DOI: 10.2507/28th.daaam.proceedings.073

USING UNITY3D AS AN ELEVATOR SIMULATION TOOL

Jiri Polcar, Petr Horejsi, Pavel Kopecek & Muhammad Latif

This Publication has to be referred as: Polcar, J[iri]; Horejsi, P[etr]; Kopecek, P[avel] & Latif, M[uhammad] (2017). Using Unity3D as an Elevator Simulation Tool, Proceedings of the 28th DAAAM International Symposium, pp.0517-0522, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-11-2, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/28th.daaam.proceedings.073

Abstract

Simulations are used for finding answers to what-if scenarios prior to making decisions. Discrete event simulation (DES) tools are usually used for industrial processes. However, these tools require the creators of the simulation model to have highly specialized knowledge and they only rarely provide easily understandable graphic representations of the modelled situation. An elevator simulation model was created using an unusual approach: the simulation model was developed in Unity3D, an IDE intended for making computer games. The model was evaluated in terms of the accuracy of the results and of the suitability of using such a tool for creating an elevator simulation with a sequence dispatcher. The paper includes validation results and discusses the advantages, disadvantages and limits of such an approach. Although a DES tool would be better for the elevator itself, separate details of the simulation can be much more easily modelled in Unity3D. It can take into account 3D space (as opposed to 1D, which is usual for DESs), for example for finding probable paths of persons, calculating volumes or finding the right door opening times.

Keywords: 3D Visualization; Discrete event simulation; Elevator simulation; Unity3D

1. Introduction

Discrete-event simulations (DES) are a very good way to optimize many kinds of processes, but in some cases, the simplification of particular events can make the results of the whole simulation inaccurate. There are a lot of DES modelling tools, such as Anylogic, Rockwell Arena, Witness Simulation or Siemens Tecnomatix Plant Simulation. All of these tools share the concept that an entity representing a product moves through the pipeline of the simulation model, which is built of blocks or coded in a scripting language.

The specifics of such an approach are that the creators of the simulations must be trained in the use of special tools, which is usually very expensive – such tools are aimed at investments to make more profit, hence the high licence costs. Other than that, these tools are based on abstract graphics, be it 2D or 3D. Although not influencing the results, the graphics can influence the ability to understand the modelled situation and the gained results. Also, such models can only be used and run in the tool where they were created, with very limited or no possibilities to connect with other systems to use the results of the simulation. This makes the model useless for other purposes.

The lack of high definition graphics is discussed in [1]. The authors propose the use of gaming engines for presentation of the simulation because of their high optimization and very realistic graphics. Based on a survey, they claim that a good-looking model is good for its validation – especially when discussing the behaviour of the models with people who are

experts in the simulated fields. Lower importance was given to the analysis phase, but a high importance for marketing purposes. Also, if the model is to be used for training purposes, the level of realism is important. The same authors also claim that simulation packages and gaming engines have a lot in common, implicating that game engines can be used as simulation platforms [2]. Another positive aspect is that the simulation itself is computed on a CPU, whereas the graphics layer is handled by a GPU – which leaves more CPU time for the actual simulation.

The situation, however, is that usually the animation engine of the simulation tool and the computational core are closely connected together, causing the animation to slow down the progress of the simulation, which equals an increase in the costs of the simulation. 'Loose coupling' of these two systems is proposed in [3]. The authors claim that loose coupling is necessary so that the simulation can run very fast and the graphic element of the simulation can be easily modified. This approach is compatible with the previous statements, where game engines could be used as a visualization layer, or a front-end, with the simulation layer as the back-end.

Use of a gaming engine is described in [4]. A large scale traffic simulation is populated by actors with their own decisions and behaviour. The whole platform runs as a distributed simulation and is powered by a Delta3D engine. The graphics are loosely coupled with the simulation model, as proposed in [3], allowing the option of turning the graphics off to speed up the simulation. The use of a game engine has the advantage of allowing the software to be modified. For example, the described simulation model allows a user to control one car directly, which means it can be used for driver training. Also, it enables non-expert users to create what-if simulation scenarios.

Another gaming engine, the Unity3D, was used in [5] for a shopping mall simulation. The multi-agent simulation was made in this tool, probably because of the ability to write in a full featured programming language (C#) rather than scripting in a specialized language or in a block diagram. Also, the ability of the agents, who act as entities, are easier to implement with such an approach.

Physics engines included in game engines can also be used for mechanism simulations, as described in [6]. The authors even claim that development of such a simulation model in Unity3D is simpler than using traditional methods. Plus, it enhances the possibilities of interaction with the model.

