Large Scale Pattern of Life Simulation for Real-Time Applications

Nick Giannias, Evan Harris
CAE
Canada
nick.giannias@cae.com, evan.harris@cae.com

Roland Geraerts
Utrecht University
Netherlands
r.j.geraerts@uu.nl

Stijn Herfst
uCrowds
Netherlands
stijnherfst@ucrowds.com

Aidan Hobson-Sayers, Alistair Thorpe
Hadean Supercomputing
United Kingdom
aidanhs@hadean.com, alistair@hadean.com

ABSTRACT

A confluence of technologies has made the simulation of large-scale pattern of life possible to support real time applications. Ubiquitous cloud computing, new generation application scalability frameworks, advanced crowd simulation and modelling software, geospatial data production tools, and high fidelity/high performance visualization all contribute to the capability not only to simulate city- and country-sized populations but also model it in real time to support visual simulation applications.

Over the past year we have been prototyping a system that allows us to simulate large-scale patterns of life supporting real time, human in the loop operations for concept development, course of action analysis, and training. Specifically, we are building a system that not only simulates patterns of life at a city and country level but allows for visualization and interaction in an immersive, 3D environment from space down to street level. Our novel approach combines a population activity model, a high-fidelity crowd simulation, a scalability framework, and a game engine. The system simulates populations in the millions updated at 10 Hz with dense crowds of up to 100,000 people simultaneously visible in a scene rendered at 60 Hz.

We have built two use case demonstrators to showcase the capabilities of the system. The first is a civil resilience scenario involving a cyberattack on the London power network causing a disruption in the patterns of life. The second is a military course of action with ethnically motivated paramilitary operations and civil unrest in the Baltics.

This paper reports on the technical implementation of the system and the results of the two use case demonstrators. Throughout the report we compare with existing work and discuss lessons learned, concluding with a view to possible future development.

ABOUT THE AUTHORS

Nick Giannias is a Senior Technical Fellow working for the Advanced Technology and Innovation group at CAE and the Chief Architect for CAE's simulator products. With over 30 years of experience, he specializes in software development and real-time simulations for concept development, analysis, mission rehearsal, and training applications. Nick holds a bachelor's degree in Mechanical Engineering (Hons.) specializing in Aerospace from McGill University.

Dr. Evan P. Harris is a Senior Systems Architect at CAE Canada. Dr. Harris has a Ph.D. in Computer Science from The University of Melbourne, Australia, and has over 20 years' experience in providing modelling and simulation, systems, and software engineering services to Defense and Security customers in Canada, Australia, Asia and Europe.

Dr. Roland Geraerts is an Assistant professor with the Department of Information and Computing Sciences at Utrecht University, the Netherlands. He has been researching game technology and crowd simulation over 20 years, which resulted in a company (uCrowds) that was involved in making simulations for big sports events, infrastructures and games. uCrowds has won the 2020 Academic Startup Competition.

Stijn Herfst is a software engineer at uCrowds with a passion for high performance computing and algorithmic optimization. His research includes the specialization of k-nearest neighbor structures for dense crowd simulations. He holds an M.Sc. in Computer Science.

Aidan Hobson-Sayers is co-founder at Hadean, focusing on architecture and developer experience of distributed systems, and also spends time volunteering on the Core Team of the Rust programming language. He studied Computer Science at the University of Cambridge and co-authored a book on Docker.

Alistair Thorpe leads solution delivery at Hadean. His background is in the development of large scale, high performance distributed systems. He holds an M.Sc. in Computer Science from Oxford. His previous experience includes gaming and technology platform development.

Large Scale Pattern of Life Simulation for Real Time Applications

Nick Giannias, Evan Harris
CAE
Canada
nick.giannias@cae.com, evan.harris@cae.com

Roland Geraerts
Utrecht University
Netherlands
r.j.geraerts@uu.nl

Stijn Herfst
uCrowds
Netherlands
stijnherfst@ucrowds.com

Aidan Hobson-Sayers, Alistair Thorpe
Hadean Supercomputing
United Kingdom
aidanhs@hadean.com, alistair@hadean.com

INTRODUCTION

In the 2003-2007 time period, the United States Joint Forces Command conducted a massively distributed simulation of a synthetic urban environment called Urban Resolve. The overall goal of Urban Resolve was to "*guide the development of critical warfighting capabilities …with a particular focus on those needed for effective urban operations*" (Anastasiou, 2006). The synthetic environment consisted of the city of Jakarta with over one million buildings and 120,000 civilian entities operating in a culturally accurate manner. Attempting to hide within the city's population were 1100 enemy combatants. Human in the loop operators had control of approximately 250 assets including aerial vehicles and ground sensors and were tasked with locating and identifying the enemy force. The scale of the experiment is impressive, even by today's standards. Two supercomputing centers utilizing over 300 computing nodes were linked via a defense research network to multiple operator cells across the United States. Over 100 gigabytes of data were collected per week and 1.7 terabytes overall (Wielhouwer, 2005; Anastasiou, 2006; Ceranowicz & Torpey, 2004; Williams & Smith, 2007).

Deadly riots sparked by alleged electoral manipulation are not new. In 2007, the results of a presidential election in Kenya triggered protests that resulted in rioting that took place over a period of two months resulting in 1100 deaths. Ten years later, at least 37 people were killed in the protests that followed yet another presidential election in the country. The Kibera settlement located within the capital of Nairobi is the largest slum in Africa and frequently the location of the most severe rioting. (Pires & Crooks, 2016) studied the emergence of riots using the 2007 events by creating a comprehensive agent-based model that incorporates physical, emotional, cognitive and social elements. Pattern of life and daily activity scheduling was one of the three submodels created "*in order to capture the full spectrum of behaviors that … leads to the emergence of riots*".

These are but two examples of the usefulness of pattern of life simulation as a supporting element in the broader context of urban military operations, civil security, and applications such as urban planning (Qin et al., 2019) and disease spread (Zhang et al., 2016); there are no doubt many more.

Over the past year we have been prototyping a system that allows us to simulate large-scale pattern of life supporting real time, human in the loop operations for concept development, course of action analysis, and training. Specifically, we are building a system that not only simulates pattern of life at a city and country level but allows for visualization and interaction in an immersive, 3D environment from space down to street level. Our novel approach combines a population activity model, a high-fidelity crowd simulation, a scalability framework, and a game engine. The system can simulate populations in the millions updated at 10 Hz with dense crowds of up to 100,000 people simultaneously visible in a scene rendered at 60 Hz.

