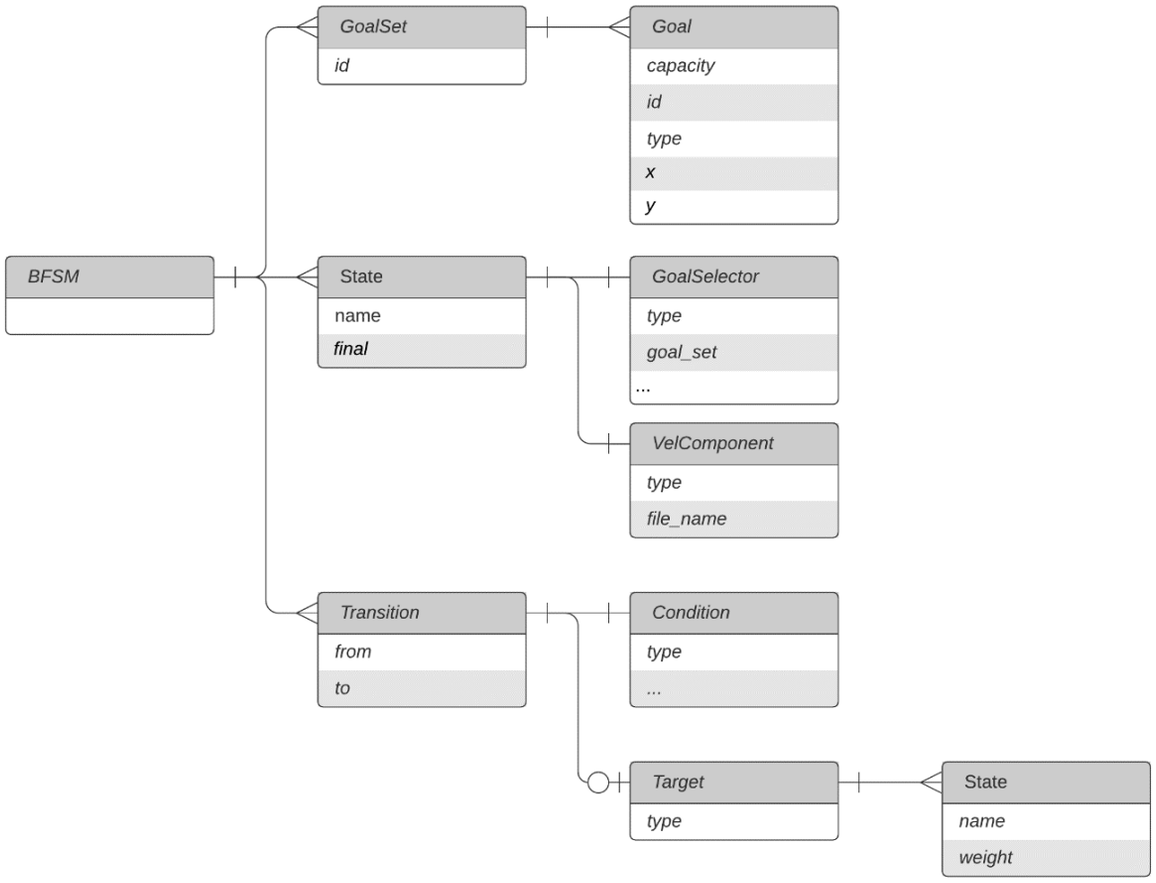


Figure 4.3 Behaviour xml file structure

The behaviours, i.e., movements and targets during the simulation of agents, are defined by behaviour XML file. There are three essential types of elements must be declared in the behaviour file, which is goal sets, agent states and transitions.

- Goal set: A goal set with a unique id contains a set of goal elements. The goal element is designed for declare the space or a coordinate that agents want to reach. An agent's goal is a region in two-dimensional space, which can be a point, a circle, an axis-aligned box or an oriented box. An agent must been selected a goal at any time during the simulation if the agent is moving. In default, the agents will move towards the nearest point in that goal's region. In additional, each goal have a capacity, if one goal reach it's maximum capacity, no more agents can be assigned to that goal.
- Agent state: Each agent must be assigned to a specific state at any time during the simulation running. An agent state defines an agent's current behaviour. In this project, two major types of agent state has been implemented, staying at a target and going to a target. To be noticed that the 'final' component is marked as true if an agent state is a final state, which means the end of simulation. Goal selector and velocity component has

been assign to both of two types of agent state. The goal selector element assigns an agent a goal when an agent enters the current state. Menge provides a wide range ways of selecting a goal, includes pre-defined goal, a uniform or weighted random selection, nearest or farthest from a goal set element. In this project, the states that assign an agent to go to a exhibit has been set to weighted randomly selection, while other states is a single chose pre-defined goal, such as the room entrance and exit. The velocity component element used to computing the preferred velocity of an agent. In most of the states in this project, the default value has been applied. However, in some complex states, such as transform from a exhibition room to another exhibition room, a road map must be used. For example, in figure 4.4, the agent A wants to go to the goal G with an obstacle placed in the middle of the road. A road map is needed for guiding the agent heading to the goal G, which is the green line in the figure below.

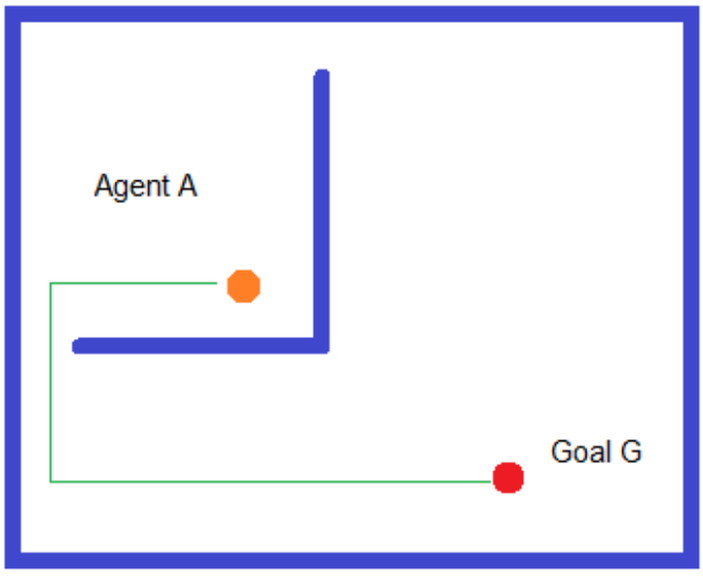


Figure 4.4 Example of a road map

- Transition: The transition element defines all the transitions between the states. An agent is able to change from a state A to a state B only if there exist a transition T that declares a transition from state A to state B. A state defined can be connected to multiple out-going transitions. Two types of transition is being used in this project, one with a specific target state, and the other with several target states and weighted randomly choose from the target state set. Both of them contains a condition element. The condition element defines a transition by how the transition should be done. For example, the condition of the transition from agent visiting an exhibit (stationary state) to other states is a timer, which randomly select a stopping time between 5 to 20 seconds at a exhibit. The agent will stay at the exhibit state (visiting an exhibit) until the timer expires. On the other hand, the type of condition for the transition from a walking state (e.g. walking to a room or walking to an exhibit) to a stationary state is ‘goal reached’, with an approaching distance. Which means

that the state transition will make automatically when the goal assigned by the goal selector has been reached with in the approaching distance. For the transitions that defines a target condition set, a target element with a weighted states set is applied to them.

4.5 Menge's examples

One of the most efficient way of learning a framework or a tool is by discovering their documentation. However, as the figure 4.5 shows, the Menge's documentation is still not completed. Only very limited information we can get from the documentation. However, Menge provides a lot of example scene that defines multiple types of scenes.