2. Simulation Model Development

Unity3D is a powerful tool used in various industrial and scientific applications. As mentioned above, common tools for DES are very specialized, requiring a lot of experience and they are very expensive. That is why the goal of this research is to discuss whether it is possible to use Unity3D for creating a simulation model that would normally be made using DES tools. Basically, anybody who can code in C# can make such a model in Unity3D using its API. Other than that, Unity3D provides possibilities for enhancing the realism of the simulation or reducing the need to simplify certain processes. These possibilities include:

- Collision detection
- Calculating navigational paths
- Implementing animations

Unity3D models can be parametrized and controlled using runtime arguments, file reading or named pipes communications. Such approaches are suitable for combination with various optimization methods, as described in [7] or in [8].

For further evaluation, an elevator simulation with a simple sequence dispatcher was created for this research. Elevators have been the subject of much research on optimization and simulation, such as [9].

The question is, whether the accuracy of a simply made Unity3D simulation model can give results comparable to simulation models made using DES tools. The persons in the simulation model are self-controlled and they interact with the virtual world like a real human would. Using a regular DES tool, they would go through the model on abstract paths and usually the objects of the model would drive them. As such, modifications to the behaviour of any objects could be done directly in their class, without worrying about modifications to other objects.

The modelled elevator has an interface for collecting requests, ascending, descending, opening and closing the door. The controller is in a separate class, which makes it easy to change the instance to another kind of controller. The implemented controller is a simple sequence elevator dispatcher, collecting requests in one direction and switching to the other direction when there are no requests left in the current direction. As for the persons, after instancing it is decided whether to use the stair or the elevator based on the fitness level. If the elevator is chosen, the person comes to the elevator where he waits in the queue. The first person in the queue calls the elevator if it has not arrived. If the elevator arrives, but the person fails to board, the person calls the elevator again. After boarding, the person makes an inside request and waits until the door opens and the elevator is on the desired floor. This is the real behaviour of an elevator and a person as opposed to standard DES models, where the decision of what the person (entity) will do next is usually up to other simulation objects. An algorithm flow chart is shown in Fig 1.

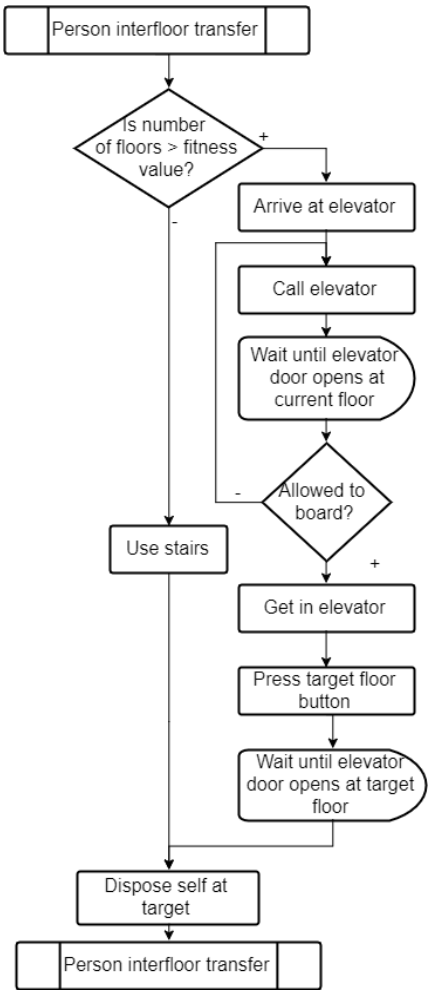

Fig. 1. Algorithm of the Person entity.

The model is designed to have as many floors as desired with a simple value changing. The floor consists of a flat 3D model, the positions of the spawn points and elevator queues. After changing the count of the floors value, the navigational mesh is computed and the model is ready to start. The model can be easily enhanced graphically, a spectator controller can be implemented or the whole model can be put into an existing scene. Also, Unity allows, after some tweaking, the persons to be controlled directly or to add some special events into the model, such as one time arrival of a lot of people and so on. The basic model can be seen in Fig 2.

