Semantic crowds

Nick Kraayenbrink*, Jassin Kessing, Tim Tutenel, Gerwin de Haan, Rafael Bidarra**

Computer Graphics and Visualization Group, Delft University of Technology, Delft, The Netherlands

Abstract

Recent advances in crowd simulation techniques have led to increasingly realistic agent and group behavior. As many crowd simulation solutions typically target only specific types of environments and scenarios, numerous special-purpose methods and systems have emerged that are unsuitable for other contexts. Solving this situation demands a higher-level approach that takes re-use and re-configuration of crowds as a priority, for adequate application in a broad variety of scenarios, virtual environments and interaction with the entities present in that environment. In this article, we propose *semantic crowds*, a novel approach that allows one to specify and re-use the same crowds for virtually any environment, and have them use the objects available in it in a meaningful manner. To have the agents autonomously interact within any virtual world, we avoid in them explicit object-related information. Instead, this knowledge is stored in the objects themselves, which can then be queried, according to an agent's needs. To facilitate creating such crowds, we developed an interactive crowd editor that provides high-level editing parameters for defining crowd templates. We illustrate the flexibility of semantic crowds by means of three cases, in which we let the same crowd populate quite differently configured airport terminal environments. These examples also highlight that this modular approach easily combines with your custom implementations of agent behavior model and/or motion planner.

Keywords: crowd simulation, crowd specification, reusable crowds, agent behavior, semantic virtual worlds

1. Introduction

Real-time crowd simulation is becoming increasingly important for a large variety of applications, such as games and simulation systems for training and education. For example, entertainment and serious games can expect a fair amount of criticism, if they feature rather unpopulated and empty cities and other environments. Recent advances in crowd simulation techniques have enabled more and more realistic agent and group behavior deploying, among other things, elaborate behavioral models, complex motion planning algorithms and impressive physics systems.

However, these improvements come at a price: most crowd simulation solutions are either too generic and high-level, or they are pretty much 'hard-wired' and customized, typically targeting only a few specific types of environments and scenarios. The former likely leads to repetitive, 'canned behavior', the latter to an ad-hoc, specialized crowd, only suited for a very narrow application domain. In both cases, we are faced with a large variety of special-purpose methods and systems that are hard to re-configure and re-use in other contexts.

Avoiding these drawbacks requires a totally different approach, one that primarily supports and promotes the specification, configuration and re-use of crowds. In this article we propose one such approach, called *semantic crowds*, that has two main characteristics: (i) it allows one to specify once, and re-use with minimal modifications, the same crowds in virtually any environment; and (ii) in each environment, the agents in these crowds are able to use whatever objects are available in a meaningful manner.

We argue that, in order to successfully apply such specifications, or *crowd profiles*, in a broad variety of scenarios and virtual environments, the latter have to contain more than just pure geometry: they have to be extended with *semantics*. In the fields of linguistics, computer science and psychology, semantics is the study of meaning in communication. When focusing on virtual environments, we call semantics 'all information conveying the meaning of a virtual world and its entities' [1]. In this definition, entities encompass everything that can exist inside a world. Semantic virtual worlds, thus, consist of entities that 'know' not only about their shape and materials, but also about their attributes, roles, functions, services, etc. Such entities have the potential to strongly improve the quality and variety of interactions with an avatar, be it of a player or of any agent from a crowd [2].

In order to have the agents in a semantic crowd plausibly and autonomously interact within any virtual world, we minimize the information in the agents relative to what objects do and how to use them. Instead, that information is stored in the objects themselves, which the agents can then purposefully query, based on what they want to achieve.

For the specification of crowds and agent behavior, we integrate several concepts, results and methods known from the literature. The novel contribution of our work lies mainly in that the same crowd, once specified, can be applied to whatever

*Current affiliation: Google inc., Mountain View, CA, USA

**Corresponding author. E-mail address: r.bidarra@tudelft.nl

different virtual environments, while its agents always exhibit plausible yet very different behaviors without having to be 'rewired' themselves. In other words, we are making such crowds effectively reusable.

