EG UK Computer Graphics & Visual Computing (2017)
Tao Ruan Wan and Franck Vidal (Editors)

Sketching for Real-time Control of Crowd Simulations

Luis Rene Montana and Steve Maddock

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

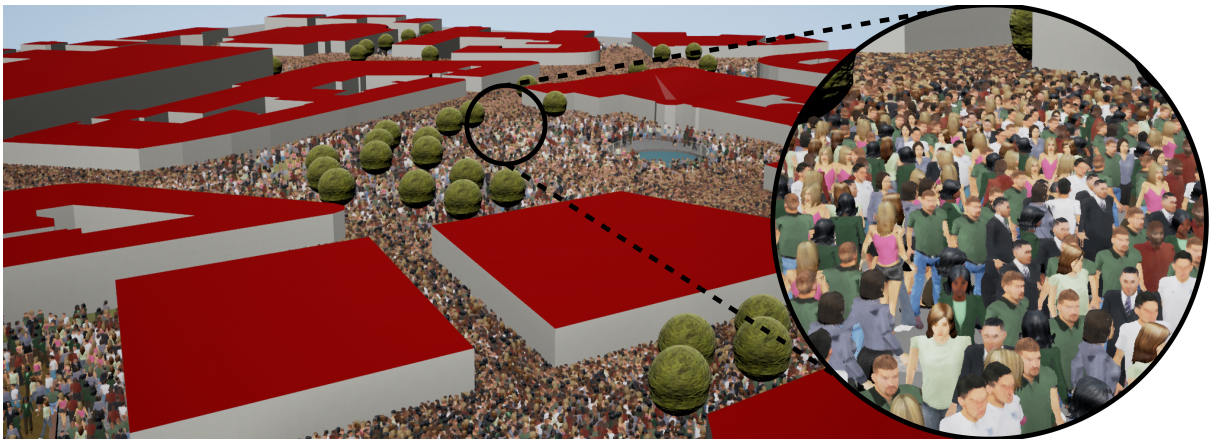

Figure 1: *Simulation with 50,000 pedestrians running at 15 frames per second.*

Abstract

Crowd simulations are used in various fields such as entertainment, training systems and city planning. However, controlling the behaviour of the pedestrians typically involves tuning of the system parameters through trial and error, a time-consuming process relying on knowledge of a potentially complex parameter set. This paper presents an interactive graphical approach to control the simulation by sketching in the simulation environment. The user is able to sketch obstacles to block pedestrians and lines to force pedestrians to follow a specific path, as well as define spawn and exit locations for pedestrians. The obstacles and lines modify the underlying navigation representation and pedestrian trajectories are recalculated in real time. The FLAMEGPU framework is used for the simulation and the game engine Unreal is used for visualisation. We demonstrate the effectiveness of the approach using a range of scenarios, producing interactive editing and frame rates for tens of thousands of pedestrians. A comparison with the commercial software MassMotion is also given.

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, I.6.8 [Simulation and Modeling]: Types of Simulation—Visual, H.5.2 [Information interfaces and presentation]: User interfaces—Graphical user interface(GUI)

1. Introduction

Pedestrian simulations have numerous applications, including films, video games, training, urban planning, and emergency evacuation simulations. Such simulations are often controlled by an agent-based model, which may produce unexpected global behaviour. Numerous system parameters must then be tuned to control a simulation, which is time-consuming and requires a certain level of knowledge of the parameter set, the simulation process and potential outcomes of changing parameters. This challenge is then compounded with advances in computer technology, which supports the modelling of bigger crowds and more complex pedestrian behaviour, with more complex parameter sets.

Several approaches involving graphical interaction, such as sketching, have been proposed to facilitate crowd control and to enable non-expert users to modify the behaviour of the pedestrians.

DOI: 10.2312/cgvc.20171282

This movement can be controlled directly by interacting with the agents or indirectly by modifying the environment - Section 2 gives details. We present an interactive graphical control approach where non-technical users can change the flow of the pedestrians by sketching in the simulation environment with a mouse.

The system consists of an agent-based pedestrian simulation, developed using the *FLAMEGPU* framework [fla], linked with the game engine *Unreal* for rendering. The user is able to sketch lines in the simulation environment to represent guiding paths or obstacles. The underlying grid-based navigation method [KRR10] is updated based on the user sketches. Pedestrians react to these environment changes in real time and modify their trajectory accordingly. The simulation runs at interactive frame rates for tens of thousands of pedestrians.

The approach we use is related to Patil *et al*'s work [PVDBC*11], in that flow lines can be sketched to direct pedestrians in our system. However, we offer three additional features. First, the novel aspect of our work is that obstacles can be created, cut and deleted by sketching in real-time during the simulation in order to direct the crowd. Second, the user can specify the pedestrian spawn locations and goals by sketching in the scene before the simulation starts (similar to Oshita and Oqiwara's work [OO09]). Third, we use multiple navigational layers to guide agents, one for each exit, following Karmakharm *et al*'s work [KRR10].

The rest of the paper is organised as follows. Section 2 covers related work in the field. Section 3 describes the implementation of the system. Section 4 shows and discusses the results. We present a range of scenarios to demonstrate paths through sets of barriers, multiple flow directions in corridors, and turnstile behaviour. A comparison with the commercial software *MassMotion* is also given. Section 5 concludes the paper.