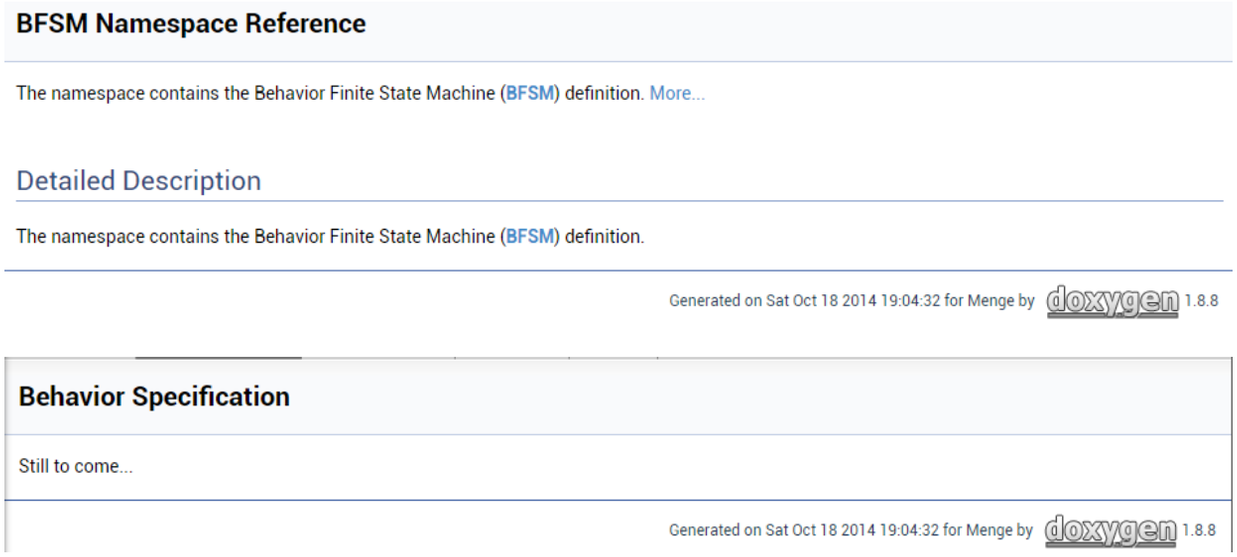


Figure 4.5 Menge's documentation

In my opinion, the best way of learning how to use Menge crowd simulation framework is the example scenes provided. For the crowd simulation for museums, the most similar example is the office scene. The office scene demonstrates the most complicated BFSM (Behavioural Finite State Machine) in the set of examples. Most agents have several states and transitions, such as going to a desk, working at a desk and leaving the building. Which is very similar to this project in the states and transitions architecture. Furthermore, the office scene also use the road map for navigation. Therefore, I got most of the ideas from here and applied to this project.

Figure 4.6 Menge's examples

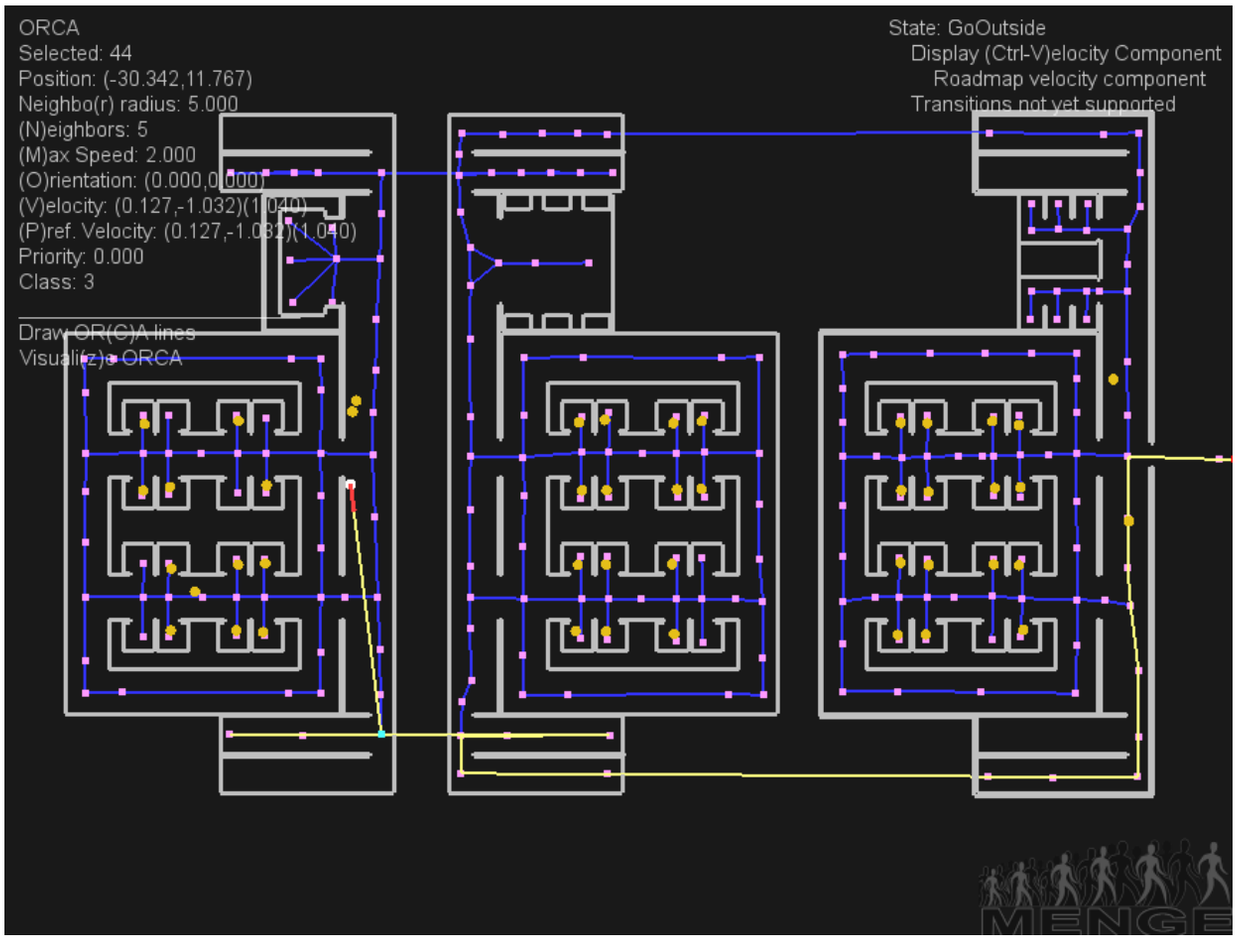


Figure 4.7 Office scene

4.6 Instantiate the scene created in Unity editor to Menge's XML files

Since one of the aim of this project is that the software implemented can be used for any kinds of uses with a low learning cost. It is impossible to learning the Menge's architecture and implement the scene and behaviour XML files. Therefore, this project provides an automatically link between the museum scene created in the software to the XML files. After the update button introduced in section 3.2.3 been pressed, the system will read the scene created in Unity editor and generate the scene and behaviour files automatically.