This article is structured as follows. We first survey previous work related to the specification of both virtual environments and crowds (Section 2). Next we elaborate on the essential aspects of semantics, particularly on its contribution to crowd and environment definition (Section 3). We then introduce the main concepts of the semantic crowd model, both in its demographics and in its agents (Section 4), and describe the main features of our prototype framework, including an interactive crowd editor that provides high-level editing parameters for defining crowd profiles (Section 5). Finally, we illustrate the potential and flexibility of semantic crowds by means of three demonstration cases (Section 6).

2. Related Work

This section surveys a selection of previous work regarding the representation and generation of virtual environments and crowds.

2.1. Virtual Environments

Beyond geometric data, virtual environments may also contain information about objects, which actions these can perform or how they can be used by agents. A Virtual Environment (VE) presenting this kind of information is normally referenced as an Intelligent Virtual Environment [3], Informed Environment [4] or Semantic Virtual Environment [5].

Thomas and Donikian [6] propose a model of virtual environments using structures suitable for behavioral animations. Using this knowledge, autonomous virtual actors can behave like pedestrians or car drivers in an urban environment.

Through the concept of synoptic objects [7], Badawi and Donikian describe an informed environment using objects which contain a synopsis of interactions that they can be subjected to. Using a set of seven basic actions, the objects can describe the interaction process to any agent using them. Smart objects [8] were a successful proposal for adding semantics to virtual objects, dealing with many of the possible user interactions in a VE. They were primarily devised for manipulation, animation, and planning purposes. Gutierrez, Vexo and Thalmann [9] present an object representation based on the semantics and functionality of interactive digital items within a VE. Each object participating in a VE application is not only a 3D shape, but a dynamic entity with multiple visual representations and functionalities. In this way, it is possible to dynamically scale and adapt the object's geometry and functions to different scenarios.

Research in artificial intelligence proposed the notion of ontologies, to overcome the lack of shareable and reusable knowledge bases [10]. Ontologies define the meaning of objects and the relations between them. Using this concept to define a world knowledge base, Grimaldo et al. [11] propose a semantics-based framework for simulation of groups of intelligent agents. Agents can use the information provided to enhance both agent–object and agent–agent interactions. An object taxonomy is used to classify the interactive objects and to specify their properties. Ontologies are also used to define social relations among agents in order to display socially acceptable decisions.

A semantic model for representing multi-layered complex environments is presented by Jiang et al. [12] and is composed of three different levels: a geometric level, a semantic level, and an application level. The geometric level contains a 3D model of the environment that is used for visualization and to extract semantic information that will feed the next semantic level. This semantic level comprehends a structure map, a topologic map and a height map which are used to identify or query semantic information of the environment. The final layer (application level) is responsible for providing efficient interaction between pedestrians and the environment. The crowd model used in their work is a modified version of the original model proposed by Treuille et al. [13].

2.2. Virtual Crowds

Virtual groups have been studied since the early days of behavioral animation. There are two main approaches used in the literature: agent-based models and force-based models. The first approach is based on modeling of virtual agents, which interact among themselves and have some level of autonomy and individuality. The second approach –force-based models– provides more global control and handles high density crowds. These methods normally yield crowd simulations that resemble particles rather than human animation. In this subsection, we survey existing approaches for crowd models.

Two seminal papers are examples of agents-based model: Reynolds [14] simulated flocks of bird-like entities, or boids, obtaining realistic animations by using only simple local rules; Tu and Terzopoulos [15] created groups of artificial fishes endowed with synthetic vision and environmental perception, which control their behavior. In both papers, only small groups were simulated, and high density crowds were not treated. Another example of agents-based models was presented by Musse and Thalmann [16], who proposed an approach with hierarchically structured crowds having different levels of autonomy. In their model, the behavior is based on a set of rules dealing with the information contained in groups of individuals, e.g. individual and group knowledge about the world, individual and group states as well as their intentions. Ulicny and Thalmann [17] proposed a model for crowd simulation based on combination of rules and Finite State Machines for controlling agents' behaviors in a multi-layer approach. The model proposed by Farenc et al. [18] describes the coordination between smart objects, intelligent environments and the virtual crowd. This work describes crowds of virtual people which were controlled by the objects in order to act, and could read the environment in order to evolve in the simulation. In this case, the virtual agents were less autonomous and more controlled by the environment. Not focused on crowds, but on groups of a few virtual agents, the model proposed by Abaci et

al. [19] describe an extended version of smart objects for AI and planning purposes.