2. Related Work

The most common approach used to model crowd simulations is based on agents. The movement of every pedestrian is individually computed following a set of rules. This agent method offers some advantages over macroscopic models [KRR10], such as behaviour diversity and emergent global behaviours. Reynolds's pioneering work [Rey87] simulated a flock of birds by controlling the agents with a set of rules. Additional rules or steering behaviours, such as seek, flee and pursuit, were defined in subsequent work [Rey99, Rey00]. These individual motions produce complex global behaviours which are complicated to predict and control. Anderson *et al* [AMC03] addressed this issue by adding spatial and shape constraints to the animated flock. Helbing proposed a different approach to model agents' dynamic using 'social forces' [Hel91, HM95], which are the result of pedestrians' interaction with the environment and with each other, an approach that has been extended and applied to simulate emergency situations [HFV00, BMdOB03]. Further sociological aspects have also been considered [MT97, PHDL07, YT07].

The behaviour of the pedestrians may be determined by two forms of control: local and global. A combination of both controls is used in most crowd simulation systems. Local motion refers to agent movement that considers immediate surroundings. Obstacle avoidance and agent interaction are typical behaviours produced by this control, as described above. Global navigation is used to navigate through complex environments. The most common approaches for global path planning are graphs and flow fields. A navigation graph is based on the idea of dividing the walkable areas of the environment into cells. After the decomposition, adjacent nodes are linked with edges to create the graph [LD04]. For flow fields, first proposed by Reynolds [Rey99], the environment is mapped to a two-dimensional grid. Each cell contains a force vector to guide agents to their goal. Several simulation systems have used this global navigation technique [JXW*08, PVDBC*11, MMHR16]. Chenney [Che04] proposed a tool to create flow fields by connecting small areas of forces called 'Flow tiles'. Karmakharm *et al* [KRR10] employed multiple layers of flow fields to guide pedestrians to different destinations. The force vectors and pedestrians were designed as agents to allow parallel computation using GPU hardware.

Virtual crowd simulations usually involve many parameters to configure in order to control the behaviour of the pedestrians. The process of tuning these parameters requires prior knowledge and can be a trial-and-error task since it is difficult to predict the resulting animation. Numerous approaches have been suggested to address these issues by creating virtual crowd simulation systems that include a graphical user interface to facilitate crowd control. Figure 2 shows a classification of these graphical control methods.

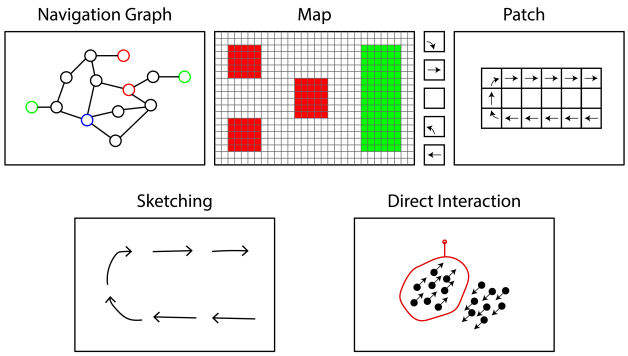

Figure 2: *Graphical control approaches of crowd simulations.*

The *Navigation Graph* category enables the creation or modification of graphs to control the movement of the agents. Yersin *et al* [YMC*05] developed a system in which a predefined environment is divided into navigable areas to form a graph. The user is able to select and label the nodes. Pedestrian goals can be reassigned to one of the marked nodes.

The *Map* approach consists of drawing maps or layers on top of the environment to add information and indirectly modify the behaviour of the agents. Sung *et al* [SGC04] created a crowd simulation where the behaviour of the pedestrians is influenced by 'situations' attached to the environment. These situations are specified by the user by drawing regions on the environment using a painting interface. Millan and Rudomin [MR05] used

several maps to define attributes such as the height and texture of the environment. Similarly, McIlveen *et al* [MMHR16] created a painting tool to draw multiple layers on top of the environment to specify entrances, exits, obstacles and areas of interest. Jordao *et al* [JCC*15] proposed a system where the user is able to determine the direction and density of the crowd by painting grayscale maps on top of the environment model.

The *Patches* method allows the user to create large environments by combining multiple small predefined patches or blocks. Chenney [Che04] used small areas of force fields called *Flow Tiles*, which could be connected together to form bigger flow fields to guide pedestrians across an environment. Yersin *et al* [YMPT09] developed a system called *Crowd Patches*, which are blocks with pre-computed animations and trajectories. The user can create an environment in two ways: from an empty scene, connecting patches; or from a predefined environment, dividing it into polygonal shapes to create the patches. Jordao *et al* [JPCC14] extended this work to create a new system called *Crowd Sculpting*. In this approach, *crowd patches* can be stretched, shrunk, bent, combined or deleted to form the environment.

