EG UK Computer Graphics & Visual Computing (2016)
Cagatay Turkay and Tao R. Wan (Editors)

PED: Pedestrian Environment Designer

James McIlveen, Steve Maddock, Peter Heywood and Paul Richmond

Department of Computer Science, University of Sheffield, Sheffield, UK

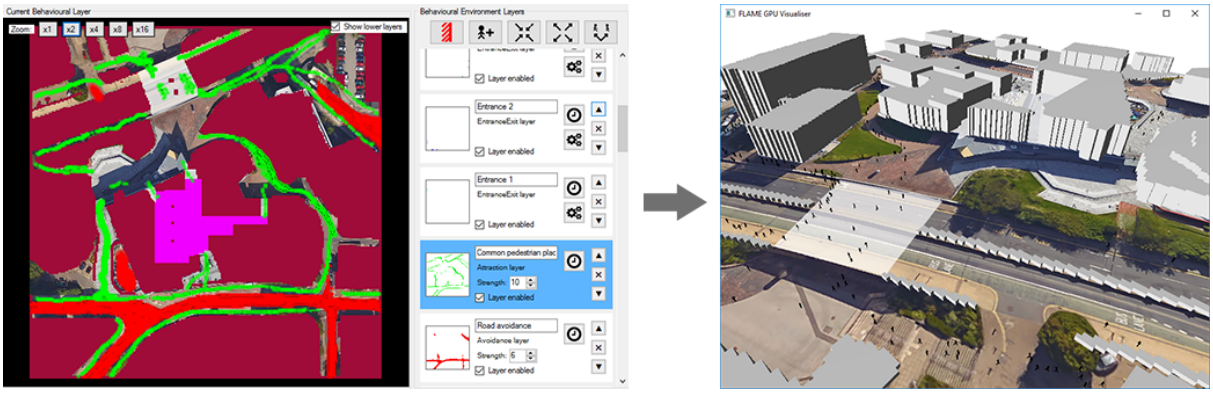

Figure 1: *Environment created in the PED interface being simulated.*

Abstract

Pedestrian simulations have many uses, from pedestrian planning for architecture design through to games and entertainment. However, it is still challenging to efficiently author such simulations, especially for non-technical users. Direct pedestrian control is usually laborious, and, while indirect, environment-level control is often faster, it currently lacks the necessary tools to create complex environments easily and without extensive prior technical knowledge. This paper describes an indirect, environment-level control system in which pedestrians' behaviour can be specified efficiently and then interactively tuned. With the Pedestrian Environment Designer (PED) interface, authors can define environments using tools similar to those found in raster graphics editing software such as Photoshop™. Users paint on two-dimensional bitmap layers to control the behaviour of pedestrians in a three-dimensional simulation. The layers are then compiled to produce a live, agent-based pedestrian simulation using the FLAME GPU framework. Entrances and exits can be inserted, collision boundaries defined, and areas of attraction and avoidance added. The system also offers dynamic simulation updates at runtime giving immediate author feedback and enabling authors to simulate scenarios with dynamic elements such as barriers, or dynamic circumstances such as temporary areas of avoidance. As a result, authors are able to create complex crowd simulations more effectively and with minimal training.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer graphics]: Methodology and Techniques—Interaction Techniques, I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation, H.5.2 [Information interfaces and presentation]: User Interfaces—Graphical user interfaces (GUI)

1. Introduction

Pedestrian simulations find many uses, from games and films through to evacuation simulations and disaster planning programs. As these simulations become more advanced, with more complex environments and greater numbers of pedestrians and behaviours, efficient and effective authoring approaches are required to create, control and direct the simulations. Our focus is on creating a system where non-technical authors can do this. For pedestrian simulation, we use an agent-based approach combined with force-vector fields [Rey99], both supported within the FLAME GPU framework [RR11, fla]. Agent-based pedestrians infer their behaviour from a defined environment, based on a behaviour model. Our Pedestrian

DOI: 10.2312/cgvc.20161304

Environment Designer (PED) is an intuitive tool for the creation of these environments.

