Dynamic Visualization of Pedestrian Simulation Data

Oliver Handel[1], Ege Gümüş[1], Efthymios Papoutsis[1] and Julian Amann[1]

[1]Chair of Computational Modeling and Simulation · Technical University of Munich · Arcisstr. 21 · 80333 München
oliver.handel@tum.de

Abstract

The visualization aspect of pedestrian simulation data plays an important role for the purpose of demonstrating simulation results to potential clients and, thus, has a beneficial impact in the real world. Animated 3D results are able to broaden the scope of the results and can help to understand and analyze the data. Game engines such as *Unity3D* or the *Unreal Engine* are powerful tools to visualize data generated by pedestrian flow simulators. This paper documents the results of a software lab project in which an existing post-visualization software for pedestrian simulation data is improved and extended with new features. The software is based on the Unity3D game engine.

1 Introduction

On the one hand, simulation of pedestrian flows can help to evaluate safety issues at different event locations in order to avoid crow disasters. The incident of the 2010 Love Parade, which ended up fatal for 21 people and left 510 more injured (HELBING & MUKERJI, 2012), is one motivation of this paper. On the other hand, these simulations can also help to tackle management issues by optimizing and evaluating layouts of event locations. Furthermore, simulations will be vital in the design process of buildings, if the planner is interested in quantities such as occurring pedestrian density or expectable evacuation time. In the hands of a designer or an engineer, tools to demonstrate simulation results in a realistic fashion are a great opportunity, allowing them to easily overview a certain situation with ease and take actions accordingly (SHAO, 2006). In other words, being able to show 2D results is good, but visualized 3D results are a lot better to make a simulation more realistic. Moreover, many scholars have stressed the value of game engines in scientific research (for example FRITSCH & KADA, 2004; HUDLICKA, 2009; LEWIS & JACOBSON, 2002).

In this project, we combine powerful simulation tools such as *Anylogic*, in which the results of the simulation are written into a plain text file (describing the positions of the pedestrians for every time step, the exact size, position and rotation of all geometrical objects existing in the scene, etc.). In connection with the Unity game engine, the text-based simulation output is then transformed into a 3D-post-visualized scene. In this paper, our post processing software *PedVis 3D* and its capabilities are going to be explained in detail. The software is a derivative of an open source software called *SumoViz 3D* (BÜCHELE, 2014b). The main challenges of this project are some debugging tasks, the implementation of new major features such as measuring density with a heat map, the ability to run the software on a web-browser and enabling to import geometries from Building Information Software such as Autodesk Revit.

2 Software Platforms

The first prerequisite to be able to visualize a scene, an event or any movement of pedestrians, is of course to run a simulation. The aforementioned Anylogic software is an adequate tool to simulate movements of pedestrians.

2.1 Anylogic

Anylogic is a simulation tool that supports discrete event, agent-based and system dynamics simulation (BORSHCHEV & FILIPPOV, 2004) It is successfully used in domains such as business process modelling, logistics and supply chains, manufacturing, healthcare and pharmaceutical simulation and, as in in the actual case, for the simulation of pedestrian movements. The developers of Anylogic stress three major issues in respect of pedestrian dynamics that are important to consider and can be faced with this software (ANYLOGIC COMPANY, 2015): 1. Time and throughput calculation: Focuses on minimizing congestions and waiting times. 2. Pedestrian traffic impact analysis: Serves to investigate what impact static and dynamic obstacles in a space have on the travel time of pedestrians. 3.Evacuation analysis: Seeks to define the optimal evacuation procedure in emergency situations.

Apart from these issues, there are also other possible purposes – such as, for example, assessing infrastructures, sales and consumption at different vendor stands over time as well as density measurements. To be able to 3D-visualize a scene, the trajectories of the pedestrians and the spatial positions of all the geometries of the environment are of fundamental importance, which is why they are required from Anylogic. For this reason, this data is exported from Anylogic in the form of interface .txt files, which are used as input files for PedViz. In summary, the purpose of AnyLogic is to execute the simulation and to deliver the required input data.

2.2 Unity 3D

Need For Speed, *Tiger Woods*, *Oddworld* and *Temple Run* are a few examples of games that are based on Unity3D. But there are also a lot of non-game related applications, such as *InMind VR*, *Virtual Car 2.0* or *3D Fleet App* that are based on Unity3D (UNITY TECHNOLOGIES, 2015). Apart from Unity 3D, there are many more game engines available for commercial and educational purposes as well as for personal and professional use. The most famous ones are the *Unreal Engine*, *CryEngine*, *Source Engine*, *Blender*, and o. The reason why Unity is chosen over the others is its availability in up to 22 different platforms such as Windows, Android, iOS and the embeddability into web browsers. Unity3D currently supports the two different scripting languages C# and UnityScript. The latter one is also known as JavaScript for Unity.

