School of Computing

FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

Intermediate Report

Automatically Designed Rooms Using Machine Learning and Crowd Simulation

Edward A H Webb

Submitted in accordance with the requirements for the degree of
BSc Computer Science

2019/2020

Type of Project: Exploratory Software - ESw

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) _____

© 2019/2020 The University of Leeds and Edward A H Webb

Table of Contents

Chapter 1 - Introduction

Safety is a primary concern when architects and others design a building. Serious thought needs to be put into the availability of features like fire escapes and other exits which provide ways out for people if disaster strikes. Crowd simulation software provides insights as to how a group of people may move in certain environments and situations – such as in the event of emergency. Due to the importance of safety in design, crowd simulation is among the most prominent techniques for doing so [1]. Machine Learning is a powerful area of computing which can be used to generate results given a scoring criteria, and it's use with crowd simulation is an area we believe is relatively uncharted and worth exploring.

1.1 Aims

This project aims to follow and deliver the development of a piece of software which will attempt to combine Machine Learning and crowd simulation to automatically design floorplans which are as safe as possible in the event of emergency. The software will randomly place a dangerous event in a floorplan populated with people, at which point their escape will be modelled using crowd simulation. The floorplan will then be assessed based on how many and how quickly people escaped, whilst avoiding the danger. A 2D model will be used, as this will maintain simplicity and because a 2D simulation is sufficient and more appropriate for this project's evacuation-style simulation [1]. Machine learning will then be used to automatically repeat this experiment many times, using previous results to deliver safer floorplans to the user. The project will attempt to start with small aims, such as only changing a single object's placement in a set room before then starting to move on to multiple rooms and more complex scenarios, depending on any problems encountered.

1.2 Objectives

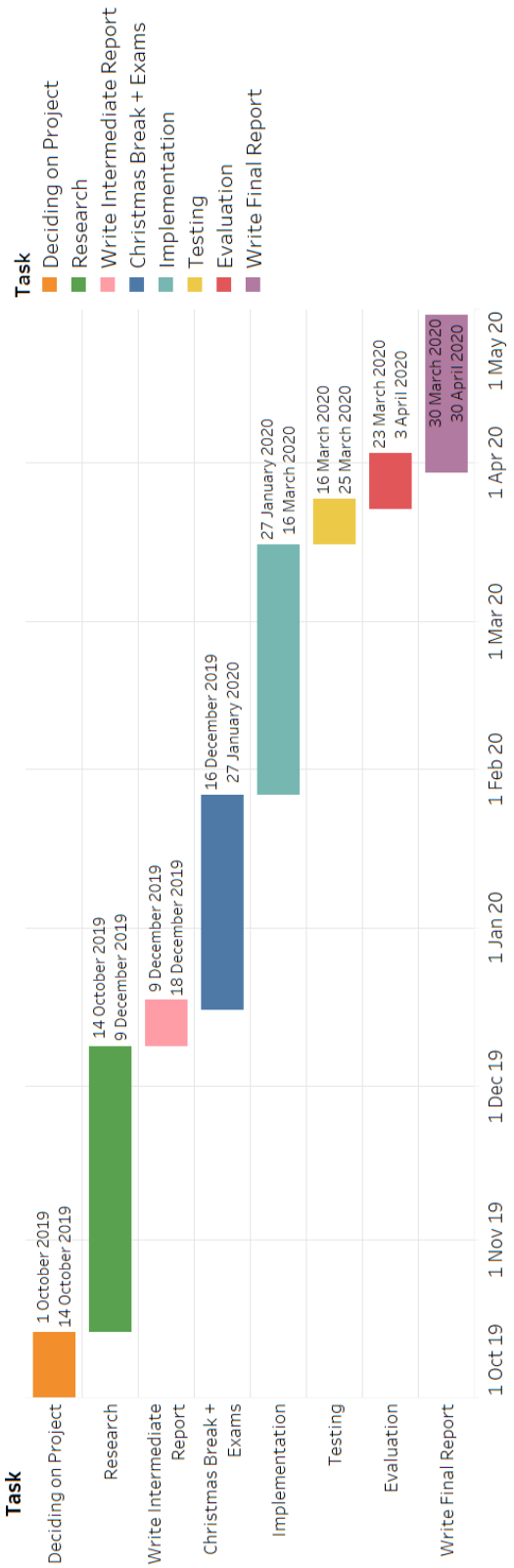
- Implement the crowd simulator using the Menge framework.
- Devise a way of placing danger in the simulation domain and scoring agents' ability to get away from it.
- Implement a way of automatically changing the floorplan to improve the safety score.