PED uses layers of two-dimensional bitmaps, each representing different aspects of behaviour within a three-dimensional environment. In analogy to raster graphics editing software such as Photoshop™, a user paints entrances and exits, areas of attraction and avoidance, and collision boundaries onto layers. Another layer might contain a reference image, e.g. a map of a train station, which can be used as a guide for the information on subsequent layers. In the resulting simulation, pedestrians (agents) use the information in the layers to control their behaviour in the environment. Complex simulations can be easily created with an approach that doesn't require any prior technical knowledge of crowd simulations. In addition, using PED, authors can dynamically update the simulation environment, producing a trial and error approach for simulation authoring, e.g. a barrier can be added or removed, the attraction of an area increased or the flow of pedestrians varied.

In order to demonstrate the capabilities of PED, two sets of results will be presented. First, we demonstrate the ease with which a range of complex environments and associated simulations can be created. Second, we present the results of an experiment in which an environment was created by users with no prior technical knowledge of crowd simulations, thus demonstrating the intuitive nature of PED.

Section 2 will first cover the related work in the field. Section 3 will then cover PED, including the features it offers and how they have been implemented. This is followed by the results and discussion in Section 4 and then the conclusions 5.

2. Related Work

Two separate levels of control can be identified in pedestrian simulations, micro and macro, although both are often used in conjunction. Micro control defines rules that affect individual pedestrians, and parameters are set on a per pedestrian basis. In contrast, macro control is concerned with parametrising behaviours shared by many pedestrians.

Micro-level control can be seen as a local model of a pedestrian's behaviour, with an agent-based approach perhaps being the most popular. Reynolds' [Rey87] work on modelling the interaction in flocks and herds used micro-level controls. Helbing then adapted Reynold's ideas to crowd simulations to produce the Social-Force model [HM95], Paris used velocity-space analysis to resolve the problem of pedestrian avoidance [PPD07], and Guy proposed the PLEdestrian model [GCC*10] which attempts to minimise metabolic work by pedestrians using the 'Path of Least Effort'.

In contrast, macro level controls can affect all pedestrians at once or particular subsets of pedestrians based on pedestrians' states. Macro level controls are often used in conjunction with micro controls in order to provide global simulation path finding and configuration. Examples include Reynolds' Force Vector Fields (FVFs) [Rey99], where an FVF is a matrix of vectors which represent directional velocities that define the motion of agents within a specific area, Chenney's Flow Tiles [Che04], which allow users to create FVFs by combining smaller tiles of reusable FVFs, and Banerjee's work [BAK08] on layering of FVFs in order to simplify the creation of complex behaviours. Another macro level approach is the use of continuum theory. Here, Hughes [Hug02] defines dynamic potential fields that can be used for global navigation and local collision avoidance. This was then improved by Treuille [TCP06] to produce continuum crowds by tuning Hughes' model with empirical data.

While micro and macro controls parametrise the features of both the pedestrians and their environment, they can interact with each other in two different ways: either directly, or by inference (indirectly). All of the work covered so far in this section is used in systems that utilise inferred control. Inferred pedestrian control does not require that pedestrian trajectories are specified, but instead requires a set of rules to be defined that pedestrians use to calculate their actions based on their state, the state of other pedestrians and their environment. Pedestrian movements are a product of the simulation's current state and the environment. Simulations of this nature require an environment to be created in order to produce the desired behaviours.

In contrast, direct pedestrian control is where micro and macro controls manipulate the pedestrians in a fully authoritarian manner. All pedestrian trajectories and actions are completely parametrised. This form of control allows a simulation author to produce fine-tuned simulations as all control is explicit, but at the cost of the required time to define this fidelity. Both Kwon [KLLT08] and Kim [KSKL14] showed that once basic pedestrian trajectories had been made, it was possible to perform motion editing on these trajectories. Takahashi showed that it is possible to create pedestrian trajectories that collectively transform pedestrians' positions from one group formation to another using a spectral-based approach [TYK*09]. Yersin proposed Crowd Patches [YMPT09] a system that defines a volume containing several pedestrian trajectories in 3D space. These patches can be combined together to create a larger grid of patches and pedestrians can traverse trajectories spanning several patches. Jordao built upon the idea of Crowd Patches with Crowd Sculpting [JPCC14], a system which proposes a Crowd Patch graph system where patches can be arranged and distorted to create specified scenarios.