Game engines like Unity provide developers with many core functionalities such as renderers, physics tools and compilers. So, instead of reinventing the wheel over and over again, developers can use game engines to create games and other applications based on the existing solutions. For this reason, it is essential to ensure interoperability between Anylogic and Unity – which can be achieved by scripting to enable data to be passed on from one program to the other.

3 From SumoViz 3D to PedVis 3D

The developed software that is described in this paper is called *PedVis 3D*. It is a derivative of the open source software SumoViz 3D, which was developed by Daniel Büchele as part of his master thesis and published under the MIT license (BÜCHELE, 2014a).

3.1 Scope of the Project

SumoViz 3D features the basic functionalities to visualize pedestrians in a generic defined scene. However, since this software was developed under the time constraints of a master thesis project, it has several limitations. Thus, the software does not offer the potential user a lot of freedom to define and evaluate a scene. When we started working on this project, our aim was to improve the software and to bring it to the next level by making it accessible for other scholars in such a way that they are able to upload their simulated pedestrian data and visualize their data via a browser. Our new features work towards these intentions. It started off with a few GUI changes for the sake of user-friendliness, followed by major feature add-ons that will be explained later on. The development of PedViz started in February 2014 under Unity 4.0 and is currently being continued in Unity 5.0.

3.2 Program Logic

Since we don't intend to use Unity 3D in the conventional manner, the largest part of the software is coded directly in C# rather than creating an actual scene by using Unity3D's own GUI. The software uses Microsoft's .NET Framework, which is currently in version 4.6. In combination, NET Framework's class libraries and Unity3D result in a powerful platform.

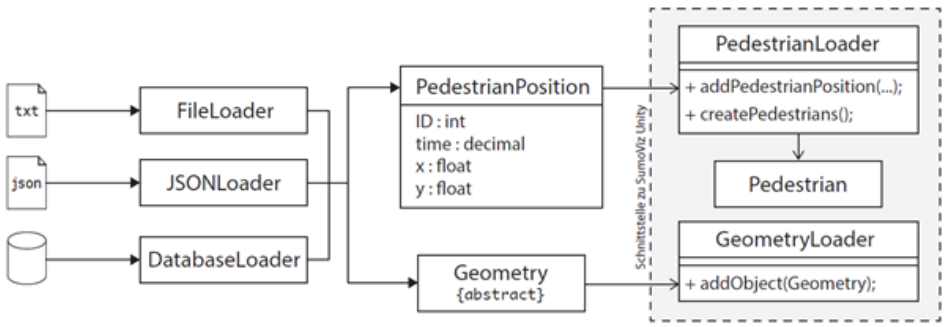

Figure 1: Creation of a scene by using different inputs. (Source: BÜCHELE, 2014b)

Figure 1 shows the logical structure of the program. The software is able to process different input types. If textual input files are used, at least one file containing the pedestrian positions and one containing the geometry is needed. The information of the positions is stored in a list of *PedestrianPosition* objects that are later used by the class *PedestrianLoader* to instantiate the pedestrians into the scene. At the same time, the latter geometry input is parsed by the script *FileLoader*, and the class *GeometryLoader* is called to instantiate all objects into the scene. The area marked with dashed dots results in a complete SumoViz scene.

3.3 Data Input Specification

With Anylogic, it is possible to view the simulation in 2D and, at the same time, export the data of the simulation into different external files to enable a post-visualization with PedVis3D. In other words, Anylogic enters the simulated data into textual files while the simulations is executed – and once this is done, the files are fed into PedViz in order to 3D-visualize the same scene (Figure 2).

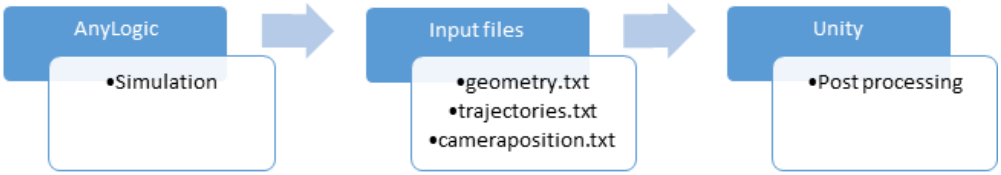

Figure 2: Data flow from simulation to post processing.