More recently, Kapadia et al. [20] proposed a multi-actor simulation in which actions, modifiers, constraints and behaviors can be expressed in a scripting language. These are used in a behavior state machine for the actors. To actions, cost effects can be defined and by expressing a minimize or maximize on particular costs in a behavior the most desired actions are chosen. Allbeck [21] introduced the CAROSA tool for authoring NPCs. The framework contains an 'Actionary' which includes a database of parameterized actions and objects. Both can be defined independent of a particular application or scenario. These actions and objects are built up hierarchically. A scheduler can be used to link particular actions to a time and place. Stocker et al. [22] introduce an approach where dynamically occuring events describe what types of actions their virtual humans can execute. Additionally they introduced the idea of agent priming to further restrict and simplify action choice. Since contexts can be described in which events occur, a contextual behavior is achieved.

The second class of crowds in literature is concerned with force-based models. Bouvier and collaborators [23] have studied crowd movements with adapted particle systems. In their work, the motion of people is modeled with charges and decision fields, which were based on the interactions between electric charges and electric fields. Helbing and collaborators [24, 25] presented a model to simulate groups of people in panic situations. Helbing's model is physically-based, and crowd movement is determined by attraction and repulsion forces between simulated agents and the environment. Both models are examples of force-based methods that deal with high density crowds, where virtual agents are homogeneously treated as particles. Braun et al. [26] extended this model by adding individuality to agents and including the concept of groups, focusing on panic situations. Despite the interesting results, this latter model does not present facilities to provide individual control. In addition, physically-based models as proposed by Bouvier et al. [23], Helbing and Molnar [24] lack simplicity and robustness, and any changes in resulting behavior should be addressed by changes in the equation that represents the global control. Treuille et al. [13] proposed a real-time crowd model based on continuum dynamics. In their model, a dynamic potential field integrates global navigation with moving obstacles, by modeling the spatial motion of agents as a function of crowd density. This model presents very interesting results, but it also lacks local control and simplicity: changes in the model are hard to achieve because of its complexity.

Finally, some hybrid methods describe models to combine local and global control, e.g. Pelechano et al. [27] described a crowd model by applying a combination of psychological and geometric rules, with social and physical forces. This model is focused on panic situations, and many parameters should be calibrated in order to exhibit expected behaviors.

In summary, some of the simulation methods presented in literature are classified as agent-based models. While this popular approach brings some behavioral advantages in obtained results, it lacks of control flexibility in high dense crowds. On the other hand, the model described in [13] is one example of a force-based model. It presents interesting assumptions dealing with individuals and groups with common goals. However, the downside is the complexity of the method and the lack of individual control.

The semantic crowds approach we introduce here builds upon various concepts presented above. Actions, described in a semantic library for the purpose of gameplay specification as well as interaction between entities, have effects on entities' attributes and state. By creating ambitions and goals that describe the preferred target state or target value of these attributes, we can query the semantically enriched game worlds to find suitable ways of reaching these goals. The main difference with existing approaches is that we strongly limit the information that needs to be specifically defined for the agents behavior. For this, we re-use gameplay and interaction specification from our semantic database and therefore, when defining agent behavior, we can focus on the desired state of these agents. A variety of ways in which the semantically enriched game worlds can provide solutions to reach this state is available for use in the semantic database. In most other frameworks available, specific scripts, parameters or functions need to be defined not only for the behavior of the agents but also for the actions, effects or events that are used by the agents.

3. Semantics

In this section, we elaborate on the semantics required for both virtual environments (3.1) and agents (3.2).

3.1. Semantics for Virtual Environments

Virtual worlds are populated with objects that are described by various *properties*, by an *appearance* and, increasingly often, by some *physics* as well. In addition, an object can exhibit some basic behavior, usually defined by scripts that, for example, can prescribe how to move, what animation to trigger, or how to interact with other objects. We developed a semantic model that provides a shared knowledge base among all these components (presentation, physics and behavior).