Finally, we focus on tools used for authoring simulations. Some of the aforementioned studies also present simulation creation tools alongside their control methodologies. Kim created a tool [KSKL14] which allows for interactive manipulation of pedestrian trajectories and Jordao's Crowd Sculpting system [JPCC14] showed this sort of interactive control was also possible with Crowd Patches. Ulicny created Crowd Brush [UCT04] which allows simulation authors to use a brush tool to create and remove pedestrians by clicking within an environment. The brush tool also allows authors to individually or collectively set properties and behaviours of pedestrians by selecting them with the brush tool. In contrast to Crowd Brush, Agent Paint [MR05] used image mapping as a way to create behavioural environments instead of agent properties. Behavioural traits within a 3D environment can be specified by painting different colours onto a bitmap which is then mapped to a 3D plane.

Our system, PED, uses the paint concept for environment behaviour that can also be seen in Agent Paint [MR05], although

Agent Paint does not split behaviours into separate layers, nor does it include functionality to dynamically update the simulation at run time to provide authors with the feedback that is necessary to tune the behaviours within the environment. PED can also be likened to Crowd Brush [UCT04], in that brush-like tools are used to influence behaviours within the environment. The main difference is that Crowd Brush directly affects pedestrians, whereas PED affects the environment of the pedestrians.

3. The System

PED focuses on enabling authors to create environments that pedestrians can infer their movement from within a simulation. Authors can create environments on a single plane by creating multiple bitmap layers that represent behaviours. There are different types of layers that an author can add, each affecting the pedestrians behaviour in different ways. These layers can be painted using tools similar to those found in common raster image editing software.

Once an author has finished creating their environment, the environment can be compiled and loaded into the pedestrian simulation program written using the FLAME GPU framework. Pedestrians can be seen to move around the environment in real-time. An author can then make changes to the environment if desired, and the environment can then be dynamically transferred to the running simulation. PED is based on the ideas of layered FVFs [Rey99,BAK08], painted behaviours [MR05] and the Social-Force model [HM95].

3.1. Interface

PED's interface (fig. 2) is split into three distinct sections: the Environment Workspace (bottom left), the Layer Viewer (bottom right) and the Toolbox (top). The Environment Workspace shows the current state of the environment, and is where the tools are used. All of the currently visible layers can be seen here, and can be zoomed and translated to allow authors to see the part of the environment they are currently working on. The Layer Viewer shows all of the layers within the current environment, and allows for the creation of new ones. A preview for each layer is displayed, and users can rename, reorder and configure the layers from this menu. The layer which is currently being edited is highlighted in blue and can be changed by clicking on another layer. The Environment Workspace only shows the current layer and those below it so that the view of the topmost layer is not obscured. The Toolbox allows the user to select and configure the current tool. The menu bar at the very top of the interface can be used for environment management (new, save, open), environment compilation and simulation execution.

3.2. FLAME GPU Pedestrian Simulation

Environments that are created in PED are run using a modified version of the pedestrian simulation created by Karmakharm [KRR10] using the FLAME GPU framework [fla]. This system represents pedestrians using agents that are controlled by the Social-Force model where FVFs provide global pedestrian navigation and obstacle avoidance. Pedestrians enter environments at a specified location and the make their way to prescribed exits. Separate FVFs are used to navigate agents to each of the available exits, one FVF per exit, irrespective of entrance, and another is used for global collision avoidance . PED is specifically tailored to creating environments that use this methodology for crowd simulation.

Figure 2: *PED's interface, highlighting the Environment Workspace and Layer Viewer.*

3.3. Layers

A layer in PED consists of a bitmap and configuration for that layer. Each layer represents a different behaviour within the environment and the collection of layers represents the environment (Figures 3 and 4). Each of the layers is only effective where the user has painted the bitmap with colour. All layers can be configured through the Layer Viewer panel and all layers can be enabled and disabled to allow for easy editing of environments. The following list describes the six types of behavioural layer available to PED authors.