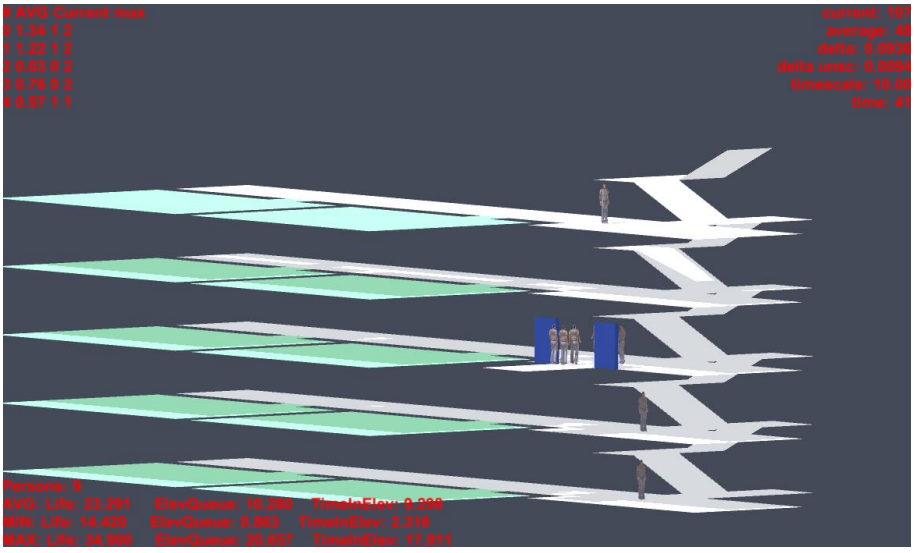

Fig. 2. Overview of the simulation model.

This approach proved to have some disadvantages as well. Because Unity3D is not prepared for discrete events, but is strictly continuous, there could be issues with increasing the simulation speed. For every rendered frame, the engine decides what will be done in the next step. The more the simulation time scale increases, the higher is the delta time, which will result for example in unnecessarily prolonging the waiting time for the lift if it arrives at the beginning of the delta time interval. This means that the maximum speed of the simulation is ordinarily more limited than the speed of a regular DES.

The value of the time scale in the Unity engine can be adjusted directly. Usually, the engine runs at 60 FPS (frames per second). The higher this value, the higher the simulation accuracy should be. The time scale directly influences how much simulation time passes between the frames. Experiments were conducted on how the combination of average FPS and time scale influences the results of the simulation.

For this case, the simulation model consisted of five floors, where each floor had an independent person source, creating one at a time at a uniform interval from 5 to 25 seconds. Every person had a uniform distribution of the target floor and disappeared after reaching the target floor, contributing to the overall statistics. The elevator interfloor travel time was 5 seconds, door opening time 1 second, and door waiting time 1 second. Simulation time was 30 minutes.

The scene was not rendered between experiments, only GUI cameras displayed the current statistics. The screen size was always 640 x 400 pixels to ensure the frame rate was always the same. The frame rates were either capped at 60 FPS using vertical synchronization or reached values around 150 FPS with V-Sync off.

The results can be seen in Fig 3:

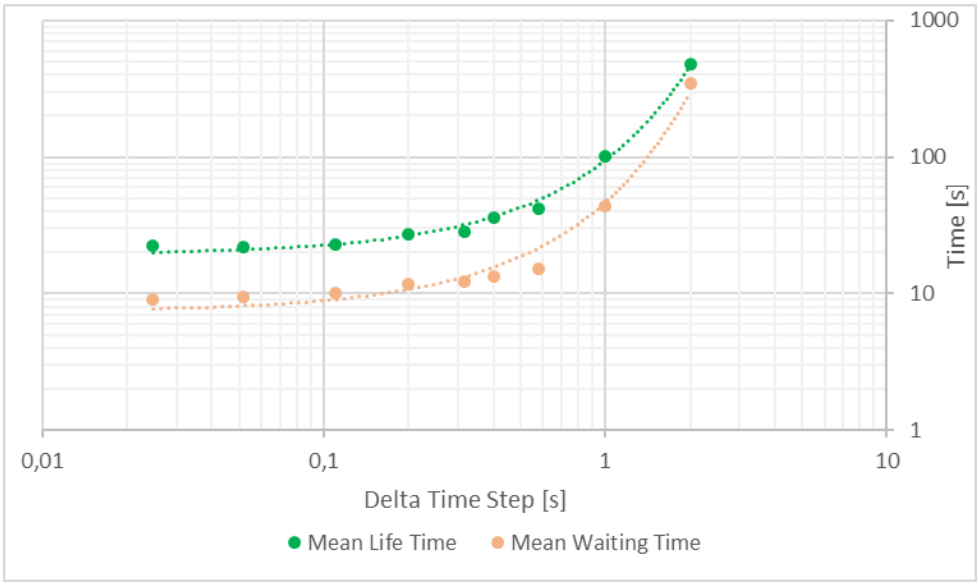

Fig. 3. Influence of the simulation time step on entities' life and waiting times.