As already mentioned, at least two input files should be provided in order to visualize a scene: a trajectory file and a geometry file. The trajectory file contains the positions of all pedestrians for every time step, and the geometry file consists of all the geometrical elements of the scene – such as walls, houses, trees or other objects. In a third file, users can optionally define different camera positions and their viewing angles, so it is possible to switch between different perspectives easily. The format of the geometry and the pedestrian input file is defined as follows. Figure 3 shows an exemplary set of data for both files.

Geometry: Object ID objectName RotationClockwise ScalingFactor x y z
Trajectory: TimeStepOfSimulation PedestrianID x y z

```
1 Object 5 backtothewoodsign 225.0 1.0 111.0 -101.0 0.0
2 Object 12 HiFimusicsystem 263.0 1.0 193.5 -56.0 0.0
3 Object 21 whiteindianhouse 0.0 1.0 99.5 -50.1 0.0
4 Object 11 greenshop 100.0 1.0 62.2 -91.9 0.0
5 Object 29 yellowseats 7.0 1.0 67.5 -92.9 0.0
```

```
1 1.04 0 40.23 40.05 0.00
2 1.04 1 40.69 37.64 0.00
3 1.04 2 40.92 36.44 0.00
4 1.04 3 40.92 38.85 0.00
5 1.04 4 40.69 35.24 0.00
```

Figure 3: Geometry input file on the left and trajectory input file on the right.

3.4 Graphical User Interface of PedVis 3D

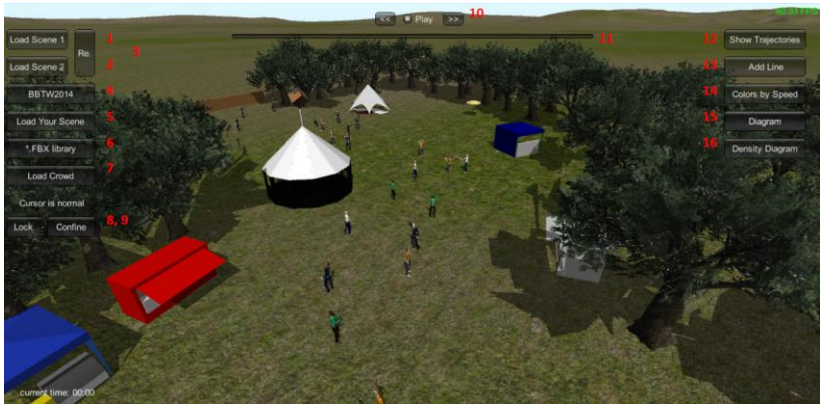

Figure 4: Graphical User Interface of the current version.

Figure 4 and Table 1 provide an overview of how the user can interact with the software. As this project is ongoing, the final look of the user interface may differ slightly.

Table 1: Explanation of the elements of the user interface.

Element	Name	Explanation
1	Load Scene 1	Loads the first preloaded demoscene. In this scene, a few dozen pedestrians are walking from point A to point B.
2	Load Scene 2	Loads the second preloaded demoscene. This scene visualizes the simulation of a festival with the name Back to the Woods held in 2014
3	Re. (Reset)	Resets the whole scene by deleting all objects.
4	BTTW 2014	Loads the 3D objects for the second demoscene. Visualizes the Autodesk Revit Integration.
5	Load Your Scene	Opens up two windows, enabling the user to visualize any simulation scene. The required format is explained in the previous section.
6	FBX Library	Opens up the PedVis library with the predefined geometry objects. The user can add various geometries to build an individual environment.
7	Load Crowd	Instantiates a set of an animated crowd in the scene.
8	Lock	Mouse cursor control. Locks the camera and sets the cursor active.
9	Confine	Mouse cursor control. Locks the camera and the cursor.
10	Slower/Faster	Increases and decreases the speed of the simulation.
11	Time Slider	Shows the current moment in time of the visualization.
12	Show Trajectories	Displays the trajectory for each pedestrian currently available in the scene.
13	Add Line	User can draw a red line to count pedestrians passing this line and to determine the current flow rate.
14	Colors by Speed, Colors by Density	Colorizes each pedestrian based on the actual speed. Another click shows the current density for each pedestrians based on the number of pedestrians around.
15	Diagram	With this button, it is possible generate a fundamental diagram.
16	Density Diag.	Turns on the density diagram. This feature is explained later.