The *Sketching* approach allows the user to sketch in the environment to alter pedestrian behaviour. Jin *et al* [JXW*08] developed a system to control pedestrians by drawing arrows in the environment. The global movement of the agents is determined by vector fields. The graphical interface allows the user to draw anchor points with a direction to determine the path of the pedestrians. Another sketching approach was proposed by Oshita and Ogiwara [OO09]. In this approach, the user is able to define the area where pedestrians will appear and the main moving trajectory of the entire crowd. Additional lines or paths can be drawn to set other parameter values such as the distance between virtual characters. Patil *et al* [PVDBC*11] suggested a graphical user interface with brush tools to draw arrows to guide the pedestrians throughout the environment. A navigation field is constructed using the drawn motion trajectories as a reference. The velocity of each virtual agent is obtained by mapping their positions into the vector field.

In the *Direct Interaction* techniques, the user directly controls the pedestrians by selecting them to modify their behaviour, to create formations or to move them to another location. In Ulicny *et al*'s work [UCT04], the user is able to directly modify the behaviour of the virtual characters by using artistic tools, such as brushes, in a 2D canvas that represents the 3D scene. The designer is allowed to create and delete pedestrians, start animations, create paths and modify the appearance of the agents. Kwon *et al* [KLLT08] devised a technique to edit an existing animation of a group of virtual characters. A graph is constructed from the existing motion clips and its vertices are sampled from each trajectory. The user can merge two graphs or deform them by pinning and dragging their vertices in order to modify the animation.

Similar work was done by Kim *et al* [KSKL14]. Here, the user is able to edit an existing animation by enclosing characters in a cage that supports space and time manipulation. An interface assists the user to construct a cage using a freehand selection tool for selecting a group of characters. The user can drag and pin down boundary vertices or interior points to manipulate the cage. Kim *et al* [KHKL09] also presented work where the user is able to interactively modify the movement data of several characters. The user can manipulate the position, direction and synchronisation with spatial and temporal constraints.

Takahashi *et al* [TYK*09] presented an approach to control group formations while keeping the adjacency relationships between agents. The system requires a set of user-defined keyframe formations to interpolate through using spectral analysis. A formation is formed by directly specifying the position of every agent. Henry *et al* [HSK12] proposed a method to control crowds using a multi-touch device. The crowd is represented by a deformable mesh; the user can modify this mesh by selecting control points and dragging them to define the final mesh.

Research for creating and controlling group formations was carried out by Gu and Deng [GD11]. The user can draw or sketch lines and curves to define the boundaries of a formation. Gu and Deng extended this work by adding new features and tools to facilitate control of group formations [GD13]. The user has three options to input the formation. Additionally, the user is able to sketch global trajectories to guide the group to a final location. Allen *et al* [APKM15] developed a similar system to control characters at individual or group level by creating formations specified by the user. An additional feature of this work is the possibility of defining subgroups and specifying a specific path for the subgroup. Table 1 summarises the research involving graphical control.

Work	Category	Agent	Environment	Navigation
[APKM15]	Direct Interaction	✓		?
[Che04]	Patches		✓	Flow field
[GD13]	Direct Interaction	✓	✓	?
[GD11]	Direct Interaction	✓		?
[HSK12]	Direct Interaction	✓		Flow field*
[JXW*08]	Sketching		✓	Flow field
[JPCC14]	Patches		✓	Flow field
[JCC*15]	Maps		✓	Graph
[KSKL14]	Direct Interaction	✓		?
[KHKL09]	Direct Interaction	✓		?
[KLLT08]	Direct Interaction	✓		?
[MMHR16]	Maps		✓	Flow field
[MR05]	Maps		✓	?
[OO09]	Sketching		✓	?
[PVDBC*11]	Sketching		✓	Flow field
[SGC04]	Maps / Direct	✓	✓	?
[TYK*09]	Direct Interaction	✓		?
[UCT04]	Direct Interaction	✓		?
[YMC*05]	Navigation Graph		✓	Graph
[YMPT09]	Patches		✓	Flow field

Table 1: *A summary of the graphical control approaches. The Agent and Environment columns indicate whether the agent behaviour is directly controlled by changing agent parameters and/or indirectly by modifying the environment, respectively. The Navigation column shows the underlying navigation method used. Papers not stating which navigation approach is used are marked with '?'. (* used for obstacle avoidance not general movement direction.)*

3. The System

The objective of the system is to allow real-time user control of the movement of the agents in an easy and intuitive manner. The user can specify the pedestrian spawn and exit locations, create barriers to pedestrian movement and force pedestrians to follow a path by sketching lines in the environment. An overview of the system is given in Figure 3. The simulation part is based on Karmakharm *et al*'s work [KRR10] which uses the FLAME GPU framework to create an agent-based simulation. The game engine, *Unreal Engine*, is employed to improve graphics quality. These simulation and visualisation parts share information through a shared memory segment. The environments are initialised based on data from *OpenStreetMap* [ope].

The behaviour of the agents is driven by three forces: navigation, collision and pedestrian avoidance. The first two forces are based on the collision and navigation maps (Section 3.1) and the location of the agents. The pedestrian positions are mapped to corresponding elements of the two maps to get the collision and navigation forces. The pedestrian avoidance force is computed using the position and velocity of each agent within a certain radius. The resulting force that determines the movement of the agents is calculated as a weighted sum of the navigation, collision and avoidance forces. This approach could be improved by using more sophisticated social force models, but serves the purposes required for the sketching work.