In order to instantiate the scene and behaviour, the scene class and also behaviour class has been created. The System.Xml namespace provided by Microsoft is the tool used to serialize the xml files. Figure 4.5 provides an example of the goal set element. A goal set contains a attribute integer id and a set of goals which is the sub-element, which means that the id is defined as a XmlAttribute class and XmlElement class for the goal set.

```
public class GoalSet
{
    [XmlAttribute("id")]
    public int id;

    [XmlElement("Goal")]
    public Goal[] goals;
}
```

```
<GoalSet id="1">
  <Goal capacity="500" id="0" type="point"
  <Goal capacity="500" id="1" type="point"
  <Goal capacity="500" id="2" type="point"
  <Goal capacity="500" id="3" type="point"
  <Goal capacity="500" id="4" type="point"
  <Goal capacity="500" id="5" type="point"
  <Goal capacity="500" id="6" type="point"
  <Goal capacity="500" id="7" type="point"
  <Goal capacity="500" id="8" type="point"
</GoalSet>
```

Figure 4.5 XML serializer example

4.6.1 Scene file

In order to recognise all of the boundaries of rooms and all the exhibits created in Unity editor, all of the wall and exhibit prefabs has been assigned a unique tag. As the figure 4.5 and figure 4.6 shows, the wall object comes with tag ‘wall’, the exhibit object is been given the tag ‘exhibit’. Therefore, when the XML needs to be updated, all of the walls (which is the obstacle in the XML file) and exhibits can be found easily by the using the tags. To be noticed that for the coordinates of walls, coordinates for four corners of rendering boundary has been used.

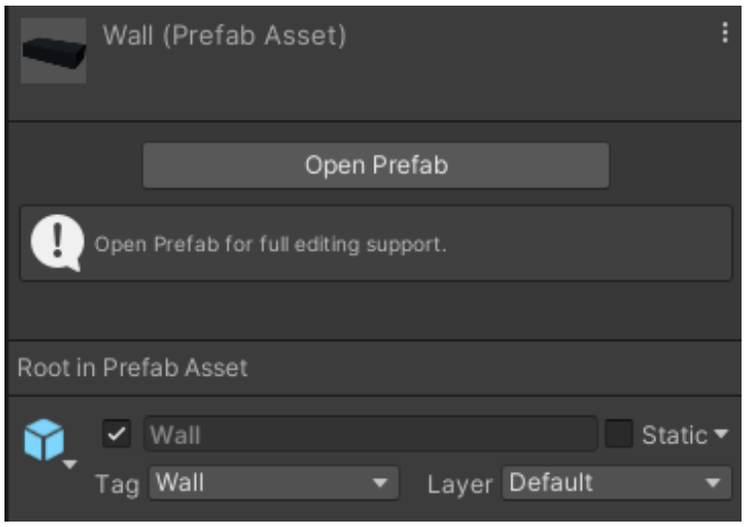


Figure 4.6 Wall prefab

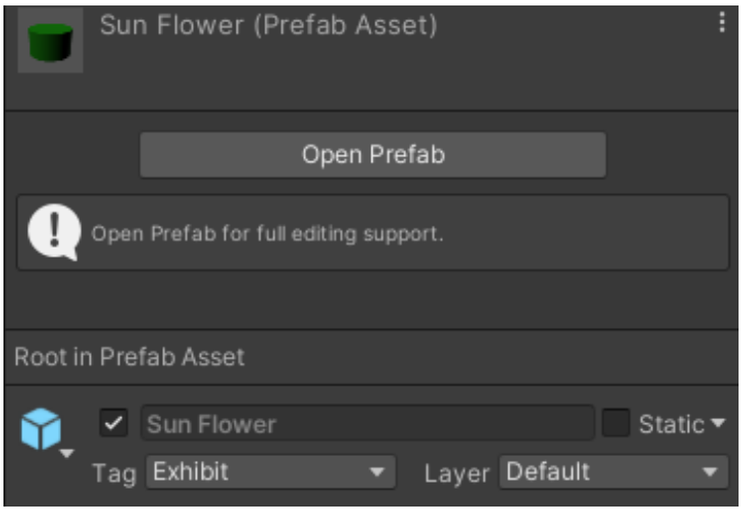


Figure 4.7 Exhibit prefab

The information of obstacles and agents is going to be passed to the save scene XML function. The function will then create an experiment object (the root of scene XML), fills in the information and writes to XML file.

4.6.2 Behaviour file

The process of saving a behaviour file is similar to the scene file. The exhibit list is passed to the save behaviour function as the exhibits goal set. There are other two goal set is created, one is the entrance of the exhibition room for the at exhibition room state and the other is the exit of the exhibition room used in at exit state.

4.7 Road map

As previous section introduced, the road map is used to provide navigation between the rooms. Road map was firstly be introduced by Latombe, Jean-Claude at 1991. A road map file defines a set of coordinates and also the transitions between the vertices. Figure 4.8 shows a simple road map used in museum model 1 introduced in chapter 5, the number 3 in first line specifies the number of vertex. And the following three rows defines entrance of the exhibition room, middle of exhibition room and the exit of exhibition room respectively. The transition between thee vertices defined by the following three lines. From vertex 0 to vertex 1 and from vertex 1 to vertex 2.

```
3
1 -21.99 0.75
2 1.59 0.75
1 1.59 -11.73
2
1 0
2 1
```

Figure 4.8 Road map for museum model 1

4.8 Debugging the XML files

As section 4.1 introduced, the Menge is a crowd simulation framework written by C++, linked with unity's script C# files by DLL files. Therefore, Unity editor will be crashed when the Menge framework process crashed and we are not able to see the log files.

In order to debugging the XML files, the GUI provided by Menge is been used outside Unity editor. In additional, not only the scene and behaviour XML file needs, but also a view XML file is declared. The view XML file defined the light and camera position in the scene so that the simulation can be rendered. The benefit of using Menge's GUI is that as long as the simulation begins, a log file will be generated for debugging.

4.9 More customized properties

- Inaccessible place: Each obstacles has a property of radius, and each goal has a reaching distance. If a large exhibit has been placed into the environment, it can be seen not only a goal, but also a obstacle. The reaching distance allows the agent reach the goal with a customized distance to the goal's coordinate.
- Staffs: The staffs with different behaviour from the visitors can be adding to the scene easily. A movement flow between the rooms is defined for the staffs.

Chapter 5
Result and Validation

5.1 Overview

In order to test the software performance, and more importantly, define the green, yellow and red for the crowdedness score, it was necessary to create several different museum scenes. The tests include different combinations of museum models, exhibit position, queue type and number of agents. Two museum models has been created, the first is a simple model with a single exhibition room, and six rooms has been created for the other room.

Some addition videos are uploaded to the YouTube, the link to YouTube video can be found in appendix A at the end of the report.

5.2 Museum models

Two museum models has been implemented. Figure 5.1 is the simple one with a single exhibition room, entrance has been placed to the top left corner and right bottom corner for the exit.