This paper is organized around two use cases that have been demonstrated and the four major technology elements that support those use cases. After a brief review of related work, each use case is described at a high level followed by detailed descriptions of each of the technology elements. Comparisons with other approaches and lessons learned are shared throughout the paper followed by a conclusion and recommendations.

RELATED WORK

A review of the published literature revealed three broad categories of simulations related to pattern of life and applications in real time environments. They are:

1. Non-real time analytic models that can simulate tens of millions of individuals. Visualization is usually in the form of summary information showing hot spots and migration patterns.
2. Real time models that can simulate tens or hundreds of thousands of individuals operating at time steps not generally associated with interactive, human in the loop requirements. Visualization is usually in the form of 2D maps with dots denoting locations of individuals.
3. Real-time models simulating limited numbers of individuals in an interactive, 3D environment. The synthetic environment ranges from the most basic (simple obstacles) to the highly detailed (video games).

An example of a category 1 model is described by (Richey & Mostowsky, 2020). It is a large-scale agent based model used to study refugee migrations. Their model was run with up to 25 million agents and the focus of the paper was on using high-performance computing paradigms to speed up processing. While they were successful in improving run time performance, it is not a real-time model, which is a core requirement for interactivity.

Category 2 models provide real-time capability and scalability, typically using simulation update rates on the order of 0.1 to 1 Hz. In our experience, a user is not able to realistically interact with a crowd at this low update rate.

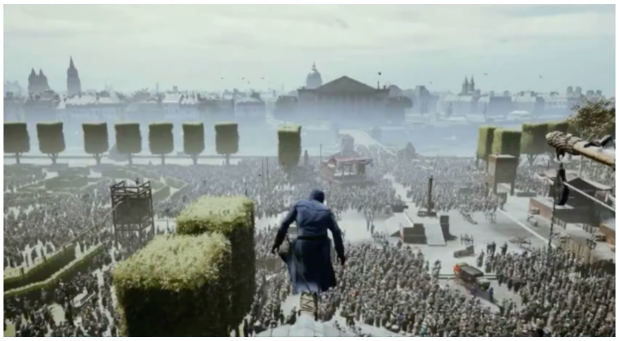


Figure 1. Large crowd scene in the video game Assassin's Creed Unity

Commercial games and crowd simulation tools are examples of category 3 models. The commercial game Assassin’s Creed has some of the most sophisticated and large-scale crowd modeling in order to support its gameplay narrative. The *Unity* installment of the franchise has a crowd scene of 10,000 (see Figure 1), the technical approach for which is described by (Cournoyer, 2015). Some of the optimizations, especially from a visualization perspective, can be leveraged for simulation; others, like dramatically simplified crowd movement and behavior would lead to unrealistic and incorrect evacuation scenarios even if visually this is not noticeable in the game. A thorough treatment of crowd rendering techniques is available in (Beacco et al., 2016).

USE CASES AND ARCHITECTURE

The large-scale pattern of life simulation solution architecture was initially designed and developed to support two demonstrators: a civil use case demonstrator and a military use case demonstrator.

Civil Use Case

The civil use case is centered on the city of London, UK and designed to exercise the capabilities of the solution rather than present a realistic scenario. The use case is as follows:

A cyberattack on London targeting critical infrastructure – the electrical grid – causes a power blackout covering large parts of central London during working hours. The blackout forces evacuation of subway stations and office buildings resulting in large numbers of pedestrians flowing onto sidewalks and city streets. As this is occurring, a sophisticated social media disinformation campaign is in play, likely launched by the same nation-state that perpetrated the cyberattack. The objective of the disinformation campaign is to provoke frustration and anger in the population at their government’s lack of ability to provide basic services and protection. A deep fake video of the Mayor of London urging calm creates exactly the opposite effect and many Londoners, egged on by social media urging them to gather in protest, make their way to key city squares, including Trafalgar Square, creating safety and *security concerns. Police are dispatched to the area to cordon off traffic and attempt to maintain order in the face of increasing crowd presence.*

The user views the progress of the scenario in real time via security cameras in fixed locations, while simultaneously monitoring population movement, the electrical grid and social media through a web-based interface.

Civil Use Case Solution Architecture

The civil use case solution architecture (see Figure 2) contains components (shown in yellow) used to generate data for the simulation, components (shown in green) used to execute the simulation, and various databases. The components associated with the pattern of life modelling and visualization that are the focus of this paper are shown with a solid border in Figure 2, other components are shown with a dashed border.

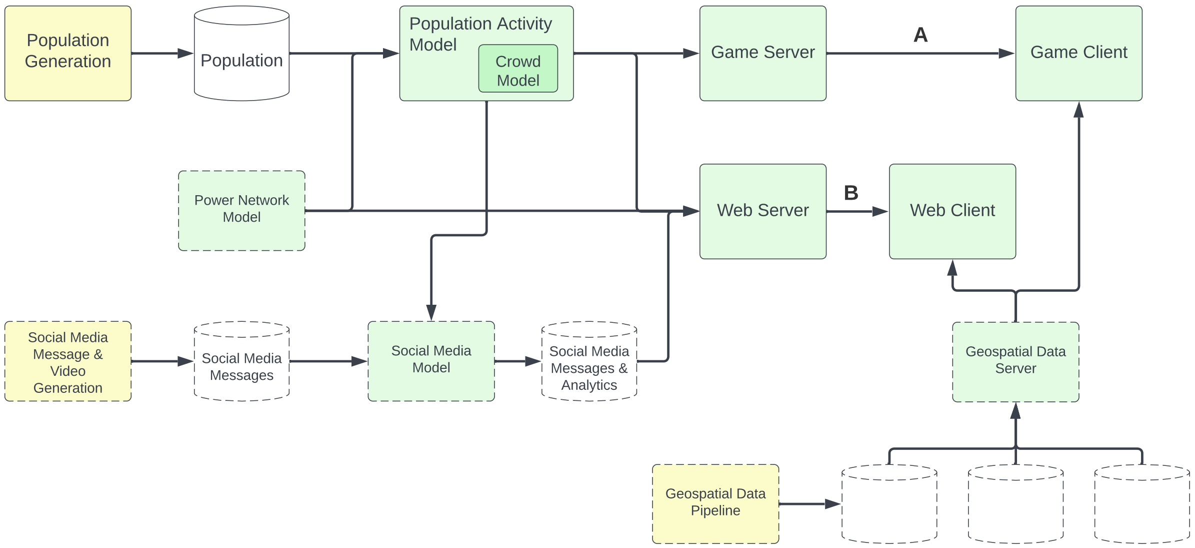


Figure 2. High level overview of the civil use case solution components

In addition to the pattern of life modelling and visualization components we describe in this paper, Figure 2 includes: a new geospatial data pipeline to create the highly detailed 3D model of central London and make it available in both web and game engine visualizations (Hobbins et al., 2022); and a social media modeling system for social media synthesis and analysis which created the trigger for the pedestrians to gather in protest, more details of which can be found in (Harris et al., 2022).