4 Features of PedVis 3D

In this section, some of the implemented features of the PedVis 3D software are explained in detail.

4.1 Autodesk Revit Integration

One of the main goals for the amelioration of the software was the ability to import scenery models from Autodesk Revit to make the scenery more realistic. Because Revit is an advanced software tool to model 3D-objects, the possibility to import objects from this software platform is very helpful. Some difficulties had to be tackled to allow Revit objects to be integrated into Unity. The process flow in Figure 5 shows our solution to get Revit objects into Unity.

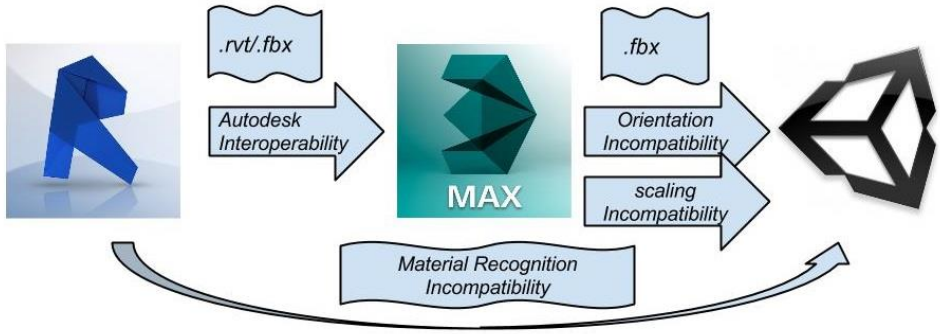

Figure 5: Process Flow Revit-Unity-integration.

As Unity is incompatible with Autodesk's material library, a direct integration from Revit to Unity is not possible without using additional software. With the software 3ds Max from Autodesk, it is possible to solve this incompatibility, as 3ds Max standard material type uses bitmaps and not procedural or mixed maps. In 3ds Max, it is furthermore possible to change the misalignment of the local x-axis by rotating the objects 90° around this axis. As Unity is not primarily a geometry modelling software, it has some limitations in comparison to the powerful 3ds Max software. Firstly, Unity does not recognize NURBS surfaces. This means that tessellation has to take place where the NURBS surface is partitioned in polygons. Secondly, every object can have just up to 65536 (= 2^{16}) vertices, which is due to the fact that Unity uses a 16 bit index buffer. Hence, bigger objects have to be separated into smaller objects.

Figure 6: Back to the Woods 2014 geometry loaded at run-time.

All objects from the Revit model were exported and converted with 3ds one by one to create a library of reusable objects in the fbx file format that are importable into Unity. Fbx

(abbreviation for *filmbox*) is a proprietary file format owned by Autodesk, developed to ensure interoperability between digital content creation applications. These fbx objects can then be easily instantiated in Unity, allowing the user to create different geometries easily. As an example of an environment based on a Revit model, Figure 6 depicts the scenery of the *Back to the Woods* festival in Unity3D.

4.2 Web Browser Integration

As mentioned in the second chapter, one of the major reasons why Unity3D was chosen over other game engines is its cross-platform compatibility. The Unity web player is available for free and is suitable to view 3D content created with Unity directly in all common browsers. We successfully tested whether the web player is supported by the Internet Explorer, by Firefox, Chrome, Safari and Opera on both Windows and Mac OS X.

One of the most important features in PedViz 3D is available via the *Load Your Scene* button. It allows users to load their simulated content directly via the Internet simply by a copy-and-paste command of textual data. To use this feature, users need to export their simulation data in the required format as described in the third chapter. Then, they select the whole text within their simulation data files and copy and paste it into the popup boxes. Once both text files are loaded, the user is able to visualize and analyze the simulation data and use all features of PedVis, without the need to install any further software on their local machine. A reset button serves to delete all the objects that are currently active in the scene, and to create a new blank scene in which new objects can be loaded.

4.3 Density Map

The density map allows users to display a live heat map in which the ground floor is colorized based on the amount of pedestrians that are currently located in the different quadratic cells. While the color red indicates the most crowded areas, the color green displays less crowded areas in the scene (see Figure 7).