1.3 Deliverables

- Demonstration of the software.
- Report on the project.

1.4 Initial Plan

The project's panned time allocation is shown on the following page as a Gantt chart. The different areas have deliberately been left relatively vague and open ended, as it is difficult to know what will take large chunks of time in the Implementation stage for example.


Task
Deciding on Project
Research
Write Intermediate Report
Christmas Break + Exams
Implementation
Testing
Evaluation
Write Final Report
1 October 2019
14 October 2019
14 October 2019
9 December 2019
9 December 2019
18 December 2019
16 December 2019
27 January 2020
27 January 2020
16 March 2020
16 March 2020
25 March 2020
23 March 2020
3 April 2020
30 March 2020
30 April 2020
1 Oct 19
1 Nov 19
1 Dec 19
1 Jan 20
1 Feb 20
1 Mar 20
1 Apr 20
1 May 20
Date

1.5 Risk Mitigation

A key aspect of risk mitigation is guarding against the loss of work due to lack of version control. To this end, Git will be used to keep an online repository of work safe. This means that in the event of computer failure or theft, the project will be backed-up online. Furthermore the use of Git will allow the use of commit and merge points to return to in the event of faults in code; and makes developing new features easier, as branches allow for testing and prototyping before being included in the codebase.

Another important risk for this project is the general aspect of venturing into the unknown in many ways, as previous lack of experience in crowd simulation and neural networks throw up steep learning curves. Therefore there is the danger of the project's scope becoming too large, resulting in planned features having to be cut short or revised. Because of this, constant evaluation will be needed as well as advice from the project coordinator to keep the project in check and make sure it doesn't get too complicated and result in failure and disappointment.

Chapter 2 - Background Research

2.1 Crowd Simulation

When simulating a crowd there are two major ways in which to do so. One approach, under the name of macroscopic models, offers higher realism of general large scale crowd behaviour and models the crowd as more of a gas or liquid-like object [2]. The others, microscopic models, are concerned with higher detail in individual simulation of movement, but with the downside of not behaving properly in highly dense crowd situations [1][3].

As macroscopic models are more suited to evacuation and emergency simulations, this project will attempt to implement a simulation of this type.

2.1.2 Menge

Menge is a modular framework developed for crowd simulation developed at the University of North Carolina [4] and has been selected for use by the project as it allows for relatively simple manipulation of the four crowd simulation subproblems discussed below.

2.1.3 Crowd Simulation Paradigm

A common paradigm in crowd simulation is to break the problem down into four subproblems: goal selection, path computation, path adaptation, and spatial queries [3]. Each of these subproblems need to be addressed and together they make up a crowd simulator's basic structure and operation. Many simulators follow this abstraction, and there are others which differ slightly – combining the path computation and path adaptation steps for example. Menge, the crowd simulation framework selected for this project, provides ease of use when changing and using the four different subproblems [5].

The first subproblem of goal selection deals with what each agent wants to do, as they each need a goal to strive for when thinking about modelling how they'd move to fulfil that goal. In the case of this project, each agent will want to escape the danger presented to them by reaching an exit without harm.

Path computation is how the simulator works out the path that the agent should take in order to try and fulfil their goal. A complete path for the agent isn't always needed, and path computation can simply serve to compute the agent's preferred direction of travel at that moment of time. Path computation can be achieved by modelling the space as a graph and then searching it using

algorithms such as A*. Potential fields can also be used, wherein the space is subdivided into a grid and negative goals push the agent away from them, and positive goals pull the agent towards them using gradients of cost functions. This method doesn't compute a complete path for the agent, instead providing a preferred direction of travel, which is then used as input for the next computation.

Path adaptation describes how agents deal with changes to their paths on the fly. For example, an agent walking straight towards their goal would be able to steer around another agent coming the opposite way using path adaptation, instead of crashing into the and stopping completely.

Spatial queries refer to an agent's ability to recognise such things as if stimuli are in their field of view, or if stimuli are in their proximity. Effective use of spatial queries would prevent an agent from starting to run from a fire it couldn't logically see or otherwise detect.