As can be seen from Figure 3, the delta time has a big influence on the simulation results. Unfortunately, even relatively low values of time scale (which is the main factor of the delta time size), have effects on the results, meaning that the simulation cannot be sped up very much – in this case, a safe value of speeding up the simulation was less than 10 times, which is unsuitable for long term simulation runs.

In comparison with different elevator DES models, a former case study [10] was replicated in this experiment. The results are quite similar, although they are not exactly same. A model with an elevator speed between floors of 10 seconds, door opening time of 20 seconds and elevator capacity of 8 people was tested in three scenarios for 12 hours of simulation:

- Triangular distribution of arriving people of (1.8, 2.4, 2.0), (20, 60, 40) and (4, 8, 6) minutes at every storey
- Uniform distribution of target floor
- Number of storeys 6, 6 and 14 respectively

Of the published results, only the number of throughputs (number of people arrived at desired levels) was published, meaning only these values are compared. Other values in the described Unity model are also available in Table 1.

Experiment number	Described model	Case study [8]
1	2076	2421
2	116	87
3	1451	1796

Table 1. Total throughputs of elevator model described in this paper compared with the model described in [10]

3. Discussion

Making a simulation model in Unity3D which would be suitable for DES tools has both advantages and disadvantages. One of the main disadvantages is that the model cannot be sped up very significantly. This is due to the continuous computations of the objects' decisions and a too large delta time step can be very disturbing. The marginal value of the step is a function of the decision making frequency in the model. There are several ways to lower the delta time steps:

- Increasing CPU and GPU power
- Adjusting the camera to look away from complex scenery
- Lowering resolution and detail
- Reducing time scale (slowing down the simulation run)

One of the main advantages of the model is that it can easily be made to be agent based and have compliance with object oriented programming – any class can be implemented as long as it uses the model's interfaces. This means disturbing events and transitions can be simulated and the aim of the simulation can be changed quite easily. For example, if the people are not deleted on the target floor, they can stay there and move through different floors according to a stated schedule. Another thing that can be modelled is for example the capacity of the building's facilities merely by adding a decision whether to go or not to go to the toilet by evaluating some criterial functions, for example in school buildings during breaks. These are factors that can sometimes be implemented much more easily in a full-featured programming language than in DES tools. Last but not least, the model can be implemented for presentations using photorealistic graphics. There can even be multiple cameras in the model that can focus on different parts of the model, as shown in Fig 4 which can portray the same moment of the simulation as Fig 2 from different angles. Or a user controllable camera can be inserted, allowing users to see areas which they are interested in and better support their understanding of the model.

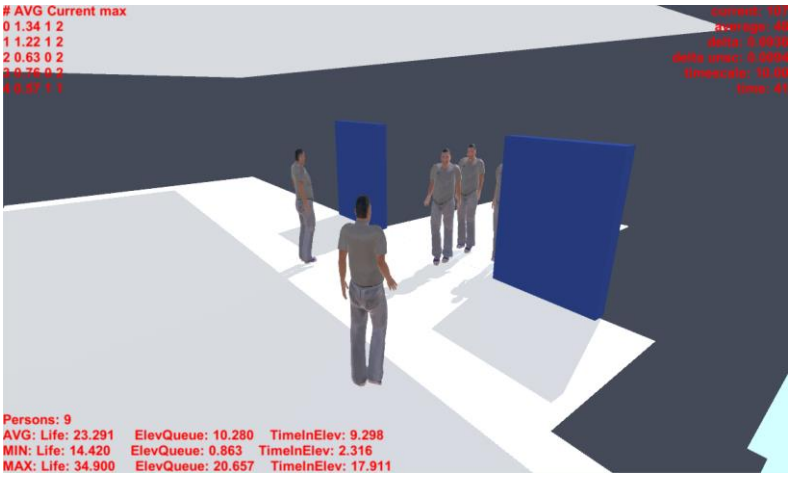

Fig. 4. Example of a custom aligned camera in the model.

Another advantage of using a completely different approach from DES tools is that the persons actually use the space to move. This allows for more complex and realistic computations when finding an optimum door opening time. If the door opening time is too short, it has to be opened again and waits for another interval to close. The walking speed of the persons can be randomly distributed, as well as the distance from the elevator door when it starts to open and the reaction time of the persons. Also, it is easy to simulate which elevator will be used if the elevators are placed in different corridors.