Military Use Case

The military use case is centered around the Baltics, specifically Estonia. It demonstrates how a tactical level event can be of major strategic significance:

A confrontation is underway in Ida-Viru county in Estonia focused on manipulating the ethnic composition in the area. The northeastern part of the country borders Russia and is home to a large ethnic Russian population. The port of Sillamäe is the object of a takeover by paramilitary forces using a combination of vehicles storming the entrance and a boat docking at the port. A civilian crowd has gathered, a mix of ethnic Russians and ethnic Estonians, creating an even more volatile situation. Simultaneously, a government organized gathering in Freedom Square in the capital of Tallinn to address the public and dispel weeks of social media propaganda and disinformation risks turning violent as paramilitary sympathizers approach the square.

This use case required the system to be extended in two ways: (i) the scaling up of the pattern of life simulation to handle the full population of Estonia of approximately 1.3 million people; (ii) the simulation of paramilitary forces with land, air and sea assets using a computer-generated forces (CGF) application. The singe process population activity model (PAM) with its embedded (distributed) crowd model in Figure 2 was replaced by a scalability platform that hosted a single instance of a CGF and a spatially distributed and dynamically scaled version of the PAM that included distributed crowd modeling.

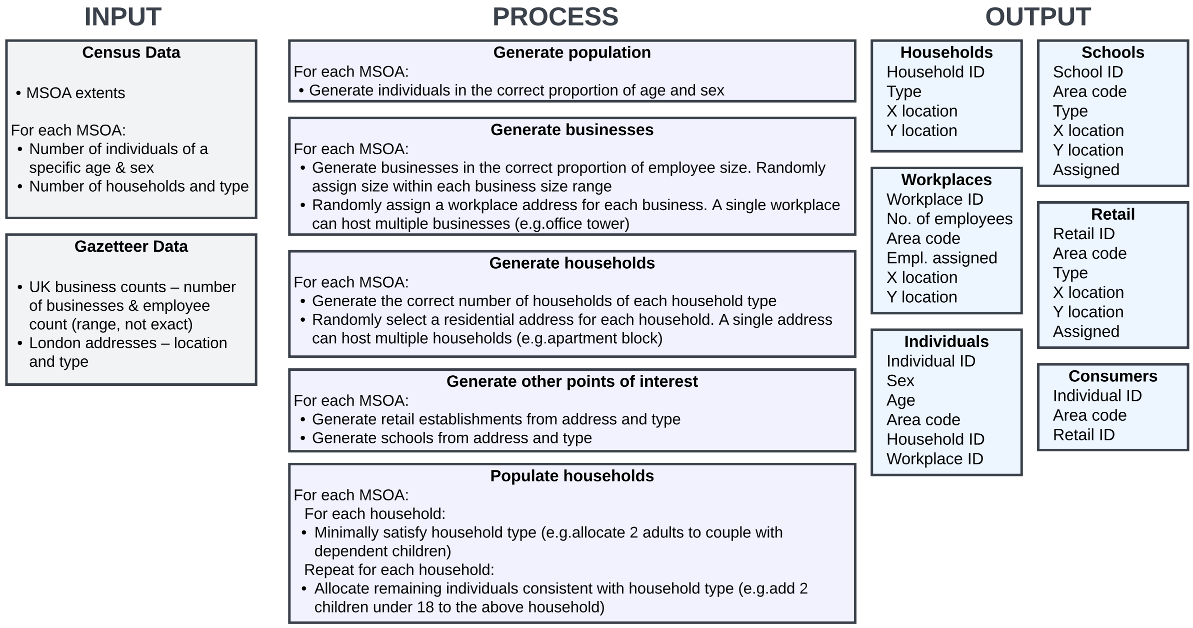
SOLUTION DEVELOPMENT AND ENABLING TECHNOLOGY

Population Generation

Our population generation was inspired by the population synthesis process of (Burger et al. 2016), starting with the referenced code although we ended up completely rewriting it for improved performance and flexibility for our use case. The overall process is shown in Figure 3.

We started with the 2011 UK census data to generate the population as a representative set of data for the population of London. When data from the most recent UK census (2021) is made available we expect the process of updating to the latest census to be straightforward and mostly automated.

UK census outputs are available at different levels (aggregate groupings) and we chose a census tract size referred to as *Middle Super Output Area* (MSOA). There are 983 MSOAs in the Greater London area (population 8.2M) leading to an average of ~8300 people in each census tract.



INPUT

Census Data

- MSOA extents

For each MSOA:
- Number of individuals of a specific age & sex
- Number of households and type

Gazetteer Data

- UK business counts – number of businesses & employee count (range, not exact)
- London addresses – location and type

PROCESS

Generate population

For each MSOA:
- Generate individuals in the correct proportion of age and sex

Generate businesses

For each MSOA:
- Generate businesses in the correct proportion of employee size. Randomly assign size within each business size range
- Randomly assign a workplace address for each business. A single workplace can host multiple businesses (e.g.office tower)

Generate households

For each MSOA:
- Generate the correct number of households of each household type
- Randomly select a residential address for each household. A single address can host multiple households (e.g.apartment block)

Generate other points of interest

For each MSOA:
- Generate retail establishments from address and type
- Generate schools from address and type

Populate households

For each MSOA:
For each household:
- Minimally satisfy household type (e.g.allocate 2 adults to couple with dependent children)

Repeat for each household:
- Allocate remaining individuals consistent with household type (e.g.add 2 children under 18 to the above household)

OUTPUT

Households
Household ID
Type
X location
Y location

Workplaces
Workplace ID
No. of employees
Area code
Empl. assigned
X location
Y location

Individuals
Individual ID
Sex
Age
Area code
Household ID
Workplace ID

Schools
School ID
Area code
Type
X location
Y location
Assigned

Retail
Retail ID
Area code
Type
X location
Y location
Assigned

Consumers
Individual ID
Area code
Retail ID

Figure 3. The population generation process, inputs, and outputs.