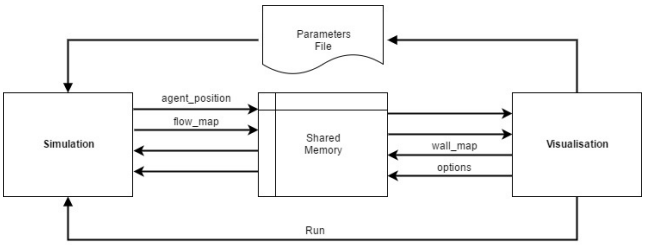

Figure 3: *System overview.*

3.1. Navigation

The approach selected to guide agents across the environment is force fields. The environment is divided into a two-dimensional grid in which every cell contains a force. Two types of map are generated: a single collision map, defining obstacles in the environment; and multiple navigation maps, guiding agents to their destination, where the number of navigation maps depends on the number of exits in the environment. The collision map is generated by ray casting each cell to obtain the height of the environment at the collision point. A predefined value determines if the cell is considered as an obstacle or walkable space. The navigation maps are represented by a 2D array, each cell containing a force pointing towards the specified exit following the shortest path. To generate these maps, cells representing walkable areas are initialised with 0. The exits and obstacles cells are marked from the collision map. A wavefront propagation algorithm calculates the distance from each cell to the corresponding exit. The force direction of each cell is towards the neighbouring cell with the shortest distance. A smoothing algorithm is applied to the navigation maps to create more natural pedestrian flows.

Figure 4 shows an example set of maps for an environment of size 16x16 cells (a size chosen purely for illustrative purposes.) A more typical environment size is 256x256 cells. In Figure 4, the top map is the collision map, which has nine obstacles and boundaries around the edge of the environment to keep the pedestrians from exiting it. There are four navigation maps, one for each exit. Pedestrians are initialised into the environment from a spawn point and use the navigation map associated with their specific exit point. Spawn and exit points can coincide.

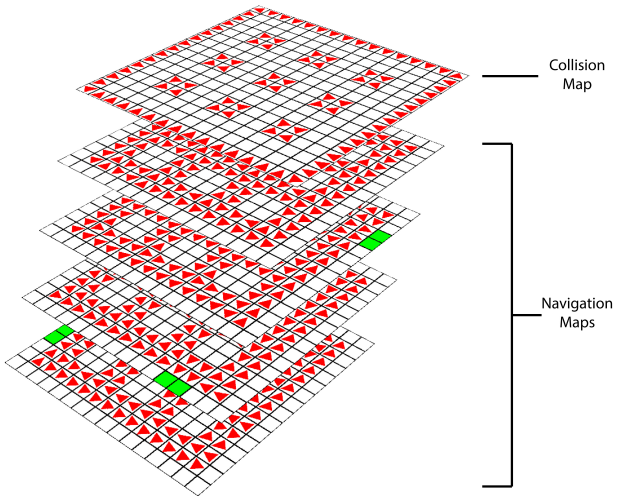

Figure 4: *Grids generated for a simple, illustrative environment with four exits. There is one common collision map and one navigation map per exit. Spawn and exit points are marked in green.*

3.2. Sketching

The user has the ability to create, cut and delete barriers by drawing lines with the mouse. The space between sampled points in a stroke depends on the drawing speed of the user in relation to the rendering frame rate [OSSJ09]. Thus, the line is resampled to ensure a minimum number of equidistant points when creating barriers. A barrier is spawned in the scene when the user releases the mouse (Figure 5). The sketched line is mapped to the collision map marking the new cells as obstacles. The width of the barriers is fixed to three cells; a smaller width of influence could result in agents being pushed through obstacles for high-density crowds, because of the workings of the grid-based approach. A barrier can subsequently be cut to create a gap where pedestrians are able to walk (Figure 5d). After editing, the navigation grids are automatically updated with the shortest path recalculated using the new collision map.

The flow of the pedestrians can also be controlled by drawing flow lines (see Figure 3). The sketching and sampling process is identical to the barrier creation. An arrow is drawn in the environment pointing towards the direction of the user sketch.

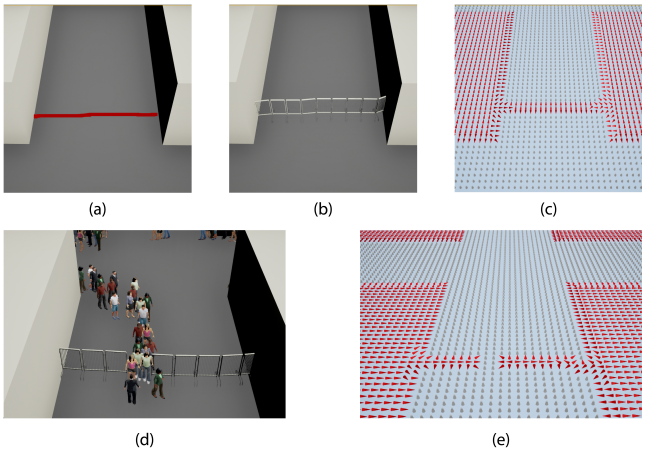

Figure 5: *(a) Line sketched by the user, (b) barrier created afterwards and (c) collision map updated with the new obstacle. (d) Pedestrians walking through a cut wall. (e) Collision map after creating the cut wall.*