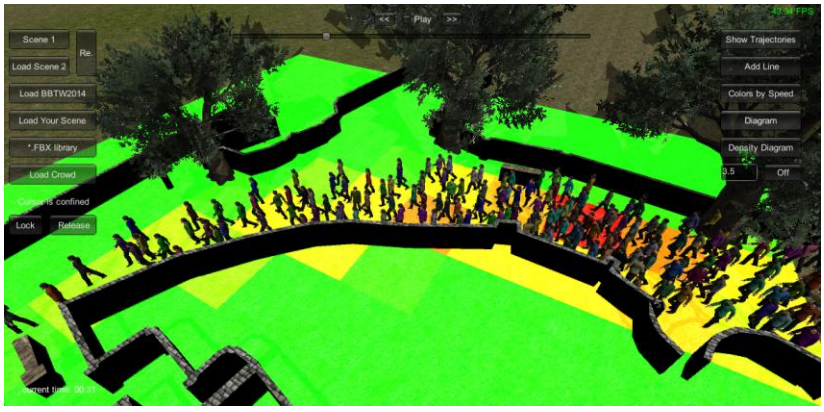

Figure 7: Demonstration of the live density map.

This feature allows users to observe areas in which high densities are to be expected in reality. For the purpose of visualizing an emergency evacuation, the user can study problematic corridors and can evaluate specific changes – first in the simulation and then, accordingly, in the real world – in order to avoid high density conditions.

The calculations are based on the amount of pedestrians currently present within the borders of each mesh cell. For each time frame, the number of pedestrians is determined and then normalized based on the total number of pedestrians. This normalized value is converted into the specific color code and then displayed on the ground floor. Once the density diagram button is pressed, the display shows a button to turn it off again – as well as an option to alter the mesh size. The default value for the mesh size is 3.5, but the user is able to change this value as well. For scenes larger than our first demoscene, a larger mesh size is recommended.

5. Conclusion and Outlook

As a result of this project, an improved visualization software based on a derivative of an existing software for pedestrian simulation data has been provided. PedVis 3D introduces new features for its users – such as the integration of 3D objects, the possibility to run the software via the Internet and to visualize user-generated simulation data, plus the possibility to analyze the data based on, for instance, density assessment. As the software is compatible with web browsers, users are able to work on their simulations from a distance, while the integration of 3rd party geometry objects from Autodesk Revit allows users to recreate their own events.

The software enables scholars to visualize their own simulation data based on the power of the state-of-the-art Unity game engine. Furthermore, the software is useful for the purpose of presenting results to potential clients, such as urban event managers, in a smarter way. This will hopefully increase the chances that critical key issues will be recognized and countermeasures will be taken – having a beneficial impact in the real world.

At the time of the publication of this paper, the software does not yet include all the features that are going to be available in the final version. For the final version, it is currently planned to implement additional features such as: 1. a possibility to visualize any pedestrian-specific variables compiled in the simulation (e.g. agent-based internal states), 2. a dynamic visualization of ground plane polygons to visualize, for example, the space requirements of individuals with Voronoi polygons and 3. a possibility to export the data from the collected statistics such as, for example, the fundamental diagram data.

Acknowledgement

This research was funded by the German Federal Ministry of Education and Research as part of the program Research for Civil Protection (disclosure Urban Safety). The authors want to thank A. Mayer and D. Biedermann for the allowance to use the *Back to the Woods Revit model* in the visualization.

References

ANYLOGIC COMPANY. (2015). Anylogic Multimethod Simulation Software. Retrieved July 6, 2015, from http://www.anylogic.com/consulting/pedestrian-traffic-flows

BORSHCHEV, A., & FILIPPOV, A. (2004). From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools.

BÜCHELE, D. (2014a). SumoViz3D at GitHub. Retrieved June 1, 2015, from https://github.com/danielbuechele/SumoViz3D

BÜCHELE, D. (2014b). *Visualisierung von Fußgängersimulationsdaten auf Basis einer 3D-Game-Engine*. Technische Universität München.

FRITSCH, D., & KADA, M. (2004). Visualisation using game engines. *Archiwum ISPRS*, *35*, B5.

HELBING, D., & MUKERJI, P. (2012). Crowd disasters as systemic failures: analysis of the Love Parade disaster. *EPJ Data Science*, *1*(1), 7. http://doi.org/10.1140/epjds7

HUDLICKA, E. (2009). Affective game engines: motivation and requirements. In *Proceedings of the 4th international conference on foundations of digital games* (pp. 299–306). ACM.

LEWIS, M., & JACOBSON, J. (2002). Game engines. *Communications of the ACM*, *45*(1), 27.

SHAO, W. (2006). *Animating autonomous pedestrians*. DTIC Document.

UNITY TECHNOLOGIES. (2015). Made with Unity. Retrieved July 12, 2015, from http://unity3d.com/showcase/gallery/