We define a *semantic game world* as 'a virtual environment that is populated with entities (either objects or agents) that are enriched with semantics' [28]. This semantics facilitates that all information in the world is kept consistent. Not only is this valid for the external appearance of an entity, but also for its internal behavior, and its relations with other entities. This opens up new possibilities to reason about the world. Virtual world designers, for example, can use semantics to create the world, aided by information about where entities should be located. On the other hand, intelligent agents can plan their actions in run-time based on the entities they can use in the virtual world, thus more closely resembling real world behavior. In a semantic game world, entities change dynamically in sensible ways one expects them to, while remaining coherent with their natural behavior.

When more entities behave like in real life, both players and agents can achieve their goals in multiple ways. For example,

in order to satisfy their hunger, agents might eat a snack, buy food in a nearby shop, or order a meal in a restaurant, assuming these entities were specified in a proper way.

The main concepts in our semantic world model are *entities* (e.g. substances, living entities, or spaces), of which characteristics are expressed in *attributes*, e.g. the hunger level of a human or the price of some merchandise. Among the physical entities we distinguish *spaces* from *physical objects*. A space is a bounded region without a direct physical presence, e.g. a parking lot. In contrast, physical objects do have a physical presence and consist of a particular type of *matter*, e.g. a table consisting of wood. Attributes from that matter, e.g. the flammability, are inherited, resulting in more specialized semantics that can be used for physics.

Many relationships can be expressed between these entities, e.g. ownership (a traveller owns his luggage). In addition, spatial relationships (e.g. cupboard is typically placed against a wall) can be used to automatically generate floor plans [29], room layouts [30] and consistent buildings [31].

The generic behavior of entities in our model is expressed through the concept of *services*, defining 'the capacity of entities to perform particular actions' [2]. Basic components of an action are its *requirements* and its *effects*. The effects determine what the action brings about, while the requirements determine when the action may be performed. For example, when someone inserts a coin into a vending machine, it will return a snack, which can be eaten to reduce the level of hunger.

3.2. Semantics for Agents

In order to simulate agents in a semantic game world, the concept of *desire* has been introduced in our semantic model. A desire describes the intent of an entity with regard to its state, and can be either relative or absolute. Absolute desires describe the state or the range of states the entity wishes to be in, and the set of states that satisfy these desires will not vary over time. For example, an agent may have the desire to have at least 10 coins. In contrast, relative desires describe changes that the entity wishes to apply to its state, and thus the set of states that satisfies these desires can change when the entity's state changes. For example, for an agent to 'obtain another hat', it does not matter how many hats the agent already has, only by having one more will the desire be fulfilled.

By querying their (known part of the) world, agents will be able to find actions whose effects have an influence on their current desires. From each action requirement, a new desire can be generated to satisfy it, for which actions can then be found that may help satisfy that desire. Actions may also have a reaction as one of its effects (the attempted start of another action), and, as such, agents may also include in their search actions performed by other entities.

4. A model for semantic crowds

The proposed semantic crowds approach comprises two major components: the crowd model and the agent model. The crowd model contains the global composition of a crowd in different demographics, as well as the ratio in which they are present. For each of these demographics, a semantic agent model describes their specific behavior. The next subsections will elaborate on both the crowd model and the semantic agents.

4.1. Crowd Model

The semantic crowd model consists of two independent components: the *crowd profile* and the *crowd socket*. The crowd profile defines all environment-independent aspects of the crowd. The crowd socket configures all aspects required to insert the crowd into a concrete environment.

4.1.1. Crowd Profile

In this section, we discuss the crowd profile of our model. We work from the bottom up, from one level above the agents themselves, up to the entire crowd. Note that with our definition of crowd, there can be more than one crowd acting in a single environment. We use the term *population* for the entire set of agents in an environment.

Demographic. A demographic is a group of agents that have something in common, and it has often been used before in virtual crowd definitions, such as [32]. What they have in common depends on the used agent model. For example, the demographic 'heavy smokers' may contain agents that will often stop to light a cigarette or maybe buy tobacco.