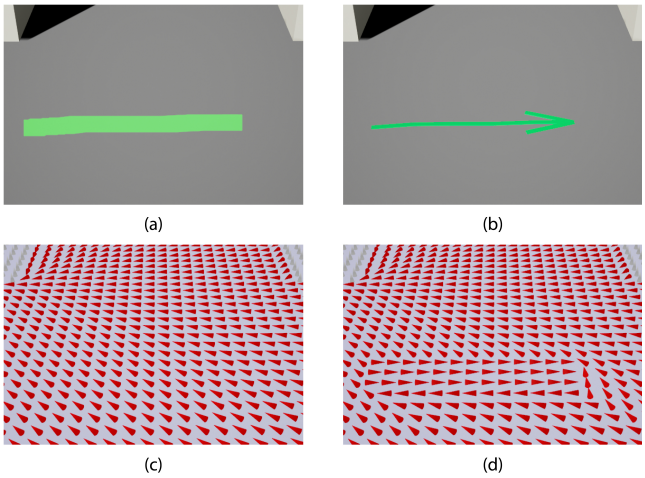

Figure 6: *(a) Line sketched by the user and (b) arrow created when finishing the sketch. Navigation map (c) before and (c) after the flow line.*

This flow line is mapped to all the navigation maps, replacing the previous force values of the involved cells. An alternative would be to blend the new values with the existing values. However, replacement avoids the problem where overlapping opposing arrows could cancel out their respective forces, resulting in a null zone of no movement. The width of influence of the arrows is set to three cells, and is not currently user-configurable.

A potential problem may occur when a flow line force and a neighbouring cell force of the navigation map are completely or nearly opposite. A pedestrian walking in that area could become 'trapped' by those two forces, circulating in a small area. The issue is addressed by considering the opposite flow line cells as obstacles. The navigation map is recalculated avoiding the opposing arrow (see Figure 6d). As a result, pedestrians avoid walking into the cells with opposite direction. In densely crowded environments, agents might be pushed into these undesired flow lines. In this case, agents will follow the arrow and then retake their original path, walking around the flow line, to their destination. Patil *et al* [PVDBC*11] solved this problem by assigning a cost to each cell. This cost depends on the direction in which the cell is traversed respecting the sketched flow line.

After sketching, the information for newly created barriers and flow lines is communicated to the simulation module through the shared memory segment residing on the CPU (Figure 3). From *Unreal*, sketched data is copied to the shared memory segment. The simulation module reads this data and uses it to recompute the maps, which are then copied to the GPU to process the next iteration of the simulation. After the iteration the simulation module copies the pedestrian positions back to the shared memory for *Unreal* to use in visualisation.

3.3. Rendering

Unreal Engine is used to render the 3D environment and the pedestrians. This game engine is linked with FLAMEGPU through the use of shared memory segments (Figure 3). The information shared between the two modules includes agent positions, navigation maps, and sketched barriers and flow lines.

Environments can be created using *OpenStreetMap*. Downloaded data can be converted to a 3D model using the free tool *OSM2World* [osm]. Figure 1 shows a model of part of the centre of Sheffield. The model was modified before being imported to *Unreal*: the billboard model of a tree was replaced with a full 3D model; the ground data was cleaned up to remove overlapping areas of grass and road; the grass and road materials were replaced with better-looking materials. The automatic generation of the collision map allows the use of any 3D model. The pedestrians are rendered as instances of eight base character models. The use of instances is to reduce the number of draw calls and considerably increase the number of agents without impacting the performance of the system. The position of each pedestrian is read from the shared memory segment and updated every iteration.

The character models were created with the free online tool *Autodesk Character Generator*. Unreal Engine does not allow the instance creation of models with animated skeletons, therefore, this approach could not be used to render the crowd. The solution to this problem was to remove the skeleton of the models and create a vertex animation using the software *Autodesk 3ds Max*. This animation was converted to a texture using an Unreal script. The texture is applied to the material used in the static meshes of the characters to animate the pedestrians. A random character model is assigned to each agent when they are generated. For 10000 agents, the frame rate is 45-55fps, whereas for 50000 agents, the frame rate reduces to 10-15 fps, depending on camera distance and the number of agents in view.

4. Results

The system provides an intuitive graphical way to interact with the simulation by modifying the environment and influencing the

pedestrian movement. This is illustrated using a range of scenarios: path control using barriers (Figure 7), controlled lane formation (Figure 8) and the use of turnstiles to control movement (Figure 9). A video of the system is available at https://tinyurl.com/yczspu77.

Figure 7 shows pedestrians adjusting their path to avoid the barriers created by the user. A barrier is used to block off access to one corridor and multiple barriers are used to produce a snake of movement for a group of pedestrians.