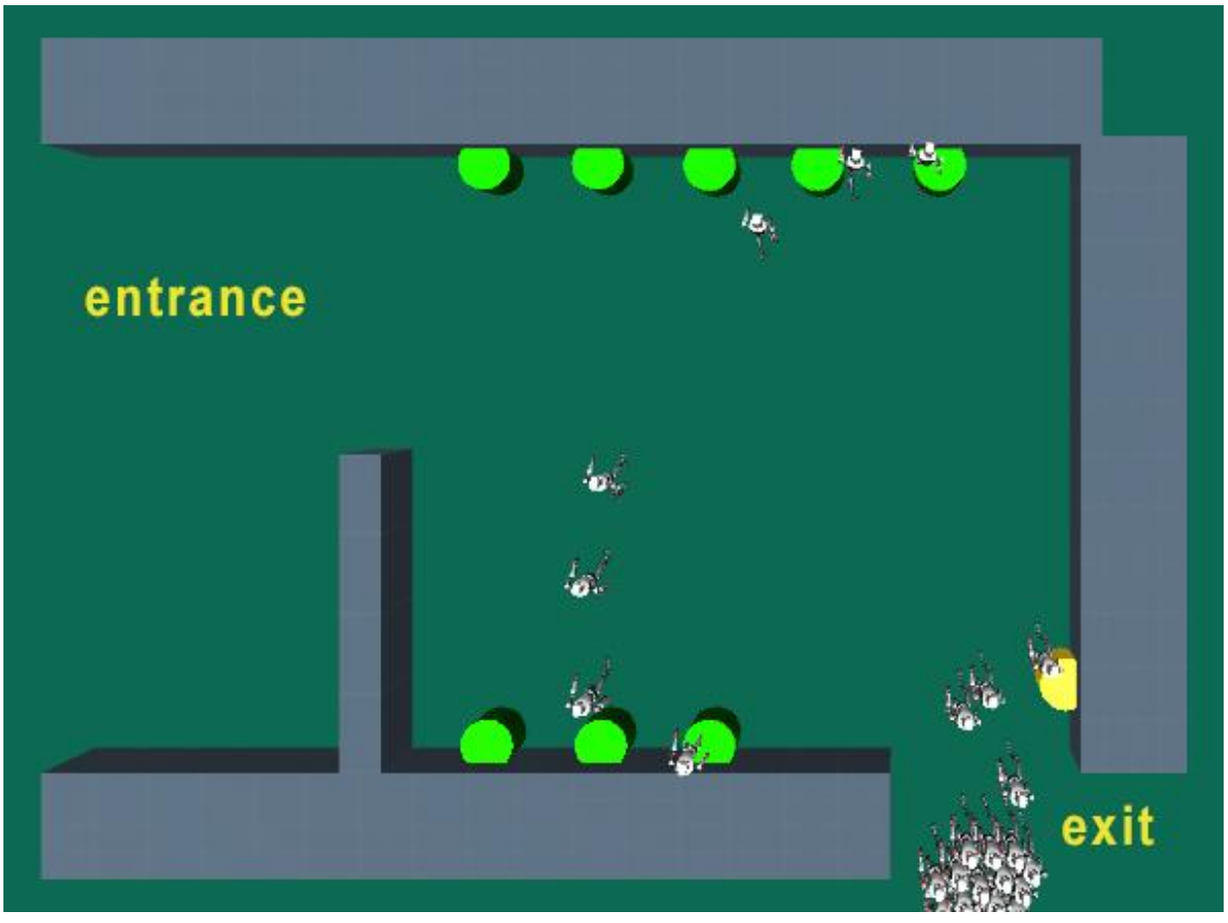


Figure 5.1 Museum model 1

Figure 5.2 is the large museum model with six exhibition rooms. Top left is the entrance and bottom left is the exit.

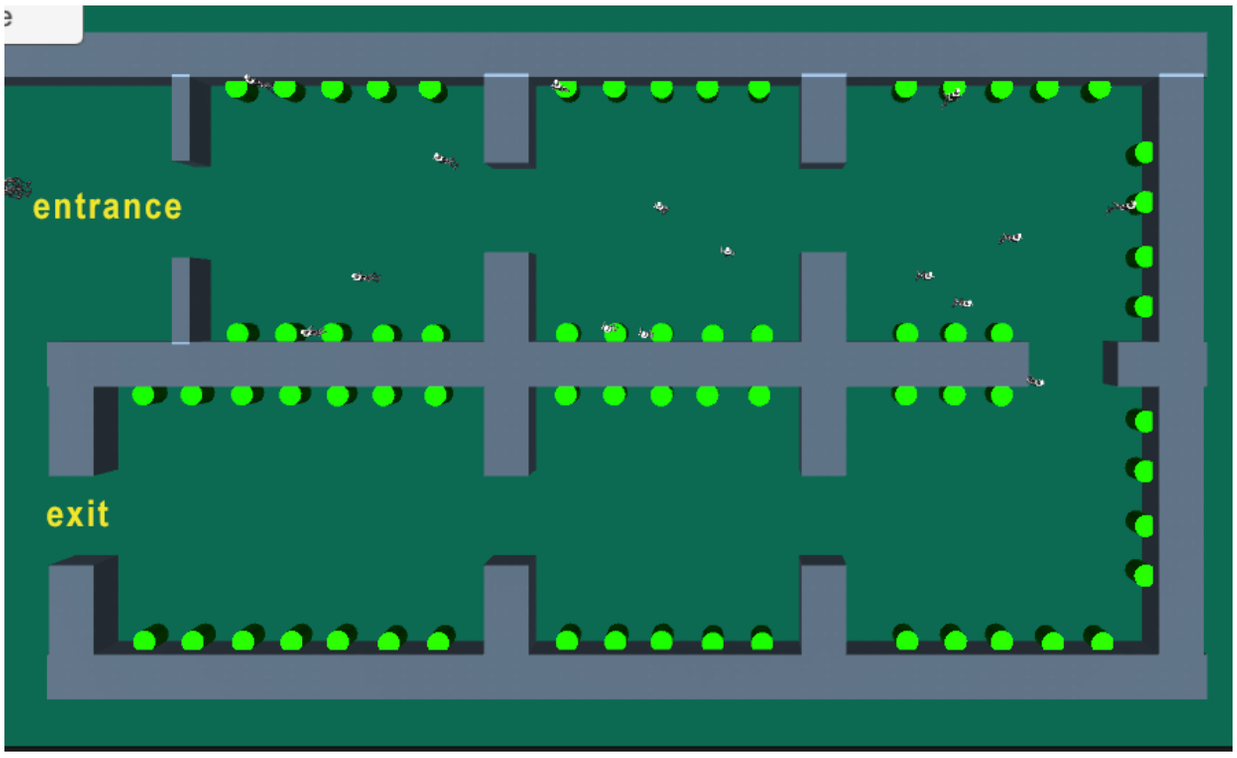


Figure 5.2 Museum model 2

5.3 Type of the queue at the entrance

Same as the museum model, two types of queue at the entrance has been create in order to compare the performance between different types of the agent group's entering flow. Two types of entering flow for 50 agents are shown in figure 5.3 and figure 5.4 respectively. The queue type 1 allows all agents entering at the same time, with almost the same spawn point.

The idea of queue type 2 comes from the reservation time slot for visiting the museums in real life. Queue type 2 split 50 agents in to several small groups with staggered entering time. The first group spawned just at the entrance of the room so they are able to enter the exhibition room just after the simulation start. While the furthest group of agents need to reach the entrance before they are able to enter the room, in order to minimized the number of agents in the exhibition room at the same time.