4. Conclusion

The basic idea was to prove whether DES simulation models need to be created strictly in DES simulation tools. For various reasons, we focused on using a different kind of tool. A simulation model was created using non-traditional tools – the Unity3D game engine and IDE. An elevator simulation model was chosen to compare using such an approach with using standard simulation tools. The development of the model proved that knowledge of C# and a basic overview of Unity3D API is enough to create such models. The tool proved to be usable for developing and evaluating such models, although, for an elevator simulation, a traditional tool would be recommended if available.

Unity3D can, however, simulate details that are usually neglected or abstracted to a very simplified level, such as boarding and exiting of the elevator with people walking at different paces and of given sizes when the elevator door has a given width. Unity3D can be used as a cheaper alternative with the possibility of using traditional programming in a full-featured programming language, whereas specialized discrete event simulation (DES) tools require a lot of special knowledge. Generally speaking, Unity3D could be used for more specialized tasks, like the behaviour of people in the system, such as which path will be chosen, and it can take into account the whole 3D space as opposed to the usual 1D paths in DES tools. Such ideas will be the subject of further research.

5. Acknowledgement

This article was prepared with the support of the Internal Science Foundation of the University of West Bohemia SGS–2015-065.

6. References

[1] Bijl, J. L. & Boer, C. A. (2011). Advanced 3D visualization for simulation using game technology, Proceedings of the 2011 Winter Simulation Conference (WSC), Phoenix, AZ, 2011, pp. 2810-2821. DOI: 10.1109/WSC.2011.6147985

[2] Bijl, J. L. & Van Nieuwenhuizen, P.R. (2009). Improving the visualization of 3D simulations using Computer Game technology, Master Thesis, Department of Computer Graphics, Faculty of Electrical Engineering, Mathematics and Computer Science, TU Delft, Netherlands.

[3] Fumarola, M.; Seck, M. & Verbraeck, A. (2010). An approach for loosely coupled discrete event simulation models and animation components, Proceedings of the 2010 Winter Simulation Conference, Baltimore, MD, 2010, pp. 2161-2170, DOI: 10.1109/WSC.2010.5678857

[4] Miao, Q.; Zhu, F.; Lv, Y.; Cheng, C. & Qiu, X. (2011). A Game-Engine-Based Platform for Modeling and Computing Artificial Transportation Systems, Proceedings of IEEE Transactions on Intelligent Transportation Systems, vol. 12, no. 2, pp. 343-353, June 2011, DOI: 10.1109/TITS.2010.2103400

[5] Christian, J. & Hansun, S. (2014). Implementation of multi-agent system using fuzzy logic towards shopping center simulation, 014 International Conference on Intelligent Autonomous Agents, Networks and Systems, Bandung, 2014, pp. 19-23, DOI: 10.1109/INAGENTSYS.2014.7005719

[6] Hu, W.; Qu, Z. & Zhang, X. (2012). A New Approach of Mechanics Simulation Based on Game Engine, 2012 Fifth International Joint Conference on Computational Sciences and Optimization, Harbin, 2012, pp. 619-622, DOI: 10.1109/CSO.2012.141

[7] Raska, P & Ulrych, Z. (2013) Testing Optimization Methods on Discrete Event Simulation Models and Testing Functions, Procedia Engineering, Volume 69, 2014, Pages 768-777, ISSN 1877-7058, DOI: 10.1016/j.proeng.2014.03.053

[8] Peschl, M.; Roening, J. & Link, N.: Distributed Simulation for Task-Driven Flexible Manufacturing Systems, Annals of DAAAM for 2012 & Proceedings of the 23rd International DAAAM Symposium, ISBN 978-3-901509-91-9, ISSN 2304-1382, pp 0171 - 0174, Editor B[ranko] Katalinic, Published by DAAAM International, Vienna, Austria, 2012

[9] Horejsi, P.; Horejsi, J.; Latif, M. & Ulrych, Z. (2011). Automatic Generator of Lift Dispatcher System Model, Annals of DAAAM for 2011 & Proceedings of the 22nd International DAAAM Symposium, 23-26th November 2011, Vienna, Austria, Volume 22, No. 1, ISSN 1726-9679, ISBN 978-3-901509-83-4, Katalinic, B. (Ed.), pp. 1217-1218, Published by DAAAM International Vienna, Vienna

[10] Horejsi, J. (2009). Use of ARENA Simulation Software for Lift Control Strategies Development in High-Level Buildings, Master Thesis, Department of Industrial Engineering and Management, Faculty of Mechanical Engineering, University of West Bohemia.