- **Entrance/Exit** layers define where pedestrians spawn and exit the environment. When a pedestrian enters the scene, an exit (on any entrance/exit layer) is chosen at random for the pedestrian to head towards, and eventually exit the environment from. A different layer is used for each unique exit and each layer is displayed in a unique colour. Emission rates can be configured and exit probabilities can be defined for each pair of layers (entrance and exit) so that realistic pedestrian flows can be imitated. Painting on this layer is discrete; pixel opacity is binary.
- **Collision** layers define where pedestrians are able to move within the environment. Painted areas block pedestrians from passing, and unpainted areas are walkable. Collision layers have a configurable height value. This height does not effect the obstructive nature of the layer, but is used to produce visualisations where the collision area is swept upwards to create simple 3D models within the resultant simulation to provide author feedback. While many collision layers can be created, all layers are combined during compilation so multiple layers are only used for organisation purposes and to define different heights in different areas. Collision layers are displayed in crimson within the Environment Workspace.
- **Areas of Attraction** layers are used to define a model of pedestrian distribution. Painting on these layers (using different levels of paint opacity) attracts pedestrians to move through the painted areas while moving towards their exit. Areas of Attraction have a strength value which can be configured to make the area more or less attractive to pedestrians. While Areas of Attraction can affect the path a pedestrian takes, the pedestrian is still guaranteed to eventually reach their exit if there is a possible path. Areas of Attraction are displayed in green.
- **Areas of Avoidance** layers are like Areas of Attraction, but instead of attracting pedestrians to move through them, they discourage movement through their specified areas. Similarly, Areas of Avoidance also have a configurable strength value to adjust their influence. Areas of Avoidance are displayed in red.
- **Areas of Interest** layers enable authors to create waypoints within environments. Normally pedestrians move from their point of emission to their exit, but pedestrians can switch to moving towards an Area of Interest on their way to their exit. Probabilities can be defined that specify a pedestrian's chances of switching to navigating towards an Area of Interest and back again. Areas of Interest are displayed in fuchsia and currently only one such layer can be added.
- **Reference** layers are the only layers that do not influence behaviour within the system. They allow the author to load a reference image, e.g. a map of a railway station, into their environment that can then be used as a guide for environment creation. Reference layers are also visualised within final simulations to provide spacial context.

3.4. Layer Editing Tools

To edit layers within PED, users are provided with three tools: the Block tool, the Brush tool and the Eraser tool. All three of these tools can be selected as the current tool and configured from within the Toolbox at the top of the system interface (fig. 2). Once selected a tool can be used to manipulate the current layer within the Environment Workspace.

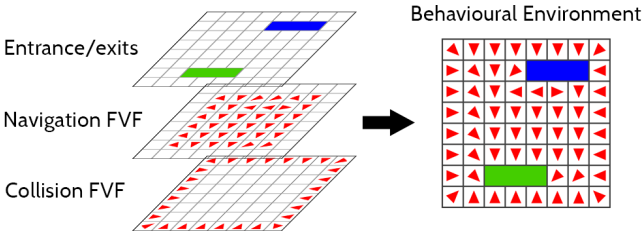

Figure 3: *Layered FVFs within the pedestrian simulation program.*

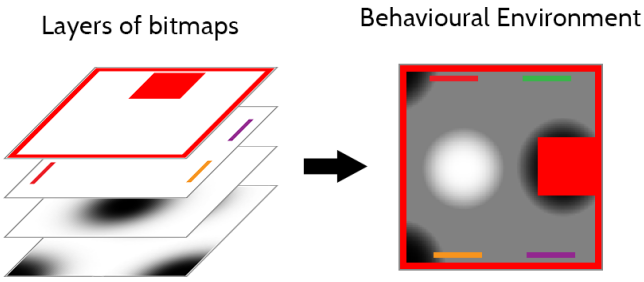

Figure 4: *Layered behaviours represented as bitmaps.*

All tools are similar to those found in common raster graphics software packages. The Block tool allows the user to click and drag to create rectangle shapes. The brush tool paints within a circle of influence wherever a user clicks or drags. The Eraser tool removes painted behaviours within layers wherever it is dragged. The brush and eraser tool have settings to change properties such as their size and step, but the block tool does not.

3.5. Environment Compilation

After a collection of layers has been produced, a compilation process is used that converts the layers into an XML model input format for FLAME GPU. PED represents behavioural environments using bitmaps, and the pedestrian simulation requires environments to be represented using a set of agents known as navmap agents. Navmap agents represent the entire environment and are organised into a grid. They specify: their coordinates within the environment, whether or not they are an entrance/exit and also vectors for each of the navigation, collision and Area of Interest FVFs for that position.