A demographic always contains a set of conditions, which constrain when an agent can belong to it, e.g. only when a market is busy enough will pickpockets appear. Other demographics may allow only a few agents to be part of it (e.g.: a soccer field may only contain 11 agents of each team). By using these conditions when authoring a crowd, 'special' agents (such as the prime minister, or an agent that will cause a ruckus) can be inserted into the environment using the same mechanic as the rest of the crowd.

Demographic Slice. A demographic slice is a description of demographics that covers at most the entire crowd. One can think of this as the distribution of agents in demographics when a cross-section of the crowd is made. Each demographic is coupled with a percentage indicating how many agents in the cross-section are part of that demographic.

A demographic slice can be seen as a specialized view of the crowd. For example, a slice may detail the distribution of agents regarding their shopping behaviors in a mall; 30% shop-a-holics, 40% shoppers for groceries, 16% impulse buyers 1% thieves, and the other 12% passers-by (having no interest in any of the shops).

Agents can be part of multiple demographics contained in the same demographic slice. However, for each slice, if each agent could exhibit at most only one of its demographics, there is a configuration of 'selected' demographics that match the distribution defined in the slice. For example, given the example slice above, another slice may also contain the demographic 'impulse buyers', this time with 30% coverage. This means

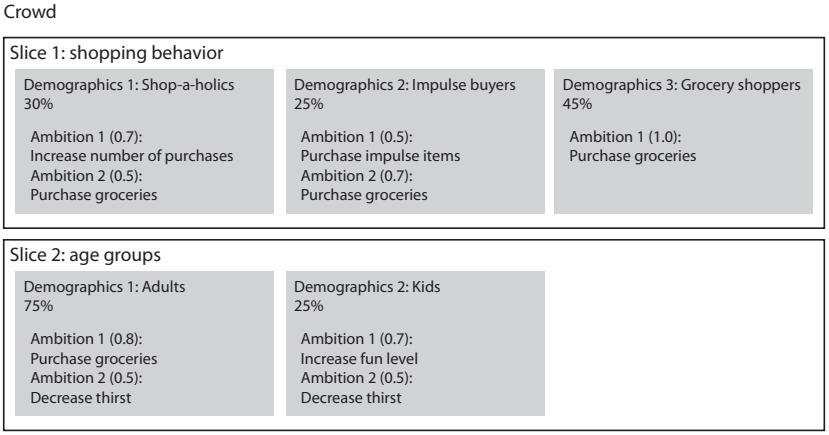


Figure 1: Example of the structure of a crowd using our agent model.

that part of the agents in the 'impulse buyers' demographic will also be in one of the other demographics contained in the slice described above.

Demographic slices may also contain conditions, which serve to limit the section of the crowd through which the cross-section slices. Such conditions make the authoring of crowds easier.

Since the demographics themselves have conditions, a demographic slice also contains an indicator on how to handle cases where the demographic is not enabled. If a demographic is disabled, it will either be treated as if it was unassigned space in the slice, or the other demographics will be 'enlarged' such that the slice covers the same percentage of the crowd.

Crowd. A crowd is a collection of demographic slices, together with a 'default demographic'. Because the slices do not need to cover the entire crowd, it may happen that an agent is part of none of the demographics from the slices, in which case it will be part of that default demographic.

A crowd also contains a distribution of agent attributes. Several attributes must always be present, such as age group (kid, adult or elder), gender and preferred movement speed. The used agent model may impose other required attributes. These distributions describe the general distribution of attributes across the crowd, and can be 'overridden' in each demographic.

A diagram of the structure of a crowd profile example can be seen in Figure 1.

4.1.2. Crowd Socket

In order to place a crowd in each concrete environment, some information on that environment is required, be it enriched with semantics or not. We now discuss the crowd socket component of our crowd model, and, in particular, how crowd models, as defined above, can be inserted in concrete environments. The specifications listed below are the bare minimum required, and can be extended to facilitate extra features of the used agent model.

Agent Object Type. First and foremost, to insert a crowd into an environment it must be known which type of entity will represent agents (e.g. a generic human, a police officer or a car). This must be a physical object, since agents need to have a position and orientation. Because a single environment may contain multiple types of crowds at once, the object type can be specified for each used crowd. Attributes of the 'controlled' entities may be set using the agent attribute distributions in the crowd and demographics.