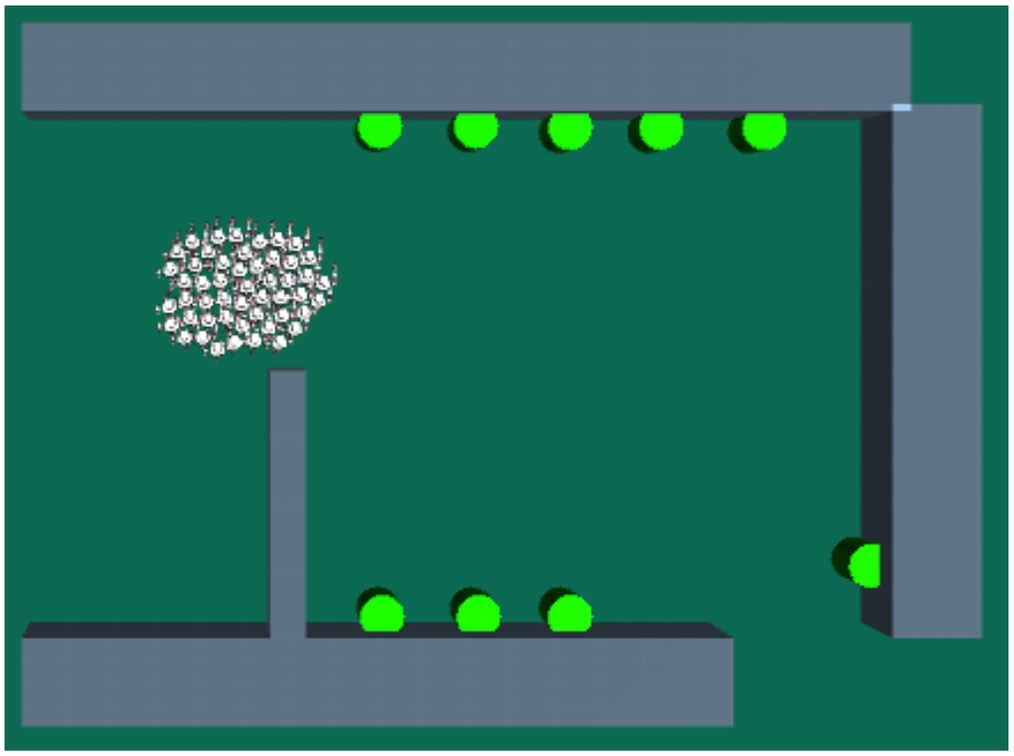


Figure 5.3 Queue type 1

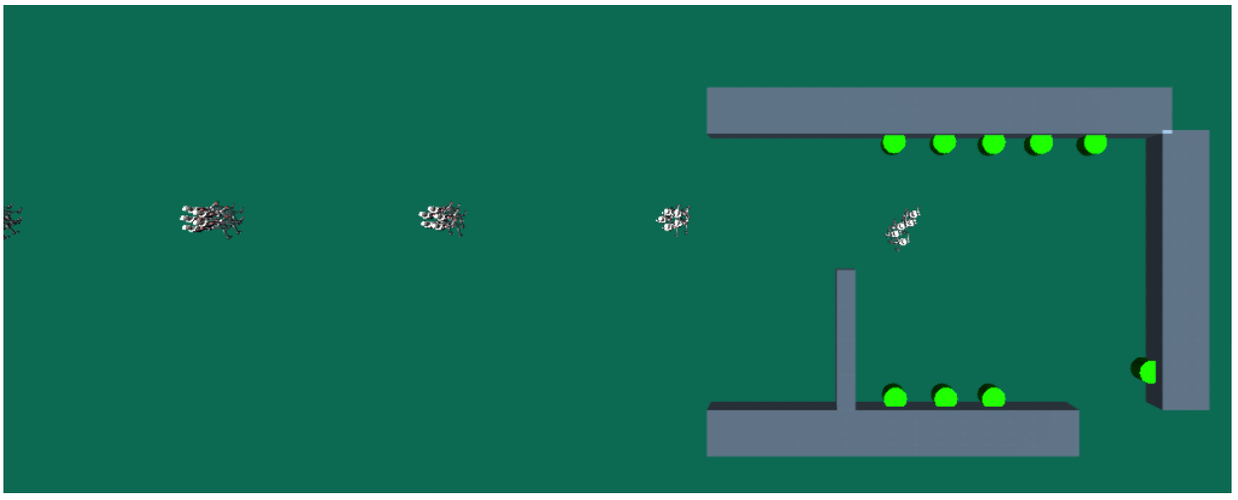


Figure 5.4 Queue type 2

5.4 Testing rules

Since the agents' targets of movements is random, different results can be shown by exactly same configuration. Therefore, all of the result scores of crowdedness are the average score of 5 or 10 simulations (depends on the simulation time and crowdedness) in same configuration.

Not all of the visitors are designed to visiting at least one exhibit in a exhibition room. Therefore, the probability of going straight ahead to room exit from the room entrance is been set to 20%. Additionally, the probability of exiting the room from visiting an exhibit has also been set to 20%, 4 of 5 people are going to continue visiting other exhibits in the same exhibition room.

In order to get the correct average crowdedness score, all of the average score was been recorded immediately after 90% of agents reached the exit.

5.5 Testing Environment

CPU: Intel Core i7-9750H

GPU: Nvidia GeForce RTX2070 Max-Q

Memory: 16GB

5.6 Single exhibition room test results

Table 5.1 shows the result of the single exhibition room model with the gallery exhibition layout (same as the layout shown in figure 5.3 and figure 5.4) and queue type 1 introduced in figure 5.3, all the exhibits are on the wall. Figure 5.5 below shows three screenshots of the moment of highest crowdedness during the simulation, with a group of 10, 20 and 30 agents respectively.

The purpose of the comparation of table 5.1 is to testing the measurement of the maximum capacity of a single exhibition room at the same time. For the result of 10 agents, the one of most highest crowdedness moment has been captured. 8 of 10 exhibits have one or zero visitors, only 2 exhibits have two visitors. Which is heathy and comfortable for this exhibition room. The average score of highest crowdedness is 5.24, which should be seen as a normal highest crowdedness score.

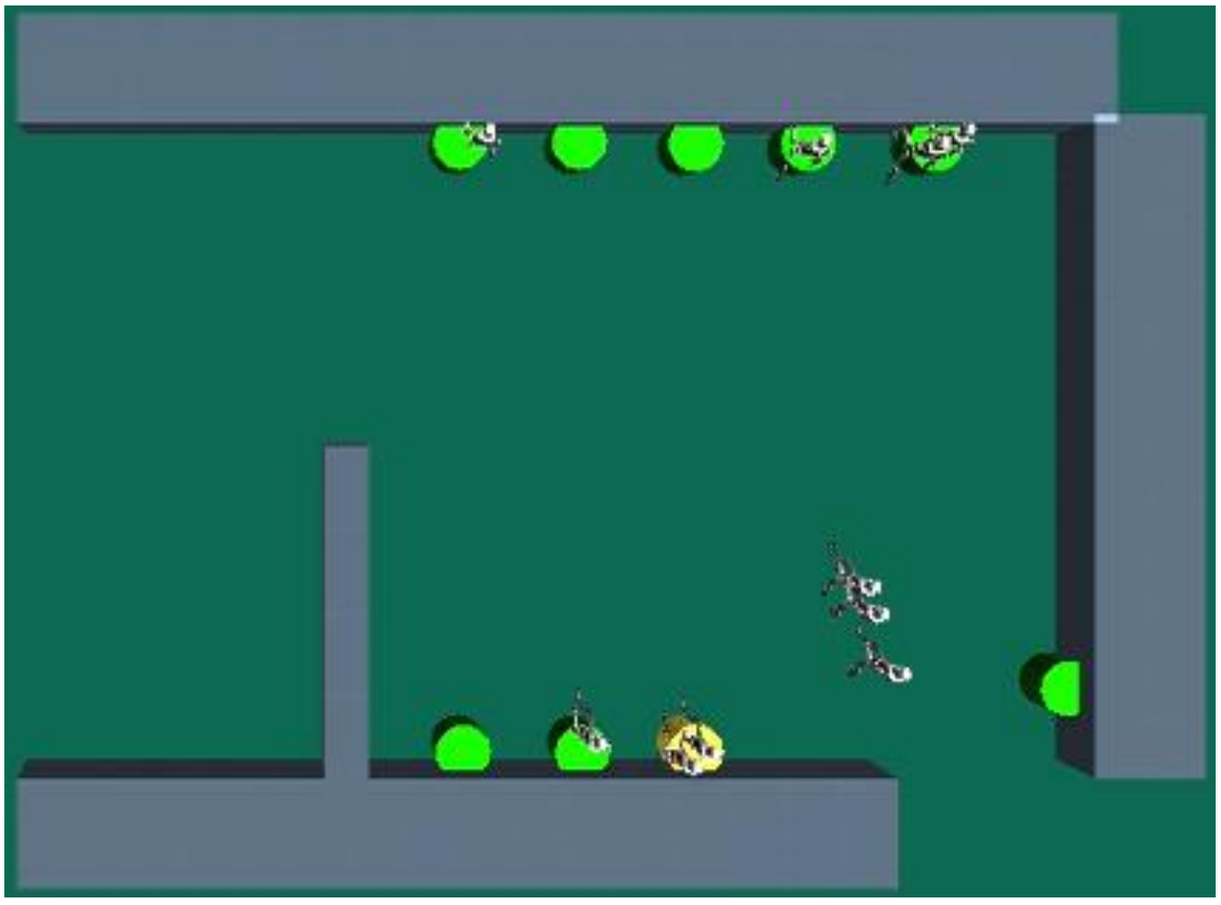
After the number of visitors has been increased to 20, the exhibition room is becoming mush more crowded than 10 visitors. Highest score 11.68 should be seen as an overcrowded score.