Navigation FVFs guide the pedestrians to their respective exit. They are calculated for each exit in the environment so pedestrians can navigate towards them in the simulation. To calculate each Navigation FVF, an iterative Djikstra floodfill algorithm is used. All collision layer bitmaps are flattened and converted into a 2D boolean array that defines collision areas. All Entrance/Exit layers are also converted into a boolean array like this as specified by their respective bitmaps. The algorithm then uses these arrays to create a distance map, which marks the distance from every cell in the environment to the current exit (fig. 5). The resultant distance map is then used to create FVF vectors that guide pedestrians to the exit

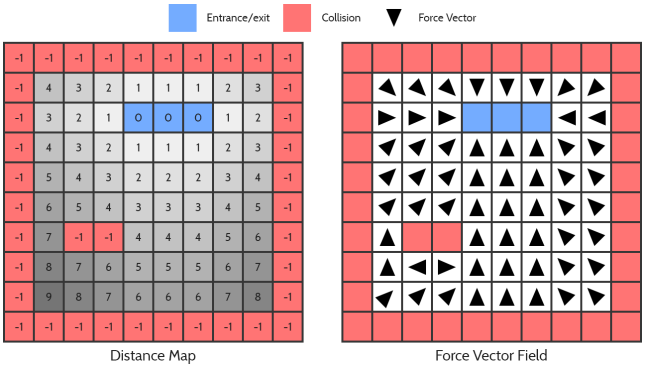

Figure 5: *Distance map and FVF creation.*

(fig. 5). If a cell is part of the collision area, or exit area, or was not reachable by the flood fill algorithm, it is not given a vector.

To create the Collision FVF used by the simulation, a similar process to the one used to create the Navigation FVF is used. All space that is not marked as a collision area is given a distance of 0 when creating the distance map and the environment is then flood-filled from those cells. The resultant Collision FVF will then always direct pedestrians outwards from the collision zone to the nearest non-collision area.

Area of Avoidance layers are accounted for by converting all Area of Avoidance layer bitmaps into a single cost mask that is used as an overlay when calculating the distance map for each exit. The alpha values for the pixels in each layer are multiplied by the layer strength and combined to create the cost mask. The result of this is a weighted array of cost values that represents all of the layers of avoidance. When calculating navigation layers, distance values are augmented by the corresponding cost value. The result is that the navigation FVFs avoid guiding pedestrians through avoidance areas where possible, but the strength of the deterrent is determined by the strength of the layer.

Area of Attraction layers are similar to Area of Avoidance layers. The only difference is that attraction layers add a cost for wherever the layer has not been painted instead of where it has been painted. The change in the algorithm is simple – the transparency value is inverted. The produced cost mask then works in the same way as the one produced by the Area of Avoidance layers.

Areas of Interest also require their own navigation FVF layers. During the simulation, pedestrians that are currently moving towards an Area of Attraction use this FVF to guide them to the area. These navigation layers are calculated in the same way that the entrance/exit navigation FVF layers are calculated except they guide the pedestrians towards the Area of Attraction instead of towards their exit. Area of Attraction and Area of Avoidance layers are all also applied to these layers.

Once all of the navigation FVFs have been created, taking into account both Areas of Attraction and Areas of Avoidance, they are then smoothed. Smoothing is used to avoid diagonal convergence of pedestrians due to harshly formed FVFs that guide pedestrians using the absolute minimum distance to their exit. Smoothing is done by using a nearest neighbour average, where each cell is an average of all its non-zero vectors neighbours within a particular radius. This is one of the more computationally demanding parts of the algorithm so summed area tables are used to compute averages efficiently.

Once all FVFs have been calculated, all fields are then converted to the pedestrian simulation's navmap data structure. Information about whether each navmap agent is an entrance/exit is also added, and also its height if it is part of the collision area. These navmap agents are then encoded into a binary format and saved. This binary also includes global configuration including emission rates, Area of Interest layer probabilities and the reference image bitmap.

3.6. Final Simulations

Once an environment has been compiled and saved it can then be used to initialise a simulation. In PED, authors are provided functionality to compile and start a simulation with a single button click. The simulation program then loads the environment, and displays the visualisation of the simulation to the user (Figure 1). Pedestrians can be seen to walk around in accordance with the environment, collision area models are swept upwards to their configured height, and the reference image can be seen underneath the pedestrians.

When a simulation is started, the simulation program continually watches the binary file that the environment was loaded from. If it is overwritten, the new environment is loaded into the simulation. All of the current pedestrians remain, but the environment is updated. From within the PED interface, the user only need press a button to update the simulation with the current version of the environment seen in the editor. This allows for dynamic environment update at runtime.