Spawn Spaces. The crowd socket will typically also specify where in the environment the agents will come (or *spawn*) from. Each type of spawn space must be specified individually, and a desired spawn rate needs to be specified. In the environment, each space of the given type will be treated as a spawn space, so the specification should be as specific as possible. For example, if the space 'doorway' is given as a spawn space, all doorways in the environment will become spawn spaces.

The complementary type of space, where the agents leave the environment, does not have to be specified in the crowd socket. Since not all exits need to be spaces, they have to be defined in the semantics by actions that remove the agent from the environment. The reason for this choice is that leaving should be an *action*, meaning that the agent must *want* to leave and simply being in the space designated as 'exit' does not necessarily mean that the agent intends to leave.

Spawn Rate & Conditions. For each type of spawn space, the desired spawn rate must be specified, which can be either a fixed value or a random distribution. This is the minimum time that will elapse between two agent spawns.

Conditions may also be imposed on the spawn space to prevent agents from spawning. For example, agents might only spawn when the door is open, or while there are less than 500 agents in the environment.

4.2. Semantic Agents

This section discusses our particular agent model, which ultimately makes use of the semantics present in the environment.

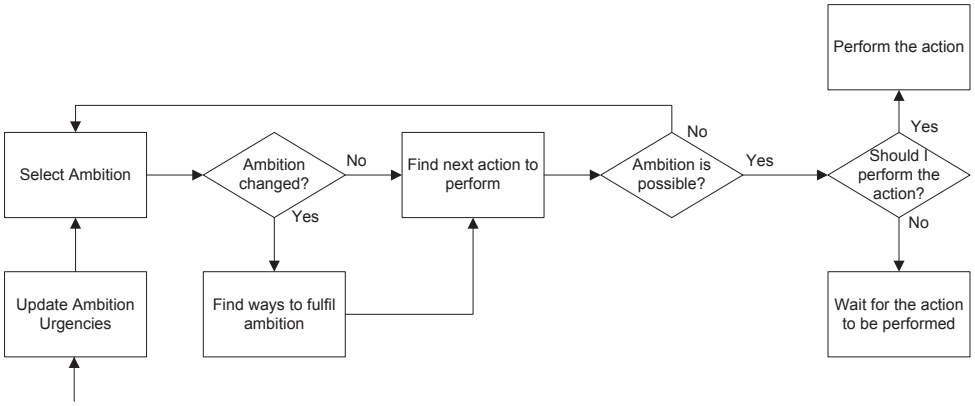


Figure 2: Overview of a single round in the 'thought process' of a semantic agent.

Aside from its global structure, including a specification of the accompanying type of demographic, we also discuss the global agent thought process.

4.2.1. Agent Model

Our agent model is a variant of the BDI (Beliefs, Desires, Intentions) agent model, first introduced by Bratman [33]. Agents have a set of goals that they wish to fulfil, each one containing one or more semantic desires (as introduced in Section 3.2). For each goal, they also have an urgency valuation function, or rather a way of determining how urgent each of the goals are. The combination of goal and urgency valuation function is what we call an ambition. Aside from assigning a numerical value to the urgency, the valuation function also determines if the ambition should be taken into consideration at all (for example, whatever happens, an agent should only buy a jacket once).

The set of ambitions in an agent are emphdesires in BDI, while emphbeliefs are implicitly present in the form of a world view; a description of the part of the environment that the agent knows about. *Intentions* are represented by the ambition selected by the agent and the paths it found to satisfy it.

In the type of demographic specific for this agent model, a set of these ambitions is specified along with a probability. This probability indicates the chance an agent, randomly picked from all agents in the demographic, has that ambition. We also introduced two extra attributes for all agents: *determination* and *distractibility*. The higher the determination of an agent, the more likely it is that it will fulfil the ambition it selected before attempting to fulfil another one.

By letting a demographic specify a set of possible ambitions, we differentiate ourselves from works such as [32]: instead of explicitly specifying *where* the agents will go at what times, we specify *what* the agents want to accomplish when certain conditions are met.