Population Activity Modeling

While we generated the full population of Greater London, for our civil use case demonstrator, we do not need to simulate a complex pattern of life behavior for the whole population. Our focus is the physical manifestation of the population with sufficient fidelity to provide a plausible level of population density and movement supporting analyses such as crowd buildup and management of evacuation scenarios. Specifically, we are targeting the left-hand side of the temporal scale in Figure 4, covering micro to macro over the course of minutes, hours and even days. While it is possible to run our models for longer periods of simulated time, it is not necessary for the identified use cases. Our models are designed to capture behavior in the sub-second range, nominally 10 Hz, which is sufficient to provide real time, 3D interactive visualization. Possible extensions to our models may, for example, consider the same level of data reconstruction as (Ge et al., 2016), the complex scheduling of (Smith et al. 1995) or the personality overlay of (Fehr et al. 2021).

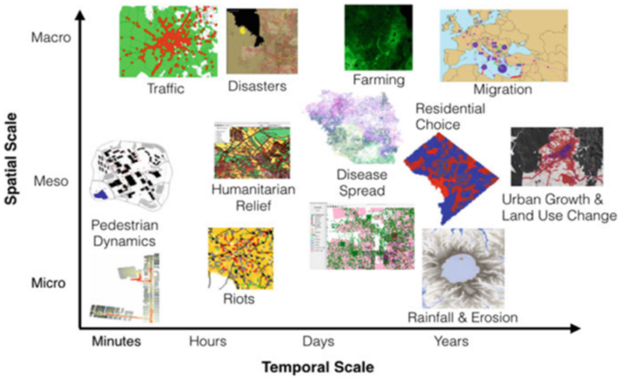


Figure 4. A comparison of geographically explicit agent-based models across spatial and temporal scales. Reproduced from (Shi 2021), page 893

(Silverman et al. 2018) provides an interesting perspective on what is a good pattern of life simulation, going so far as to create a scoring matrix based on three areas: activities, relations and behaviors. Each area is further analyzed in terms of capability into three categories: low, medium, and high. The result is a 3x3 scoring matrix shown in Table 1, with one point assigned to each cell in the table the simulation satisfies. Based on this table, our agents would score either 2 or 3, and would be characterized as "Limited".

Table 1. Scoring matrix for pattern of life representation. Reproduced from Silverman (2018)

	Low	Medium	High
Activities of Daily Life (ADLs)	• Pre-scripted ADLs following clock • Navigates on own to destinations, avoiding obstacles.	Simple rules to handle a few common ADL issues in 1 area (e.g., errands, OR combat, OR crowds)	Dynamically (re)sets priorities and (re)plans ADLs due to shifts in internal needs and external events
Nets/Social Skills	Reacts to nearest neighbors on a landscape	Connects to agents across a single-layer network	Has relationships and connections across multi-layer nets
Cognition	Can express values & reactions passed into it from a "god" source	Limited rules to react to a domain (single-layer net)	Adaptively appraises world against its own values.

A pattern of life model would not be complete without a discussion on the validity of its behavior. The use of location-based social network (LBSN) data is the most frequently used approach to validate models. (Kim et al. 2020) provides a comprehensive review of existing approaches using publicly available data sets which they refer to as "trivially small" (not covering a significant amount of the population) and sparse (insufficient number of check-ins to validate individual behavior). As a result, they create a model to generate LBSN data for research applications based on their previous work on patterns of life (Kim et al., 2019). For obvious reasons, this synthetic LBSN data could not be used to validate our pattern of life model since it would be an implicit comparison to their own pattern of life model. An ideal dataset for the validation of pattern of life would be mobile phone location data. An anonymized set of data would allow for validation of pedestrian movement and density with time of day, however that data is highly subject to privacy concerns and, in many cases, specific privacy laws such as the European General Data Protection Regulation (GDPR) apply. Without access to external validation data, we performed our pattern of life validation by inspection of pedestrian movement patterns and crowd behaviors versus the expected behavior from our original design and judged it to be appropriate for our use case.

Crowd Dynamics Modeling

Pedestrian movement and crowd behavior are key elements of our demonstration, and the selection of crowd modelling software was based on those requirements. In summary, they were:

1. Simulation of large numbers of pedestrians (up to 100,000) including dense crowds with update rates that are compatible with real-time interactive visualization.
2. Geographic coverage of large, dense urban areas, minimally 5x5 km, ideally 15x15 km or larger for more sparsely built-up areas.

3. A flexible API allowing detailed control over pedestrian behavior and the ability to integrate into our environment.
4. Scalability across multiple processes and processing nodes, required to achieve a 10 Hz update rate and a population in the millions.

We also had a requirement to render the crowd in a 3D environment and therefore investigated solutions based on game engines (see Related Work above and Game Engine Integration below). We chose a crowd simulation package which is delivered as a standalone, headless library that can be integrated into any application. Importantly, the crowd simulation is not responsible for the rendering of the pedestrians. It computes exact navigation meshes for the target environments, performs pathfinding, and outputs position, velocity and orientation of each agent at 10 Hz, sufficient to enable smooth rendering in 3D.

The crowd simulation takes a set of polygonal geometry that can represent any combination of walkable areas, non-walkable areas and obstructions, computes the intersection of these to derive the overall walkable area, and internally generates a navigation mesh that is used for pathfinding and obstacle avoidance. The creation of walkable areas for a large urban environment posed many challenges and is covered in detail in (Hobbins et al., 2022). Our initial demonstrator was based on a fixed size, 5x5 km area in central London. To satisfy our performance requirement, we spatially partition the simulated area and distribute it across multiple computers and processes. This process is dynamic, occurring at any point in the simulation, and can create arbitrary sized partitions. Our crowd simulation loads arbitrarily sized walkable areas at runtime and is able to compute the required navigation mesh and other internal data structures in a matter of a few seconds. (Van Toll et al. 2020) provides a comprehensive review of navigation meshes including performance metrics. The explicit corridor map (ECM) approach is used by the crowd simulation and provides some of the best performance of all approaches. In practice, we are able to achieve environment load times on the order of a few seconds and simulate 100,000 pedestrians updated at 10 Hz in real time in a single application instance running 4 threads on 4 cores, which satisfies the goals of our use case demonstrator. Figure 5 summarises the performance characteristics of the crowd simulation showing an almost linear increase of time with the number of agents. Performance with increasing number of threads is also nearly ideal, with execution time being halved with every doubling of the number of cores, up to about 8 cores when memory access starts to become the bottleneck.