4. Results and Discussion

4.1. Sample Environments

Six sample environments were created to exercise the system: a train station, a university student union, a supermarket, a simple environment with doors, a festival, and a swirling path. Table 1 details how many layers each environment used, the average number of pedestrians and the time required for an experienced user to create the environment. While the number of layers may seem high, most are simple layers (e,g, an entrance/exit), and the number was also increased by splitting collision layers into multiple collision pieces, both for organisation purposes (i.e. focussing on different areas of collision in the environment) and to create a visualisation of different 3D heights in the simulation.

The creation of these sample environments showed that the system is capable of creating specific scenarios. Entrance and exits could be created to define the flow of pedestrians in a controlled fashion. Collision layers could be used to create buildings, fences or other obstacles within the environment. Areas of Attraction were particularly useful for defining paths and in some cases, such as tight corners, proved useful in making pedestrian flows look more organic (Figure 6). While Areas of Avoidance acted in much the same way to Areas of Attraction, they were useful for easily

Environment	Layer count	Peds.	Time
Train station	28	540	50 mins
Student's Union	31	560	60 mins
Supermarket	25	220	60 mins
Festival	15	52000	10 mins
Doors	12	2400	10 mins
Swirl	11	18500	10 mins

Table 1: *Sample environment metrics*

defining concepts such as roads. Environments including roads that used Areas of Avoidance to mark them would see pedestrians either use a crossing if provided, or otherwise minimise their time spent in the road by crossing orthogonally rather than diagonally.

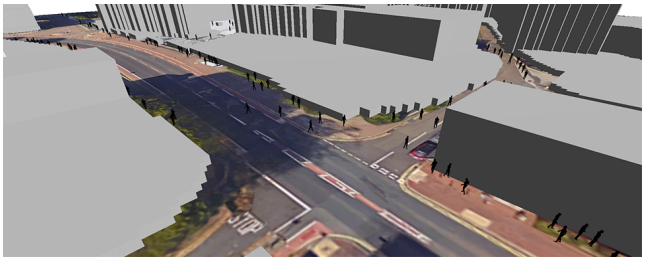

Figure 6: *Pedestrians navigating a street corner.*

Figure 7: *Birds-eye view of Student's Union sample environment.*

Areas of Interest were effective at filling buildings with people (Figure 7). The train station (Figure 8) environment used them to populate a main lobby where pedestrians would view arrival boards and the student union environment used them to populate inside the union building. Reference layers proved useful in providing spatial context for the created environments, especially when used in conjunction with the swept Collision layer models. Dynamic update provided the means to create environments that included doors, and also vary pedestrian distribution models. Dynamic update was also used extensively while creating the sample environments to iteratively improve the environment based on a short feedback loop of editing, compiling and viewing.

The sample environments also highlighted that there are still improvements that can be made within the system. The behaviours

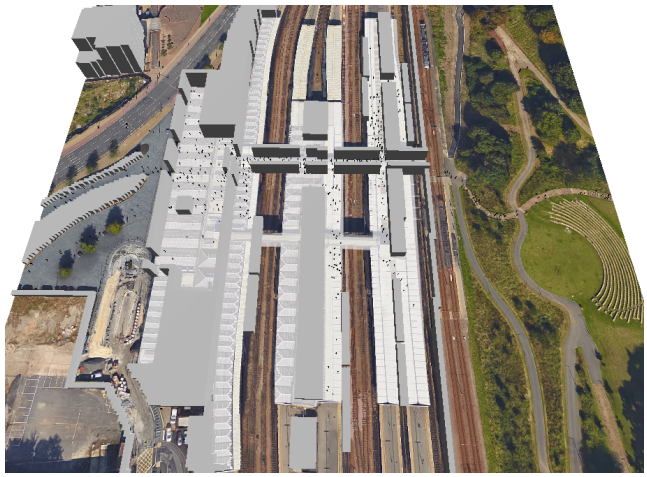

Figure 8: *Train station sample environment.*

available in the system were shown to have limitations in certain situations. The Areas of Attraction and Avoidance were shown to not be able to model pedestrian heterogeneity adequately in all situations. Pedestrians moving to a single exit tend to all move in the same direction, and creating environments where pedestrians would take multiple routes proved difficult. Areas of Interest were good at easily being able to change pedestrian goals within the environment, but exhibited a lack of pedestrian control when pedestrians reached the defined area as they no longer had a goal to infer navigation from (Figure 9).