To make further use of ambitions, we also extended the crowd socket for our agents. Aside from the basic information, one can also specify actions that insert ambitions into agents, along with an insertion strength. Whenever those actions are successfully performed on an agent, the agent has a chance (determined by the insertion strength, as well as the agent's distractibility) to gain another ambition. Such a behavior of actions most closely resembles advertisements: they are designed to make agents want to do things they would never have thought of on their own.

4.2.2. Agent Thought Process

An overview of the 'thought process' of our agents can be seen in Figure 2. Note that this flow diagram only indicates the steps an agent will take when it is actually allowed to do something. If it is already performing an action, or certain actions are being performed on it, this whole process is not performed.

Update Ambition Urgencies. First, the agent re-evaluates all its current ambitions, and determines which ambitions may be considered.

Select Ambition. Here, the ambition with the highest urgency is selected. If the previously selected ambition has a high urgency as well, that ambition may be selected instead, depending on the agent's *determination*; the higher this determination, the higher the discrepancy in urgency values can be, before the agent switches ambitions.

Find Ways to Fulfil Ambition. If the selected ambition is different from the one the agent was trying to fulfil in the previous round, it needs to find out how to achieve its new ambition. The agent will query its world view for entities able to perform an action affecting one of the desires in its ambition. For each of those, desires are generated for its requirements, and so forth. This search only stops when all possible paths are found.

Find Next Action to Perform. Once both the ambition to achieve, and the ways to achieve it, are known, the agent can select the action to perform next. The action selection mechanism will not try to find the optimal way of achieving that ambition: since the state of the environment can change while the agent is trying to fulfil it, finding that optimal path is unfeasible.

Our agents use an action selection mechanism based on expected action cost. One could see this as the agents being lazy

and short-sighted; they desire the least amount of effort, but will not take into account the amount of effort future actions might cost. A more graceful way of putting it is that agents opt for immediate gratification for instant feedback, instead of thinking of the long-term benefits. If multiple actions have the same cost, the agent will randomly select an order.

If there is no possible next action, the ambition cannot be fulfilled in a way know to the agent. In that case, a new ambition will be selected, with the current ambition taken out of consideration. On the other hand, if the agent did find an action that should be performed next, there are three possibilities:

1. If the agent is the actor of the action, the agent should perform the action.
2. If the agent is the target (either the direct target or indirect target), it should most likely wait for the action to be performed.
3. If the agent is neither, it cannot know when the action has been performed, except possibly by detecting the results of the effects of that action. The agent will assume the action will be performed soon, and will select a different action instead.

5. Framework

We have implemented the crowd and agent models described in the previous section, along with an interactive editor for both. This section discusses the resulting prototype framework. First, an overview of the system will be given, along with descriptions of each component. Then, the editors for the two models are introduced.

5.1. System Overview

A schematic overview of our system can be seen in Figure 3. The dashed boxes indicate components that can be replaced by custom versions relatively easily. The rest of this section will give a more detailed description of the various components.

5.1.1. Simulator

The simulator is the central part of the framework, and connects the various components. It will not do anything significant itself; it only propagates the data between the different components, and allows the components to interact.

It does however contain the simulation update-loop. The first step is letting the semantic virtual world know to update all spawn triggers, to see if any more agents should be added to the environment. After that, the semantic virtual world is updated to make the environment state up-to-date. The agents themselves are updated afterwards, and finally the navigation engine is requested to update the position of the agents.

5.1.2. Agent Engine(s)

The agent engines take care of the agent update process, and is thus specific for the used agent model. For our agent model, the engine does not need to do much more than delegate the update process to each of the agents. However, if the agent model

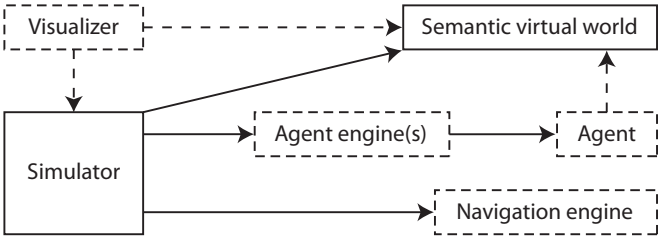


Figure 3: Schematic overview of our framework. The full arrows represent components requesting an update, the dashed arrows represent components querying other components. The dashed boxes represent components that are easily interchangeable in our approach. For example, our approach is independent of the specific algorithms for navigation of an agent towards a target position.

requires some form of communication between agents that can not be handled using semantics, the accompanying agent engine should handle that.