Figure 7: *Pedestrians following the path created by barriers.*

Figure 8 shows lane formation control. Such motion can be observed in real crowds, for example when groups of pedestrians walk in opposite directions at road crossing points. This behaviour can be simulated by sketching opposite arrows next to each other. In Figure 8, multiple lanes are created in the same corridor, and pedestrians avoid collisions with pedestrians walking in a different direction. Whilst lane formation can emerge in agent-based simulations, our system offers easy control over where it occurs.

Figure 9 shows the use of barriers, cuts and arrows to create turnstile-like behaviour, as might be seen at the entrance/exit of a train station. A barrier is created to block the path and two holes are cut to allow pedestrian flow. Pedestrians trying to walk through a narrow space in opposing directions cause congestion for each turnstile, as illustrated in Figure 9c. To address this issue opposite arrows are drawn, one in each gap, to force pedestrians to move in the specified direction. A similar scenario is shown in Patil *et al*'s work [PVDBC*11], however, a predefined environment is needed, whereas in our system the entire scenario can be recreated by real-time sketching at any position in any environment.

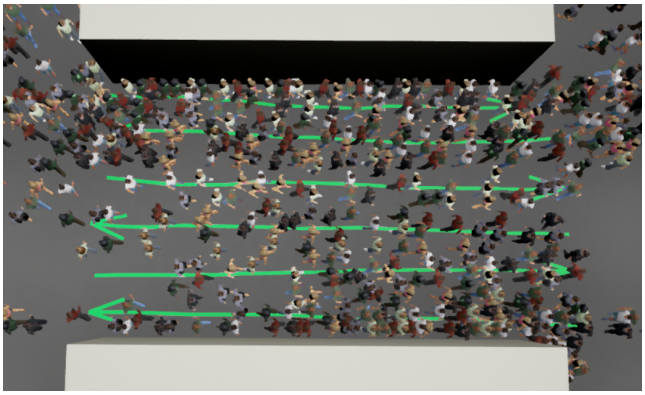

Figure 8: *Lane formation behaviour simulated with multiple flow lines.*

We can also compare our sketching work to other research work. Jin *et al*'s work [JXW*08] uses flow fields to control multiple crowds, which can be updated in real time by the user adding or deleting 'anchor points' with associated direction. However, as the number of points increases, the generation of the vector fields by radial basis functions based vector interpolation becomes more expensive having an impact on the simulation performance. In our work, the number of arrows does not affect the performance since only the existing grid forces are altered. Oshita and Oqiwara's work [OO09] uses 'guiding paths' for pedestrians, but does not allow the user to update these in real time, unlike our system. Also, neither of these approaches has the feature of adding obstacles to modify the environment.

Our sketching approach is a more intuitive and user-friendly way to interact with a simulation than that offered in commercial simulation software such as *MassMotion*. The creation of barriers in *MassMotion* requires the user to terminate the simulation, enter the scene editor mode and then create the obstacle. These steps have to be done offline and then the simulation is rerun, whereas in our system the barrier can be created in real time by sketching a single line. As another example, turnstile behaviour simulation is a more complicated process in the commercial software. The ground plane must be split and then linked with special connection objects. The direction of the flow can be set in the object properties. This task requires the user to have more than basic knowledge of the software interface. Using the sketching approach, this task can be done by drawing barriers and sketching arrows in real-time whilst the simulation is running. Of course, as commercial software, *MassMotion* has extra features that can be specified such as gate behaviour and delay between agents.

Our system has some limitations. The width of a stroke and its area of influence is set to a specific value, which means that very thin flow lines or barriers cannot be drawn and multiple arrows must be sketched to produce a wide path. Also, the underlying grid-based navigation approach means that drawing guiding paths for individual agents or small groups of agents is not possible. A sketch modifies areas of the underlying navigation grids and the force fields stored in the grids control all the pedestrians.