Figure 9: *Supermarket sample environment.*

The lack of support for multiple environment heights was also problematic. Several sample environments are spread over multiple levels (e.g. the train station environment which has stairways and corridors over train tracks, see Figure 8) and the single plane that PED offers is insufficient to model these. In cases requiring multiple heights, compromises needed to be made to model areas on a 2D plane. Also, the FVF system used within the simulation does not account for pedestrian congestion, which

means pedestrians do not alter their navigation to try and avoid build ups.

4.2. User Testing

To assess the system's usability and to check whether it is usable by non-technical authors, eight users were asked to use the tool to create a prescribed environment. Participants were first given written instructions that guided them through making a sample environment using the tool, and then were asked to make one of their own. Participants then filled in a survey that detailed what they thought about the system and their experience creating the prescribed environment. The environment they were asked to make was the area around a church in the middle of a city (Figure 10). All participants knew the area well in real life. They were given a satellite image as a reference image and a maximum of an hour to create a visually-convincing pedestrian simulation of the area.

The eight participants almost unanimously agreed that the system was intuitive, and that pedestrian simulations were easy to create using the system. They thought that the system provided tools that were simple to use even with no prior technical knowledge and it could be used to adequately control pedestrians within an environment to produce pedestrian flows that the felt mimicked real life. The results also show that the participants valued the ability to dynamically update the environment and the immediate feedback directly aided their creation process.

All participants were able to create the church environment (Figures 10 and 11) and were convinced that what they had created was accurate. The average time that participants used to make the environment was 44 minutes. Table 2 details how many layers each participant used, the average number of pedestrians in the church environment and the creation time required. Participants created the environment using varying numbers of layers. This was due to personal participant preference to the level of layer organisation and collision visualisation creation. The most used layers were collision layers which made up 36% of the layers used followed by Entrance/Exit layers that contributed 32% of the used layers. It is possible to condense all of the collision layers for a PED environment into a single layer, but these statistics show that users value the organisational and visualisation benefits gained from using separate layers.

Participant	Layer count	Peds.	Time
A	14	140	54 mins
B	22	150	47 mins
C	23	200	43 mins
D	18	190	30 mins
E	15	90	40 mins
F	18	150	42 mins
G	19	210	51 mins
H	17	134	40 mins

Table 2: *Church environment metrics*

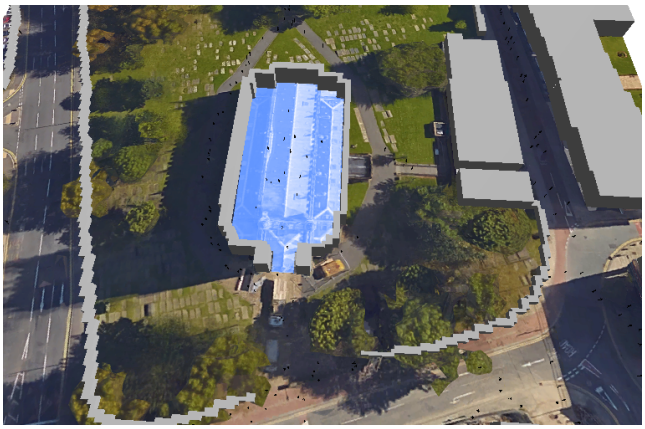

Figure 10: *Evaluation church environment 1.*

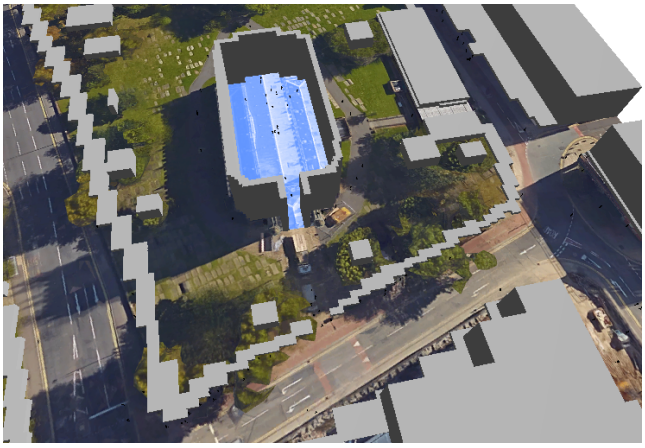

Figure 11: *Evaluation church environment 2.*

5. Conclusions

We have shown that PED can be used to create complex environments and can be used by non-technical authors to create pedestrian simulations quickly and efficiently. Simulation environment update has been shown to make dynamic scenarios possible while also providing a mechanism that authors can use as an interactive feedback loop to help easily fine-tune their environments.