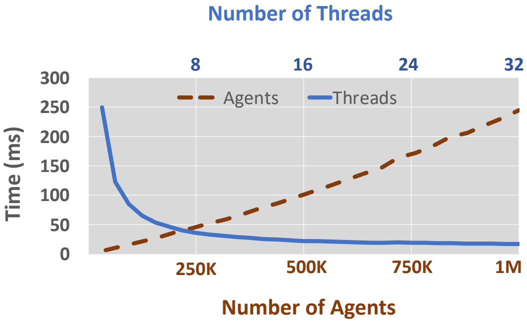


Figure 5. Performance characteristics of the crowd simulation library.

The crowd simulation allows dynamic creation and deletion of pedestrian agents, control of pedestrian size (radius), walking speed and unique origin-destination for each agent. The PAM creates a pedestrian agent at the appropriate time and location (for example, leave home to go to work) of the appropriate type (adult, child or senior), size and walking speed, set its destination, and assigns it to the crowd dynamics model. The PAM then deletes the agent when it arrives at the destination. This approach optimizes computing resources by only invoking an agent in the crowd dynamics model when a pedestrian is required. The agent remains active until they arrive at an interior destination like work or school. If the destination is an exterior plaza, such as Trafalgar Square, the agent is active even if they are stationary and are factored into the collision avoidance and path planning of other agents. The core simulation has been validated against industry benchmarks for evacuation scenarios proposed in (IMO, 2007; Ronchi et al., 2013).

Scalability Platform

As previously mentioned, for the military use case demonstrator we foresaw a need to scale our population modeling beyond a single process/computing node to simulate millions of agents. We could have developed a custom approach specifically tailored to scaling our population activity model and crowd dynamics library but instead we chose to leverage a general-purpose scalability platform. These platforms can be applied to the development of large-scale simulations, distributed cloud gaming infrastructures and even digital twins and the metaverse.

The scalability platform deployed in our solution includes three components: (i) a distributed cloud platform that runs in a public cloud infrastructure (ii) a simulation spatial distribution engine and (iii) an external data connection infrastructure. The distributed cloud platform operates within a secured Linux environment and provides distributed resource allocation, optimized networking, process and resource isolation, distributed operating system scheduling and supports dynamic scaling at runtime. Each application integrated within the scalability platform may be spatially divided. Based on simulation density, complexity and client interaction, the spatial distribution engine dynamically splits and merges virtual spaces to smaller cells and dynamically allocates computing resources as required. The external data connection infrastructure provides a mechanism for producing a real time filtered stream of data to a large number of connected clients with each receiving their own unique view. It aggregates data from one or more cells transparently to the client(s) which remains unaware of the underlying spatial partitioning. Scalability from a client perspective is achieved through multiple data server instances each with its own customizable filter logic. The spatial distribution engine and data connection infrastructure are both built on the distributed cloud platform.

Crowd Simulation Integration

Enabling models and simulations in (any) scalability platform is not a trivial task. Many existing simulations were never designed to be multi-threaded much less able to completely transfer state between two running processes without interruption or loss of data. Our previous experience in porting an existing CGF application to a different scalability platform made us wary of embarking on this task. Our decision to go ahead was based on two factors: (i) the population activity and crowd dynamics models were much smaller in scope than a full-featured CGF and (ii) a distributed version of the crowd simulation library had already been prototyped but yet not commercialized.

The work was successfully completed within a few weeks; however, we did omit one aspect related to pathfinding: developing a global pathfinding solution that allows an entity (agent) to navigate from an origin cell to a destination cell via an intermediate cell. Navigating to an adjacent cell is supported but the introduction of an intermediate cell that does not contain the destination location means that no path can be computed through that cell. A global pathfinding solution would automatically create a waypoint at the edge of the intermediate cell, essentially telling the agent “go to that point and when you get there, you will plan the rest of your path to the final destination”. We plan to implement this in future. In the meantime, we have extended the boundary region of each cell allowing pedestrians to reach destinations up to a configurable distance (5 km by default) beyond the edge of a cell.

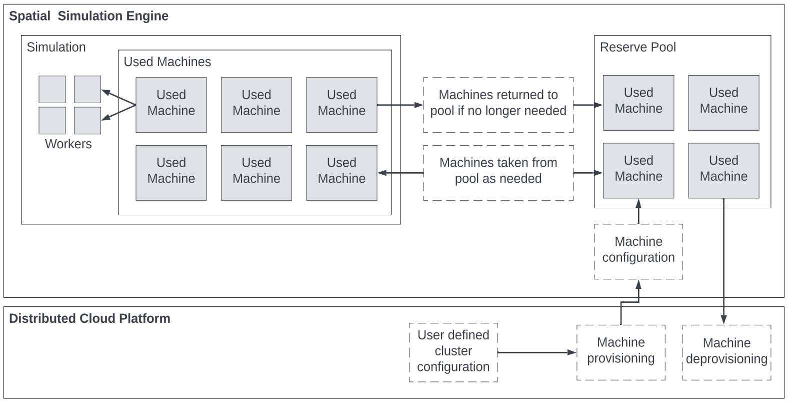


Figure 6. Overview of the dynamic scalability platform process.

The scalability platform dynamically scales based on factors that are under application control. Initial setup involves the creation of a cluster of available resources (virtual machines) that are allocated and deallocated, as shown in Figure 6. The resulting scaled population simulation can be seen in practice in Figure 7, which is a closeup view of the city of Tallinn with each pedestrian shown as a dot on the map. The thin horizontal and vertical lines denote the rectangular cells and associated spatial distribution of the simulation. Clearly visible in the figure are cells of different sizes, they are approximately related to the total number of pedestrians in each cell. At initialization, the system starts with a single cell and begins to subdivide as the PAM populates the country, a process that occurs over the first 10 seconds of the simulation. Initially, the parameter used to subdivide cells is based on the total population in each cell. Over time, as pedestrians navigate between work and home, cells may split or merge. Determining the optimal parameters must take into account processing time, memory usage and networking performance.

Computer Generated Forces Integration

The military use case required the integration of a CGF (VR-Forces 4.10 by MAK Technologies) that includes land, sea and air assets. As the total number of simulated entities is fairly low (<1000), a single instance of the CGF engine is sufficient to meet our real time performance requirements. As a result, when integrated within the scalability platform, the CGF does not need to be spatially subdivided. Integration within the scalability platform provides three advantages: (i) the CGF has full access to all the entities across all the cells of the pattern of life simulation and vice versa (ii) the CGF and pattern of life simulation are synchronized (iii) the CGF is automatically deployed and managed along with the pattern of life simulation in the computing cluster.