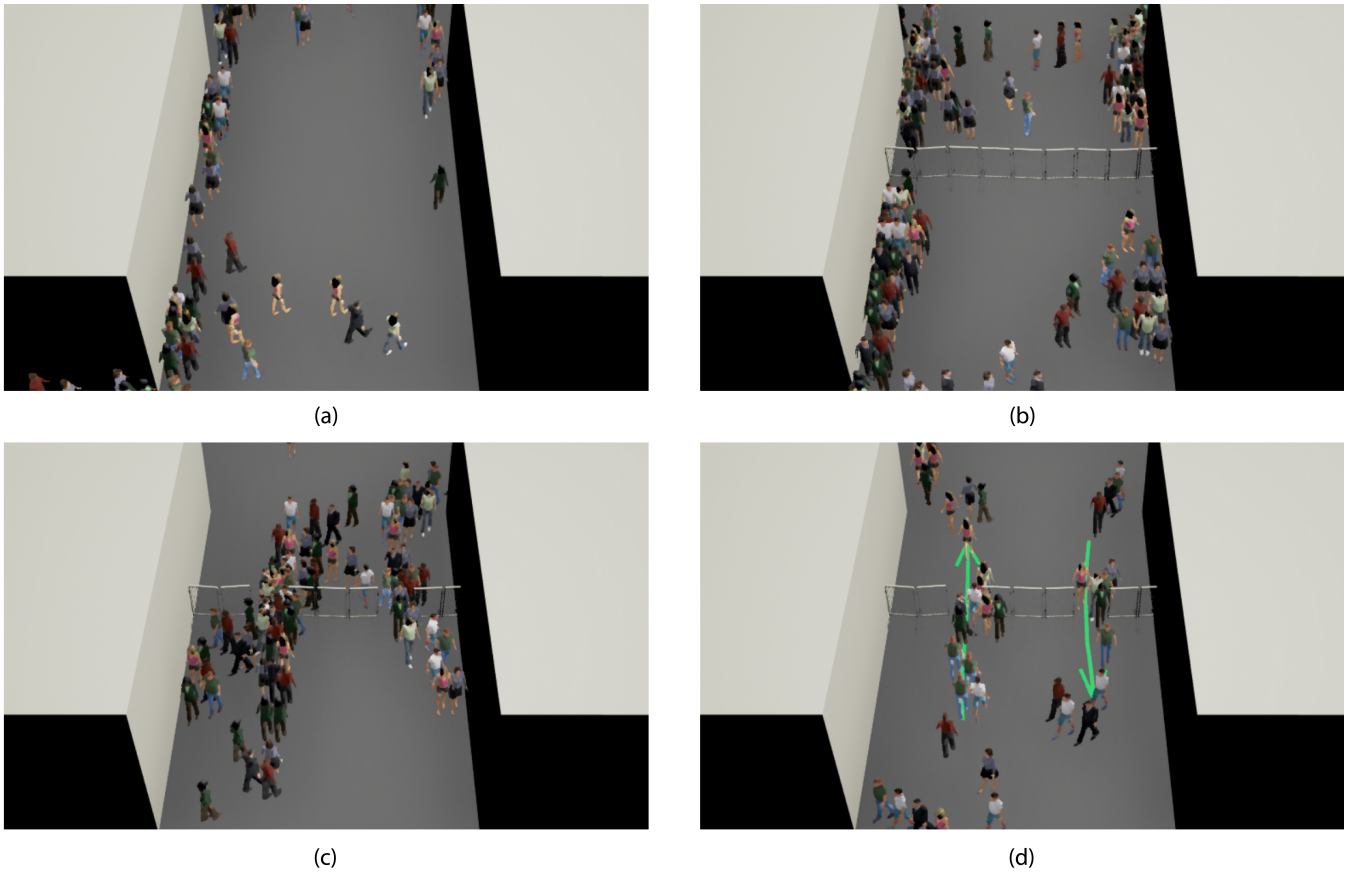

Figure 9: *Simulation of turnstile behaviour. (a) Pedestrians moving towards their assigned exit. (b) Agents changing direction after a barrier was created. (c) Congestion created due to opposing agent flow in small gaps. (d) Free pedestrian flow following the direction specified by the flow lines.*

5. Conclusions

We have presented an intuitive, graphical approach to control virtual crowd simulations by sketching lines directly in the simulation environment. A user is able to define the pedestrian spawn and exit locations, alter agent trajectories using barriers and force pedestrians to follow a desired direction using arrows of motion, giving control of global behaviours such as lane formation. Our approach does not require prior knowledge of a complex parameter set and saves on the time-consuming task of parameter tuning. Additionally, the use of the FLAMEGPU framework for simulation and the use of instance rendering in Unreal Engine means that we can simulate tens of thousands of agents while running at interactive frame rates.

The navigation grid-based approach does have some limitations. For example, individual agents or small groups of agents cannot be independently controlled. Also, the interaction between the sketch-based approach and the underlying grid has potential limitations. For example, when arrows are sketched in the current system, the arrow direction replaces the existing navigation grid force field. Further experimentation could be done on blending the arrow direction into the force field grid. However, the complexity of transitions between grid force field directions to an exit and prescribed sketched arrow directions would need careful consideration so as not to produce null force fields, stranding pedestrians in 'the doldrums', or perpetual circular motions, with no route to an exit.

The system could be used for investigating crowd control at mass demonstrations, during the design process of large venues, for emergency evacuation situations or to improve the flow of people in existing buildings. However, these applications may require a more sophisticated agent model, a different navigation approach and the ability to deal with multi-layer environments connected by complex stairways.

References

[AMC03] ANDERSON M., MCDANIEL E., CHENNEY S.: Constrained Animation of Flocks. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 286–297. 2

[APKM15] ALLEN T., PARVANOV A., KNIGHT S., MADDOCK S.: Using Sketching to Control Heterogeneous Groups. In *Proceedings*

Computer Graphics & Visual Computing (CGVC) (2015), The Eurographics Association. 3

[BMdOB03] BRAUN A., MUSSE S. R., DE OLIVEIRA L. P. L., BODMANN B. E.: Modeling individual behaviors in crowd simulation. In *Computer Animation and Social Agents, 2003. 16th International Conference on* (2003), IEEE, pp. 143–148. 2

[Che04] CHENNEY S.: Flow Tiles. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), SCA'04, Eurographics Association, pp. 233–242. 2, 3

[fla] Flexible Large Scale Agent Modelling Environment for the GPU. http://www.flamegpu.com. 2