There is still room for improvement. Currently different levels within a 3D simulation cannot be modelled, e.g. the stairs and passages over train tracks. One way to address this may be to introduce a teleportation layer type which could be used to transport pedestrians from one level or area of the environment to another. Other functionality could also be considered, such as Guidance Fields [PVdBC*11], which define areas in which pedestrians can only move in one direction, and continuum dynamics [TCP06] to address issues such as congestion avoidance. Further work could also be done on the software system itself so that users could specify and compile new layer types themselves.

References

[BAK08] BANERJEE B., ABUKMAIL A., KRAEMER L.: Advancing the layered approach to agent-based crowd simulation. In *Principles of Advanced and Distributed Simulation, 2008. PADS'08. 22nd Workshop on* (2008), IEEE, pp. 185–192. 2, 3

[Che04] CHENNEY S.: Flow tiles. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), Eurographics Association, pp. 233–242. 2

[fla] FLAME GPU, http://www.flamegpu.com/. 1, 3

[GCC*10] GUY S. J., CHHUGANI J., CURTIS S., DUBEY P., LIN M., MANOCHA D.: Pledestrians: a least-effort approach to crowd simulation. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation* (2010), Eurographics Association, pp. 119–128. 2

[HM95] HELBING D., MOLNAR P.: Social force model for pedestrian dynamics. *Physical review E 51*, 5 (1995), 4282. 2, 3

[Hug02] HUGHES R. L.: A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological 36*, 6 (2002), 507–535. 2

[JPCC14] JORDAO K., PETTRÉ J., CHRISTIE M., CANI M.-P.: Crowd sculpting: A space-time sculpting method for populating virtual environments. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 351–360. 2

[KLLT08] KWON T., LEE K. H., LEE J., TAKAHASHI S.: Group motion editing. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 80. 2

[KRR10] KARMAKHARM T., RICHMOND P., ROMANO D. M.: Agent-based large scale simulation of pedestrians with adaptive realistic navigation vector fields. *TPCG 10* (2010), 67–74. 3

[KSKL14] KIM J., SEOL Y., KWON T., LEE J.: Interactive manipulation of large-scale crowd animation. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 83. 2

[MR05] MILLÁN E., RUDOMIN I.: Agent paint: Intuitive specification and control of multiagent animations. In *Proceedings International Conference in Computer Animation and Social Agents (CASA)* (2005). 2, 3

[PPD07] PARIS S., PETTRÉ J., DONIKIAN S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. In *Computer Graphics Forum* (2007), vol. 26, Wiley Online Library, pp. 665–674. 2

[PVdBC*11] PATIL S., VAN DEN BERG J., CURTIS S., LIN M. C., MANOCHA D.: Directing crowd simulations using navigation fields. *Visualization and Computer Graphics, IEEE Transactions on 17*, 2 (2011), 244–254. 7

[Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH computer graphics* (1987), vol. 21, ACM, pp. 25–34. 2

[Rey99] REYNOLDS C. W.: Steering behaviors for autonomous characters. In *Game developers conference* (1999), vol. 1999, pp. 763–782. 1, 2, 3

[RR11] RICHMOND P., ROMANO D.: Template-driven agent-based modeling and simulation with cuda. *GPU Computing Gems Emerald Edition* (2011), 313. 1

[TCP06] TREUILLE A., COOPER S., POPOVIĆ Z.: Continuum crowds. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 1160–1168. 2, 7

[TYK*09] TAKAHASHI S., YOSHIDA K., KWON T., LEE K. H., LEE J., SHIN S. Y.: Spectral-based group formation control. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 639–648. 2

[UCT04] ULICNY B., CIECHOMSKI P. D. H., THALMANN D.: Crowdbrush: interactive authoring of real-time crowd scenes. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), Eurographics Association, pp. 243–252. 2, 3

[YMPT09] YERSIN B., MAÏM J., PETTRÉ J., THALMANN D.: Crowd patches: populating large-scale virtual environments for real-time applications. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games* (2009), ACM, pp. 207–214. 2