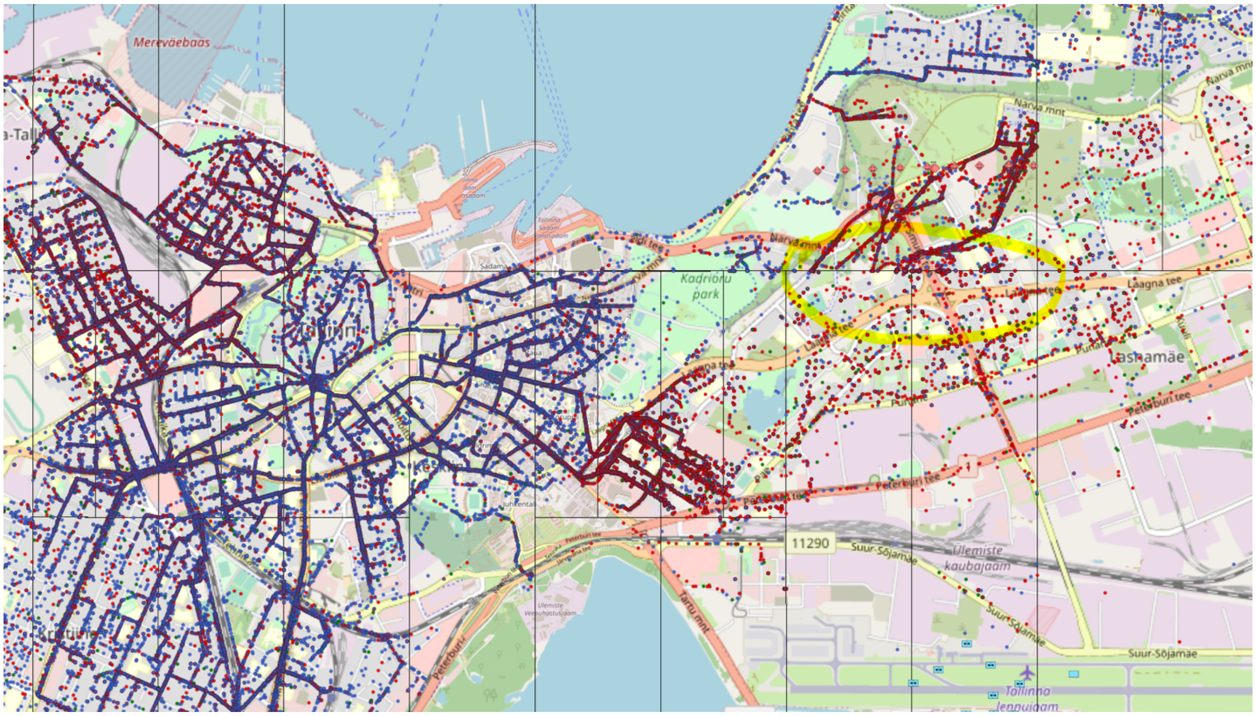


Figure 7. Closeup view of the city of Tallinn showing the dynamic spatial partitioning result. Pedestrians are shown as blue and red dots on the map.

The addition of a CGF requires a strategy to handle the interactions between pattern of life pedestrians and CGF-managed military vehicles. Pedestrians need to avoid vehicles and vehicles needed to stop when sufficient numbers of pedestrians are blocking their path. The former was easy to address as the crowd dynamics simulation supports externally controlled agents as inputs which are included in the collision avoidance and path planning algorithms. The latter was more of a challenge because we do not want to overwhelm the CGF with a crowd of thousands of pedestrians. There is also the matter of what the appropriate reaction of a military vehicle should be when facing a small number of pedestrians walking (does it keep going and allow the pedestrians to move over?) or a large number that it can't expect to push through. We determined that further analysis is required and as of this writing, the transfer of pedestrian crowds to the CGF has not been implemented.

Visualization Client Integration and Networking

There are two main data connections between the cluster running the pattern of life simulation and CGF and the two visualization methods in the system. The first drives a God's eye view of every entity in the simulation on a web-based map viewer similar to Figure 7. The second is a filtered dataset based on the current viewpoint in the game engine (figure 8).

The God's eye view presented challenges based on the sheer volume of data, the web viewer's ability to process and render the data, and the requirement for a responsive user interface. To solve these issues, we developed a web server to receive all the entity updates on a regular basis along with entity metadata on a request basis. The server connected to the web client (connection A in figure 2) filters the data based on viewpoint, zoom level and degree of overlap of symbols and icons, to avoid drawing the same pixels repeatedly.

The game engine viewpoint is supported directly by the networking component of the scalability platform. At every frame, the location and orientation of the viewpoint is sent to the simulation cluster and only the entities visible within a user-defined radius are sent to the client. In a dense crowd scene, the total number of entities may be in the tens of thousands.

Game Engine

As shown in Figure 2, a game engine server and client (Unreal 4.26 by Epic Games) provides a real-time, immersive 3D environment to visualize our pattern of life simulation. Initially, we tried to use standard features of the game engine to create large numbers of characters (pedestrians) that could be individually controlled. Testing showed that we could only support a few hundred characters while maintaining adequate frame rates for interactive visualization. The tradeoff against this performance penalty was the high level of fidelity and control available, typical for first person shooter type applications.

One approach to mitigate the performance limitations is to use viewpoint-based area-of-interest filtering to limit the number of characters required to be simultaneously displayed within the field of view. This approach has at least two significant drawbacks however: (1) our requirement is to model large crowds numbering in the tens of thousands seen simultaneously using simulated cameras mounted on low-flying drones or security cameras pointed at the street or large open spaces; (2) we must support multiple simultaneous, independent viewpoints in which the views are correlated. Filtering out characters from the scene in one view risks having them present in another view, especially in crowded scenes.

Satisfying these two requirements proved to be a significant challenge. We tested several third-party plugins (extensions) to the game engine but they all suffered from either one or both of the above issues. We also attempted to use the game engine's native particle system, which was much more scalable, but encountered limitations in the ability to control each particle independently.