[GD11] GU Q., DENG Z.: Formation sketching: an approach to stylize groups in crowd simulation. In *Proceedings of Graphics Interface 2011* (2011), Canadian Human-Computer Communications Society, pp. 1–8. 3

[GD13] GU Q., DENG Z.: Generating Freestyle Group Formations in Agent-Based Crowd Simulations. *IEEE Computer Graphics and Applications 33*, 1 (Jan. 2013), 20–31. 3

[Hel91] HELBING D.: A mathematical model for the behavior of pedestrians. *Behavioral Science 36*, 4 (Oct. 1991), 298–310. 2

[HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature 407*, 6803 (Sept. 2000), 487–490. 2

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

[HSK12] HENRY J., SHUM H. P. H., KOMURA T.: Environment-aware Real-time Crowd Control. In *Proceedings of the 11th ACM SIGGRAPH / Eurographics Conference on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2012), EUROSCA'12, Eurographics Association, pp. 193–200. 3

[JCC*15] JORDAO K., CHARALAMBOUS P., CHRISTIE M., PETTRÃL' J., CANI M.-P.: Crowd art: density and flow based crowd motion design. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games* (2015), ACM, pp. 167–176. 3

[JPCC14] JORDAO K., PETTRÃL' J., CHRISTIE M., CANI M.-P.: Crowd sculpting: A space-time sculpting method for populating virtual environments. *Computer Graphics Forum 33*, 2 (May 2014), 351–360. 3

[JXW*08] JIN X., XU J., WANG C. C., HUANG S., ZHANG J.: Interactive Control of Large-Crowd Navigation in Virtual Environments Using Vector Fields. *IEEE Computer Graphics and Applications 28*, 6 (Nov. 2008), 37–46. 2, 3, 6

[KHKL09] KIM M., HYUN K., KIM J., LEE J.: Synchronized multi-character motion editing. In *ACM Transactions on Graphics (TOG)* (2009), vol. 28, ACM, p. 79. 3

[KLLT08] KWON T., LEE K. H., LEE J., TAKAHASHI S.: Group Motion Editing. In *ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 80:1–80:8. 3

[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. 2, 4

[KSKL14] KIM J., SEOL Y., KWON T., LEE J.: Interactive Manipulation of Large-scale Crowd Animation. *ACM Trans. Graph. 33*, 4 (July 2014), 83:1–83:10. 3

[LD04] LAMARCHE F., DONIKIAN S.: Crowd of Virtual Humans: a New Approach for Real Time Navigation in Complex and Structured Environments. *Computer Graphics Forum 23*, 3 (Sept. 2004), 509–518. 2

[MMHR16] MCILVEEN J., MADDOCK S., HEYWOOD P., RICHMOND P.: PED: Pedestrian Environment Designer. In *Proceedings Computer Graphics & Visual Computing (CGVC)* (2016), The Eurographics Association. 2, 3

[MR05] MILLAN 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

[MT97] MUSSE S. R., THALMANN D.: A model of human crowd behavior: Group inter-relationship and collision detection analysis. In *Computer Animation and Simulation'97*. Springer, 1997, pp. 39–51. 2

[OO09] OSHITA M., OGIWARA Y.: Sketch-based interface for crowd animation. In *International Symposium on Smart Graphics* (2009), Springer, pp. 253–262. 2, 3, 6

[ope] OpenStreetMap. https://www.openstreetmap.org/. 4

[osm] OSM2world. http://osm2world.org/. 5

[OSSJ09] OLSEN L., SAMAVATI F. F., SOUSA M. C., JORGE J. A.: Sketch-based modeling: A survey. *Computers & Graphics 33*, 1 (Feb. 2009), 85–103. 4

[PHDL07] PAN X., HAN C. S., DAUBER K., LAW K. H.: A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations. *AI & SOCIETY 22*, 2 (Oct. 2007), 113–132. 2

[PVDBC*11] PATIL S., VAN DEN BERG J., CURTIS S., LIN M. C., MANOCHA D.: Directing crowd simulations using navigation fields. *IEEE Transactions on Visualization and Computer Graphics 17*, 2 (2011), 244–254. 2, 3, 5, 6

[Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH computer graphics 21*, 4 (1987), 25–34. 2

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

[Rey00] REYNOLDS C. W.: Interaction with groups of autonomous characters. In *Game Developers Conference* (2000), vol. 2000, pp. 449–460. 2

[SGC04] SUNG M., GLEICHER M., CHENNEY S.: Scalable behaviors for crowd simulation. In *Computer Graphics Forum* (2004), vol. 23, Wiley Online Library, pp. 519–528. 2, 3

[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. 3

[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. 3

[YMC*05] YERSIN B., MAÃŔM J., CIECHOMSKI P., SCHERTENLEIB S., THALMANN D.: Steering a virtual crowd based on a semantically augmented navigation graph. In *Proc. The First International Workshop on Crowd Simulation (V-CROWDS'05), Lausanne, Switzerland* (2005), Citeseer, pp. 169–178. 2, 3

[YMPT09] YERSIN B., MAÃŔM J., PETTRÃL' 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. 3

[YT07] YU Q., TERZOPOULOS D.: A decision network framework for the behavioral animation of virtual humans. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), Eurographics Association, pp. 119–128. 2