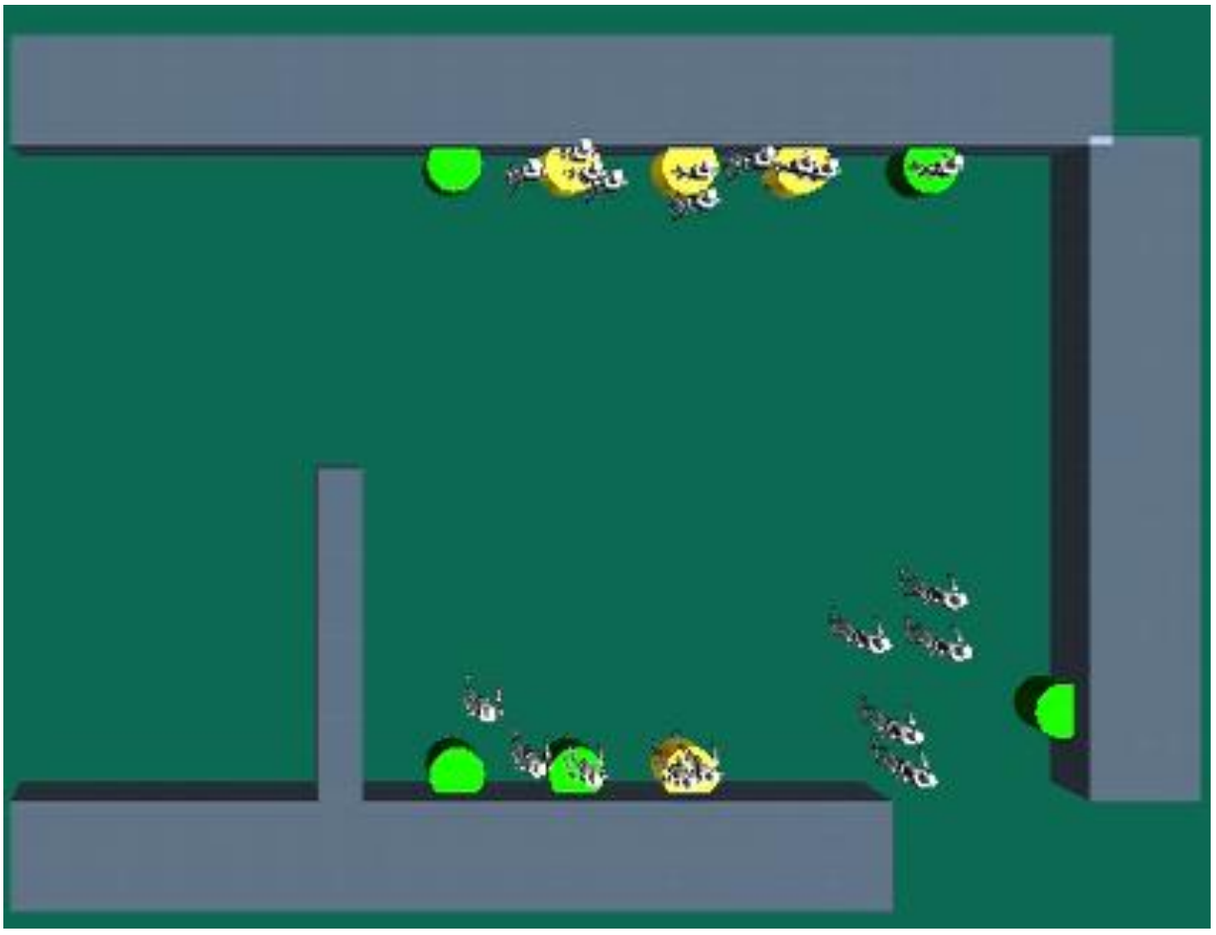
For the number of 30 agents, the highest crowdedness is unacceptable, with a score of 66.34.

To be noticed that the average crowdedness score is mush lower than the highest score. The reason of this is that there are always a few visitors that spent much more time in the room than others, therefore the simulation time has been increased, leads to a low average score. However, the most crowded moment should always be considered much more than others. Which means that the highest score should be seen as the most important score.

Queue type	Number of agents	Highest crowdedness	Average crowdedness
1	10	5.24	0.54
1	20	11.68	1.55
1	30	66.34	9.06