The solution we eventually developed relied on the use of individually controlled, hierarchical static meshes (HISM). The term hierarchical refers to the ability to define different levels of detail versus simple instanced static meshes. This was an important performance improvement that allowed us to reduce the level of detail for characters that were far from the viewpoint. We also took advantage of several other features of HISM's including that allowed us to customize pedestrians on an individual basis. A single game object was used to process all the pedestrian data received from the crowd simulation at every simulation frame, assigning position, orientation and animation based on velocity coupled with prediction and smoothing. The final performance optimization we implemented was to precompute the animations (a process known as "baking") thus avoiding the computation of vertex positions for each static mesh at each frame. This process limits animations greatly compared to regular game animations, but it does allow for full body animation and proved sufficient for our needs. This technique is commonly used for background crowds in sport and racing games for example. The combination of all these elements allowed us to meet our objectives with crowds steadily filling the square, as can be seen in figure 8.

Networking

A client-server approach was adopted for our game engine architecture, to allow multiple users to connect to a shared simulation of the synthetic environment. The game engine server was run in a headless or non-rendering mode and the pedestrian entities were synchronized with the pattern of life simulation. In the case of the London demonstration, a message queue was used to populate the game server with the full 100,000 entities. In the Estonia demonstration, our scalability framework's client network code was used to synchronize only the entities visible within a programmable radius around each of the viewpoints of the connected clients. That left us with the choice of approach to synchronize the game server with each of the connected clients (connection B in figure 2). Our first attempts were to use the existing game engine networking capability ("replication"). Not surprisingly, it was never designed to handle up to 100,000 entities at 10 Hz and repeatedly dropped packets, even after optimization of the amount and structure of the data.

The experience with the game engine's native networking and visualization taught us an important lesson: there may be situations where the out of the box capability, even that of an advanced game engine, may not meet the specific simulation or visualization requirements of your application. In those situations, it's important to (i) realize that you have reached a limit in what the product provides and (ii) be prepared to work around those limitations either with your own development or third-party capabilities. In our case, we took the major step of replacing the existing client-server networking with our own system that we had used previously in real-time visualization applications.

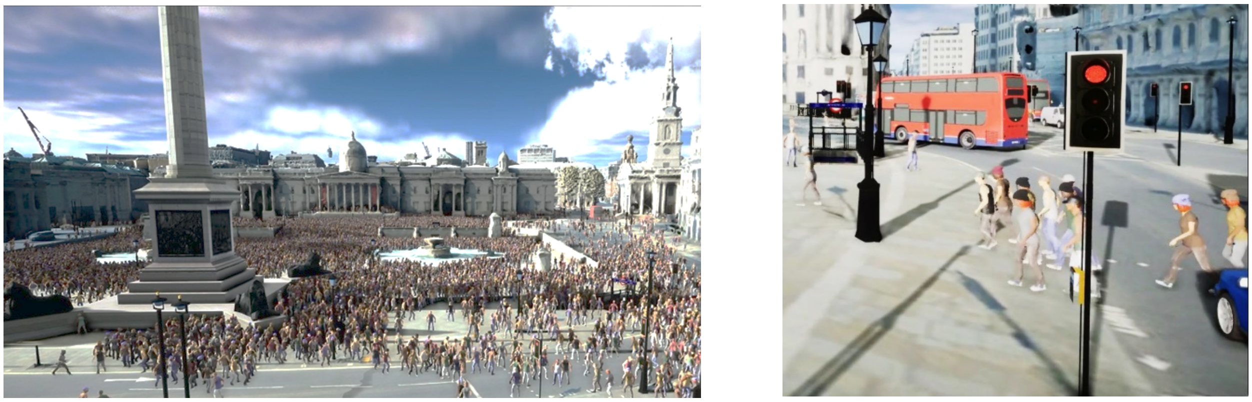


Figure 8. Visualization of a crowd in Trafalgar Square, London (left) and a closeup of pedestrians showing the level of detail in the characters (right).

CONCLUSIONS AND FUTURE WORK

We developed a proof of concept that allowed us to test the capabilities and limitations of a number of technologies when integrated together. Our objective was to create a synthetic environment at scale that would support real-time, interactive, simulation and visualization. This proof of concept was developed with two specific use cases, civilian and military. We developed a pattern of life model to provide the scalable simulation. The model included a crowd simulation and was integrated as both a standalone simulation and inside a scalability framework to further test the performance limits. The entire system was deployed in a public cloud infrastructure and tested in dense urban and country-sized environments. Visualization was via a web-based map-centric interface and an immersive game engine.

Our experience has shown that while it is possible to achieve scale using modern frameworks, simulations, game engines and the cloud, none of these elements can be taken for granted, either individually or collectively. The crowd simulation was adapted to run with the spatial distribution of the scalability framework but work remains to enable global path planning. The game engine provides compelling 3D interactive visualization but virtually the entire data and rendering pipeline had to be modified to display the number of pedestrians in our simulation. The scalability framework provides networking capability but our “God’s Eye View” requirement for millions of entities viewable in a web browser required a custom network protocol and filtering implementation. We have yet to resolve the optimal approach for the interaction between crowds and military entities, given that the CGF was not designed to handle tens of thousands of pedestrians.

Commercial and open-source software is constantly evolving. In the short span of time since this project began, there has been a major new release of our game engine and two of our authors recently tested visualizations of hundreds of thousands of pedestrians in real time. There are some promising approaches using messaging frameworks as the communication backbone for disparate simulations, potentially replacing some of our custom developed software. When fidelity and performance are critical however, as with real-time, interactive simulation and visualization, there will always be a need for traditional integration activities.

ACKNOWLEDGEMENTS

This project would not have been possible without the innovation and hard work of an international group of people across four countries - too numerous to mention here, you know who you aware and we are grateful for your contribution.

REFERENCES

Al Jazeera. At Least 37 People Were Killed in Election Violence. *https://www.aljazeera.com/news/2017/10/9/at-least-37-people-were-killed-in-election-violence*, retrieved 13 June, 2022.

Anastasiou, Alexander B. (2006). Modeling Urban Warfare: Joint Semi-Automated Forces in Urban Resolve. *Air Force Institute of Technology Theses and Dissertations, 3276.*

Beacco, A., Pelechano, N., and Andujar, C. A survey of real-time crowd rendering. *Computer Graphics Forum*, December 2016, vol. 35 number 8, p. 32-50.

Burger, Annetta, Oz, Talha, Crooks, Andrew & Kennedy, William G. (2017). Generation of Realistic Mega-City Populations and Social Networks for Agent-Based Modeling. *CSSSA '17, October 2017, Santa Fe, New Mexico*.