2.1.4 Scoring the Layouts

In order to improve the layouts, each one will need to be evaluated for its safety score. Each simulation will see a certain number of agents reach their goal in a certain amount of time. These numbers will be used to calculate how many agents reach safety per second. That will be the floorplan's safety rating, with higher numbers representing a safer floorplan. The time limit will be capped to reduce experimentation time and to simulate agents succumbing to the danger. For example, a floorplan who's simulation sees 50 agents all reach safety in 20 seconds would receive a score of 50/20 = 2.5; whereas a floorplan which sees 40 of 50 agents reach safety in 60 seconds – the simulation's time limit – would receive a lower rating of 40/60 = 0.66. In this example the limit is chosen arbitrarily and will need to be more carefully chosen when implementation is taking place.

2.2 Automatically Changing the Layouts

The third and final aspect for this project will be automatically changing the floor layouts. There are many different approaches to this which have been done in the past using techniques like Bayesian Networks [6] Generative Adversarial Networks [7]. Procedurally generating them is a possible approach but has not been chosen as we would like to see the computer start to learn or appear to learn how to make floorplans more safe. It is with this in mind, then, that we look to neural networks.

Menge's architecture relies on a 'Scene' file to specify the locations of a simulations obstacles inputted as coordinates of rectangles [5]. It is these coordinates we will be attempting to change

automatically for each simulation and scoring attempt, as these obstacles' coordinates will dictate the layouts.

2.2.1 Generative Adversarial Networks

A Generative Adversarial Network (GAN) is a type of machine learning system in which two neural networks compete in a game. One of the networks, the generative network, generates new objects (such as images), which are then judged by the other network, the discriminative network. The discriminative network's goal in the game is to identify which objects are from the real data set and which have been made by the generative network, while the generative network's goal is to produce objects which fool the discriminative network, making it think the generated objects are in fact from the data set [8].

The discriminative network is trained on an existing data set, and the generative network is trained according to how well it manages to fool the discriminative one. Through the use of Backpropogation, the generative network is able to improve at generating new objects whilst the discriminator improves at catching the fake ones. In this way the generator is eventually able to manufacture objects which are extremely like those in the original data set. The power of GANs has been shown by the use of a particular GAN, StyleGAN2, which can produce incredibly realistic portraits of fake human faces which are indistinguishable from real ones [9].

Using a GAN for the floorplan generation is tempting due to their power and novel concept. A potential pitfall using it and any other kind of neural network however is the dataset it'd be trained on, which would need to be generated beforehand.

Chapter 3 - Ethics

As this project will involve software to evaluate the safety of different building layouts, ethical issues are non-trivial. It's in this project's hopes it could help develop a more sophisticated program that could be used by an architect to make a building as safe as possible. However, if a building was constructed based on a design supplied by the program and a fire caused human harm within it, serious ethical concerns could be raised. Who would be responsible should the design be found weak to fires? Some parties may claim that it was the builders, the architect, or potentially the program's author – as they may claim it could have provided a false sense of security to the architect using it.

As such, if something like this project were ever to be used by designers, legal measures may have to be put in place to make sure that no writers of the code would be liable if something like the previously described event occurred. The author of the program would likely not be accredited by the Royal Institute of British Architects (RIBA) and so shouldn't be held accountable – the onus is on the architect, engineers, and construction workers to ensure a building's safety.

List of References

[1] Jin, X., Xu, M., Jiang, H. and Deng, Z., 2016. Crowd Simulation and Its Applications: Recent Advances. [Online] Available from: https://link.springer.com/article/10.1007/s11390-014-1469-y

[2] Henderson L. The statistics of crowd fluids. Nature, 1971, 229(5284): 381-383.

[3] Funge, J., Tu, X., And Terzopoulos, D. 1999. Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In Proc. of SIGGRAPH, 29–38.

[4] Curtis, S., Best, A., and Manocha, D., 2013. Menge. [Online] Available from: http://gamma.cs.unc.edu/Menge/

[5] Curtis, S., Best, A., and Manocha, D., 2015. Menge: A Modular Framework for Simulating Crowd Movement. [Online] Available from: http://gamma.cs.unc.edu/Menge/files/MengeTechReport.pdf

[6] Merrell, P., Schkufza, E., & Koltun, V. 2010. Computer-generated residential building layouts. *ACM Transactions on Graphics (TOG)*, *29*(6), 1–12. https://doi.org/10.1145/1882261.1866203

[7] Chaillou, S. 2019. AI + Architecture | Towards a New Approach. [Online] Available from: https://towardsdatascience.com/ai-architecture-f9d78c6958e0

[8] Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014) Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.

[9] https://thispersondoesnotexist.com/