Table 5.1 Result for museum model 1 with gallery exhibition layout





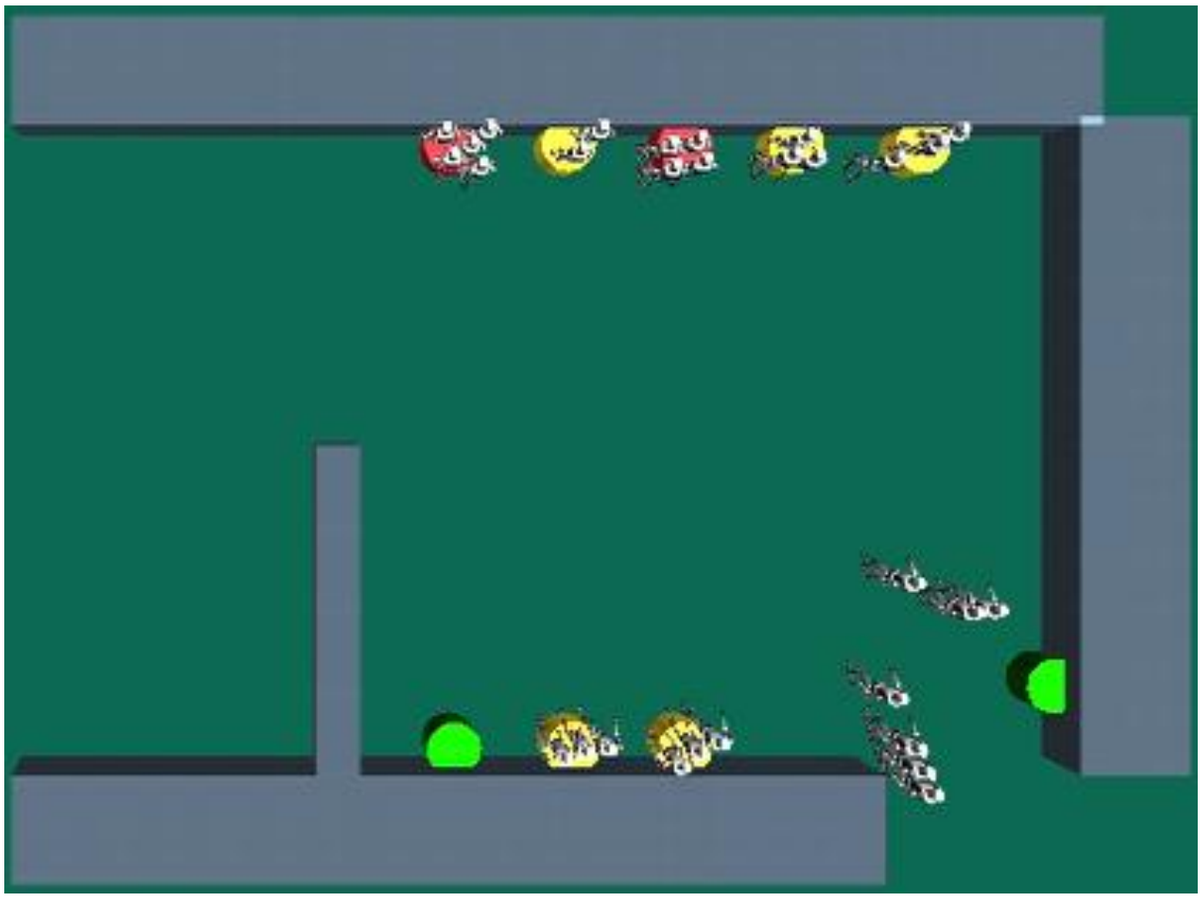


Figure 5.5 Single exhibition room test with 10, 20 and 30 agents respectively

Table 5.2 provides the result of intensive exhibition layout (figure 5.6). The purpose of the comparation between table 5.2 and 5.1 is to see the impact of exhibition layout to the crowdedness level. For the group of 10 agents, the testing shows a similar result on highest score, but a doubled average crowdedness score. This is because the simulation with an intensive exhibition layout has a smaller simulation time, visitors do not needs to travel a long distance between exhibits. The agents that are travelling between exhibits are not counted in to the crowdedness score.

This comparation proves that users are able to generate a better exhibition layout through this software.

Queue type	Number of agents	Highest crowdedness	Average crowdedness
1	10	5.38	1.16
1	20	24.68	3.12
1	30	47.54	6.60

Table 5.2 Result for museum model 1 with intensive exhibition layout

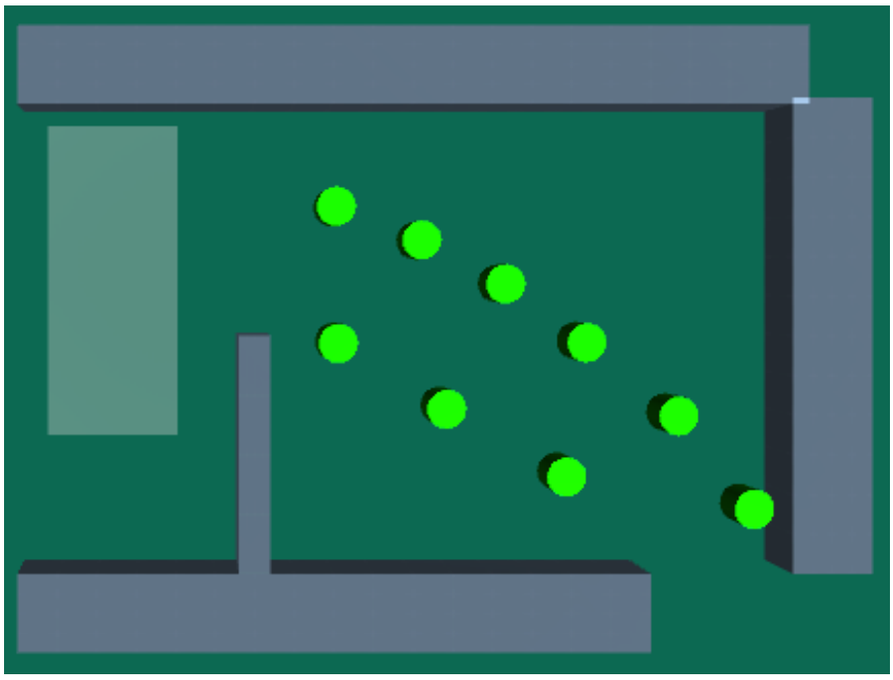


Figure 5.6 Intensive exhibition layout

Table 5.3 is the comparation between different visitors' entering flow. The queue type 2 is been set to 10 agents per group. The second group is able to entering the exhibition room after the agents in first group finish visiting the first exhibit. With queue type 2, both of the highest and average crowdedness score decreased significantly, with a slightly increasement of simulation time.

This comparation proves that users are allowed to optimizing the visitors' entering flow using this software.

Queue type	Number of agents	layout	Highest crowdedness	Average crowdedness	Simulation time
2	20	gallery	6.77	1.07	24s
1	20	gallery	11.68	1.55	20s
2	20	intensive	15.80	2.04	24s
1	20	intensive	24.68	3.12	20s

Table 5.3 Result comparation for museum model 1 with different queue type

The average fps for this evaluation is 190 frames per second.

5.7 Multiple exhibition room test results

The evaluation for the large museum model aiming at the measurement of performance and the stability. Since queue type 2 is more efficiently and also is able to holding more agents

simulating at the same time with a low crowdedness, all of the evaluation for the large museum model used the queue type 2.

The result of the six rooms museum model is shown in table 5.4. All of Interestingly, the increasement of agent group size does not affect the crowdedness score. This is because these two testing configurations has have number of agents per small group. 100 agents has been split to 10 groups, while 200 agents has been split to 20 groups. When the 11th group entering the first exhibition room, the agents in the first group has all left the first exhibition room, which means that the agents in group 11 will never staying at the same exhibit with agents from group 1.

However, the increasing number of agents brought a increasing number of crashing rate. Only the Menge's crashing is able to causing a crash of Unity. Two reason of crashing has been considered. The first possible reason is the road map's problem, which caused several crashing while implementing the software. The other reason is the Menge's bug while simulating a large number of agents. In order to find the accuracy reason, I tested this large museum model with only 20 agents 20 times, no crashed has happened. Therefore, I believe the reason of crashing is the bug of Menge.

The average framerate of 100 agents is 100 frames per second, and 90 frames per second for 200 agents.

Queue type	Number of agents	Number per group	Highest crowdedness	Average crowdedness	Crashing rate
2	100	10	8.36	1.97	50%
2	200	10	7.92	1.90	70%