Ceranowicz, Andy & Torpey, Mark (2004). Adapting to Urban Warfare. *Interservice/Industry Training, Simulation and Education Conference, 1554.*

Cerri, Tony, Laster, Nicole, Hernandez, Alejandro, Hall, Steven B., Sleevi, Neil F. & Johnson, Andrew (2017). Simulation of Non-Combatant Population Movement in the Battlespace. *Interservice/Industry Training, Simulation and Education Conference, 17214.*

Cournoyer, Francois. Massive Crowd on Assassin's Creed Unity: AI Recycling. https://www.gdcvault.com/play/1022141/Massive-Crowd-on-Assassin-s, retrieved 15 June, 2022.

Fehr, Ashley, Stoffa, Joseph A., Newton, Jared & White, Josh (2021). Growing People: Generating Realistic Populations and Explainable, Goal Directed Behavior. *Interservice/Industry Training, Simulation and Education Conference, 21253.*

Harris, Evan P., Kaur, Jaspreet, Chaouachi, Maher, Durocher, Martin, Tiwari, Rakesh, Emirov, Alex and Giannias, Nick. Scoial Media Synthesis using AI for Decision Support. *Interservice/Industry Training, Simulation and Education Conference, 22175.*

Hobbins, Joanna D., Merrick, Simon, Giannias, Nick, Kopan, Melisa, Lilley, Sean, Mohammed, Shehzan and Coleman, Ralph. Geospatial Data Pipelines for Urban Digital Twin Applications. *Interservice/Industry Training, Simulation and Education Conference, 22180.*

International Maritime Organization (IMO). Guidelines for evacuation analyses for new and existing passenger ships. *msc/circ.1238*, 2007.

Islam, Samiul, Gandhi, Dhruv, Elarde, Justin, Anderson, Taylor, Roess, Amira, Leslie, Timothy F., Kavak, Hamdi & Zufle, Andreas (2021). Spatiotemporal Prediction of Foot Traffic. *5th ACM SIGSPATIAL International Workshop on Location-Based Recommendations, Geosocial Networks, and Geoadvertising, November 2–5, 2021, Beijing, China.*

Jiang, Na, Burger, Anenetta, Crooks, Andrew T. & Kennedy, William G. (2020). Integrating Social Networks into Large-scale Urban Simulation for Disaster Responses. *3rd ACM SIGSPATIAL International Workshop on Geospatial Simulation, Seattle, WA.*

Kerbusch, Philip, Smelik, Ruben, Smit, Selmar & Kuijper, Frido (2012). Easy Pattern of Life Generation using Physical and Human Terrain. *Interservice/Industry Training, Simulation and Education Conference, 12180.*

Kim, Joon-Seok, Crooks, Andrew, Kavak, Hamdi, Pfoser, Dieter, Züfle, Andreas, Manzoor & Wenk, Carola (2019). Simulating Urban Patterns of Life: A Geo-Social Data Generation Framework. *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, November 5-8, 2019, Chicago, IL, ACM, New York, NY, USA.*

Kim, Joon-Seok, Jin, Hyunjee, Kavak, Hamdi, Rouly, Ovi Chris, Crooks, Andrew, Pfoser, Dieter, Wenk, Carola & Züfle, Andreas (2020). Location-Based Social Network Data Generation Based on Patterns of Life. MDM.

Martin, David, Gale, Christopher, Cockings, Samantha & Harfoot, Andrew (2018). Origin-destination geodemographics for analysis of travel to work flows. *Computers, Environment and Urban Systems, 67*, 68-69.

Mikhailov, Sergei and Kashevnik, Alexey (2020). Tourist Behaviour Analysis Based on Digital Patter of Life – An Approach and Case Study. *Future Internet* **2020**, 12(10), 165.

Monday, Paul (2007). Architecture of the Counter Insurgency Experiment. *Interservice/Industry Training, Simulation and Education Conference, 7261.*

Pires, Bianica & Crooks, Andrew T. (2016). Modeling the emergence of riots: A geosimulation approach. *Computers, Environment and Urban Systems 61, 66-80.*

Pouke, Matti, Goncalves, Jorge, Ferreira, Denzil & Kostakos, Vassilis (2016). Practical simulation of virtual crowds using points of interest. *Computers, Environment and Urban Systems, 57*, 118-129.

Qin, Kun, Xu, Yuanquan, Kang, Chaogui, Sobolevsky, Stanislav and Kwan, Mei-Po (2019) Modeling Spatio-Temporal Evolution of Urban Crowd Flows. *International Journal of Geo-Information*, (2019), 8, 570.

Richey, Melonie K. & Mostowsky, Zachary (2020). Leveraging HPC Techniques for Large-Scale Agent-Based Models in Python. *Interservice/Industry Training, Simulation and Education Conference, 20210.*

Ronchi, E, Kuligowski, E.D., Reneke, P.A., Peacock, R.D. and Nilsson D.. The process of verification and validation of building fire evacuation models. *US Department of Commerce, National Institute of Standards and Technology*, 2013.

Shi, Wenzhong, Goodchild, Michael F., Batty, Michael, Kwan, Mei-Po & Zhang, Anshu (2021). *Urban Informatics*, Singapore: Springer.

Silverman, Barry G., Bharathy, Gnana K. & Weyer, Nathan (2018). What is a Good Pattern of Life Model? Guidance for Simulations. *Simulation, August 2018.*

Smolak, Kamil, Rohm, Witold, Knop, Krzysztof & Siła-Nowicka, Katarzyna (2020). Population mobility modelling for mobility data simulation. *Computers, Environment and Urban Systems, 84*, 101256.

Van Toll, Wouter, Triesscheijn, Roy, Kallmann, Marcelo, Oliva, Ramon, Pelechano, Nuria, Pettre, Julien & Geraerts, Roland (2020). Comparing navigation meshes: Theoretical analysis and practical metrics. *Computers and Graphics 91*, 52-82.

Wielhouwer, Peter W (2005). Preparing for Future Joint Urban Operations: The Role of Simulations and the URBAN RESOLVE Experiment. *Small Wars Journal, July 2005*, http://www.smallwarsjournal.com/documents/urbanresolve.pdf

Williams, Richard & Smith, Shane J. (2007). UR2015: Technical Integration Lessons Learned. *Interservice/Industry Training, Simulation and Education Conference, 7259.*

Zhang, Mingxin, Verbraeck, Alexander, Meng, Rongqing, Chen, Bin, Qiu, Xiaogang (2016). Modeling Spatial Contacts for Epidemic Prediction in a Large-scale Artificial City. *Journal of Artificial Societies and Social Simulation 19(4) 3.*