Table 5.4 Result for museum model 2

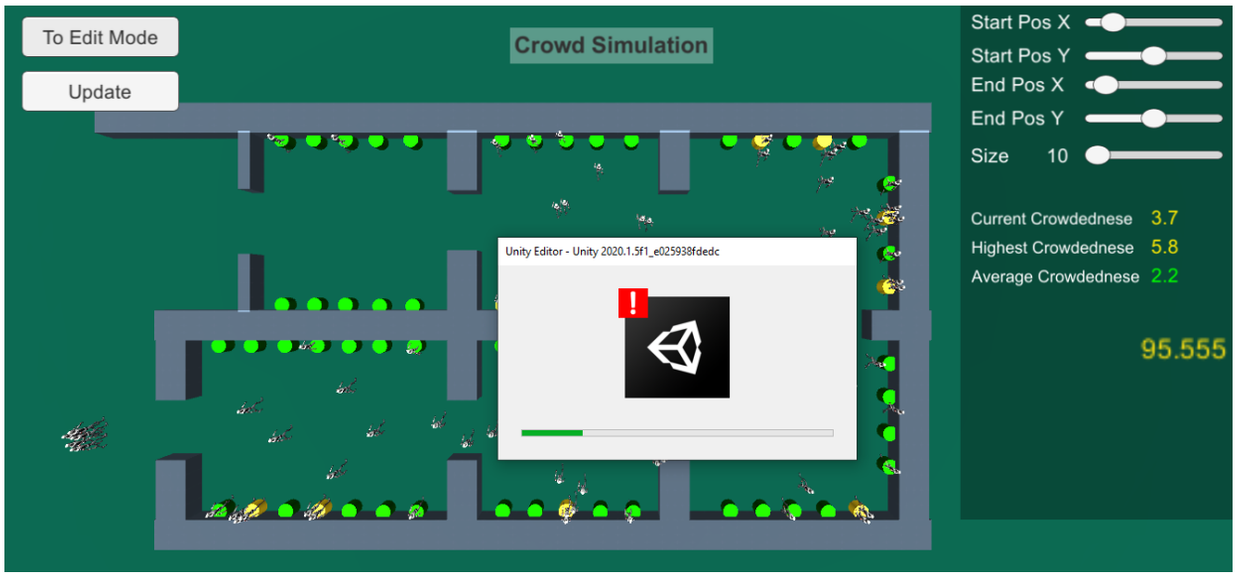


Figure 5.7 Crashing while simulating the large model

Chapter 6
Conclusion and Future Plan

6.1 Overview

This project successfully delivered a software that is able to helping museums' management. From measures the visitors' flow to the rooms and exhibits layout, this result met the designed requirements. This chapter evaluates the success of the project, as well as the future works and personal review.

6.2 Project accomplishments and shortcomings

Project accomplishments compares with other existing software:

- Customizable museum model.
- A much lower learning cost.
- A good expansibility.
- Visualized crowdedness score.

Project shortcomings:

- The software do crashing when simulation a large number of agents.
- Some of the features needs to be further implemented.
- The independent personality has been gave up due to the hardness of implementing.
- For current version of software, the model of museums can only be modified through Unity editor.
- The properties of exhibits cannot be customized.

6.3 Future works

- Most importantly, in order to free users from the Unity editor, a model modifying interface needs to be added into the software.
- Currently, some properties can only be modified by the XML files. More properties of agents and exhibits should be able to modified through the user interface in order to avoiding users reading and editing the XML files.
- Build more exhibit prefabs with different properties, some exhibits could be more attractive.
- The core code of Menge do comes with several bugs that causing a crash. Fix the bugs in Menge should provide a better user experience.

6.4 Personal review

Although the background research is done before starting the development of the software, the initial plan for this project was still too much to implementing, which leads to a bad management of implementing tasks. Some tasks do cost me a lot of times on it , but are not showing at the final result since the bad completeness of those tasks that they are more difficult and complicated than I thought.

The greatest difficulty in this project was understanding the Menge crowd simulation framework since the bad documentation. Sometimes I must look in to the source code for a specific function.

List of References

Thalmann, D. and Musse, S.R., 2012. Crowd simulation. Springer Science & Business Media.

Impact PR & Communications. 2018. The Importance of Venue Selection in Event Planning. [Online]. [Accessed 31 July 2021]. Available from: http://wwwhttps://prwithimpact.com/the-importance-of-venue-selection-in-event-planning/

Geraldine K. A. 2021. Museums stick with safety measures as England's Covid rules ease. [Online]. [Accessed 31 July 2021]. Available from: https://www.museumsassociation.org/museums-journal/news/2021/07/museums-stick-with-safety-measures-as-englands-covid-rules-ease/

Hackernoon. 2019. Insights to Agile Methodologies for Software Development. [Online]. [Accessed 20 March 2021]. Available from: https://hackernoon.com/a-case-study-type-insight-into-agile-methodologies-for-software-development-cd5932c6

GOV.UK. 2020. New guidance for reopening of museums, galleries and the heritage sector. [Online]. [Accessed 25 March 2021]. Available from: https://www.gov.uk/government/news/new-guidance-for-reopening-of-museums-galleries-and-the-heritage-sector

Network of European Museum Organisations. 2020 Survey on the impact of the COVID-19 situation on museums in Europe Final Report. [Online]. [Accessed 25 March 2021]. Available from: https://www.ne-mo.org/fileadmin/Dateien/public/NEMO_documents/NEMO_COVID19_Report_12.05.2020.pdf

RocketStock. 2017. Crowd Control: The VFX Behind Dynamic Crowd Simulations. [Online]. [Accessed 26 March 2021]. Available from: https://www.rocketstock.com/blog/crowd-control-the-vfx-behind-dynamic-crowd-simulations/

National museums. 2021. NMDC Good Practice Guidelines for Reopening Museums. [Online]. [Accessed 25 July 2021]. Available from: https://www.nationalmuseums.org.uk/coronavirus-update/nmdc-good-practice-guidelines-opening-museums/

Oasys MassMotion. 2019. Adding Proximity Modelling to your workflow with Oasys MassMotion. [Online]. [Accessed 04 April 2021]. Available from: https://www.oasys-software.com/news/proximity-modelling-massmotion/

Incontrol Simulation Software. 2019. Crowd Management for Museums. [Online]. [Accessed 15 March 2021]. Available from: https://www.incontrolsim.com/crowd-management-for-museums

Robert K.. 2008. What is Visualization? A Definition. [Online]. [Accessed 16 March 2021]. Available from: https://eagereyes.org/criticism/definition-of-visualization

Analytiks.. 2020. Why Data Visualization Is Important. [Online]. [Accessed 16 March 2021]. Available from: https://analytiks.co/importance-of-data-visualization/

Xu, M. Jiang, H. and Jin, X. 2014. Crowd Simulation and Its Applications: Recent Advances. Journal of Computer Science and Technology. Volume 29, pp.799-811.

Menge crowd simulation. 2019. Menge crowd simulation GitHub page. [Online]. [Accessed 22 March 2021]. Available from: https://github.com/MengeCrowdSim

Curtis, S. Best, A and Manocha, D. 2016. Menge: A Modular Framework for Simulating Crowd Movement. Collective Dynamics. Volume 1, pp.1-40.

Dorine C. Duives, Winnie Daamen and Serge P. Hoogendoorn. 2015. Quantification of the level of crowdedness for pedestrian movements. Physica A: Statistical Mechanics and its Applications. Volume 427, pp.162-180.

Microsoft. System.Xml Namespace. [Online]. [Accessed 10 May 2021]. Available from: https://docs.microsoft.com/en-us/dotnet/api/system.xml?view=net-5.0

Unity community. 2013. Saving and Loading Data: XmlSerializer. [Online]. [Accessed 10 May 2021]. Available from: https://docs.microsoft.com/en-us/dotnet/api/system.xml?view=net-5.0

Strohmaier, R. Sprung, G. Nischelwitzer, A and Schadenbauer, S. (2015). Using visitor-flow visualization to improve visitor experience in museums and exhibitions.

Latombe, J.C. 1001. Robot Motion Planning. Springer, Heidelberg.

Appendix A
External Materials

External tools used:

- Unity game engine 2020.1.5f1
- System.Xml Namespace
- Menge Crowd Simulation Framework
- Adobe Photoshop 2020 and Adobe Premiere Pro 2020

Extra links for this project

- Source code:

  https://leeds365-my.sharepoint.com/:f:/g/personal/sc20xh_leeds_ac_uk/Ekb9hV96TJNOrjOztcQx8YwBMg4DWSm87M_-7pgiiN4RTQ?e=jHcnsz

  password: XuchenHeHPG

- YouTube video demo link:

  https://youtu.be/5IqjARFAO_s

Appendix B
Ethical Issues Addressed

There is no ethical issues to consider in this project.