Detecting actions that insert ambitions into agents is the only extra inclusion in our agent engine. This detection could also be part of the agents themselves, but that would mean that the agents already know what ambitions they may eventually get, which is less plausible.

5.1.3. Agent

The abstract agent, as defined in our simulator, requires three properties to be specified by the model-specific implementations. These properties are all related to the movement of the agents; visual information of the agents is assumed to be specified in the semantic representation that the agent controls, and any other information is specific for each agent model:

Motion Type. Each agent must indicate if it takes care of the movement by itself, or if the navigation engine should handle it. This value need not stay the same throughout the life span of the agent.

Target Position. To let the navigation engine know where to send the agent, each agent specifies its target position. This will not be used internally if the motion type indicates that the navigation engine should leave this agent alone.

Preferred Movement Speed. This property is by default the value obtained from the crowd profile. However, if the agent model decides that the agent is in a hurry, or very tired, adjusting this value will let the navigation engine know to adjust the movement speed (when possible).

The agent can query the semantic virtual world to find ways to fulfil an ambition.

5.1.4. Navigation Engine

The navigation engine handles the locomotion of the agents, based on their orientation and target position. Any collision

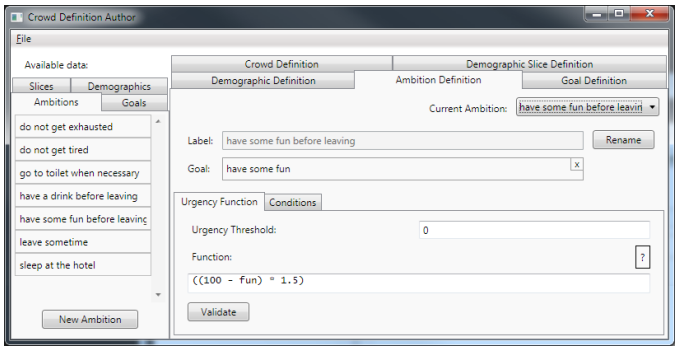


Figure 4: Screenshot of the Crowd Profile Editor.

avoidance should be handled here, although collision detection should be left to physics handlers provided to the semantics.

We chose to allow only a single explicit motion engine, to encourage simulating all movement with a single approach. Having multiple navigation engines makes it much harder to create a realistic simulation. If a different engine is still required for part of the agents, it can still be achieved by incorporating navigation into the agent engine.

The navigation engine used in our system is a small wrapper around a path planner using Explicit Corridor Maps [34].

5.1.5. Visualizer

The visualizer is not strictly a component in the simulation system, since even without it the simulation will work. The simulator gives access to the semantic virtual world as well as the agents controlling the entities in that environment, which is all that is required to visualize what is happening in the simulation.

5.2. Crowd Editor

To aid in the creation of crowds and their addition to semantic virtual worlds, we have created interactive editors for Crowd Profiles and Crowd Sockets. To emphasize the independence of the two components of our crowd model, they are separate editors. The editors are also specifically built to create crowds of agents using our semantic agent model.

5.2.1. Crowd Profile Editor

The Crowd Profile editor allows the user to create a crowd profile from the bottom up, as well as to load an existing crowd and edit it at will. A screenshot of the editor can be found in Figure 4, displaying the editing tab for ambitions.

General structure. The left column of the editor provides aggregated lists of goals, ambitions, demographics and demographic slices, and provides easy access to them at any point in the editing process for either selection or drag and drop functionality. These components also all have a label, which must be unique, in order for them to be more easily distinguishable for the human eye. The rest of the editor displays what the user is currently editing. If the editing requires the indication of one of the components in the left column, the user only needs to drag and drop the desired component into the appropriate location.

Goals. Some agents may have the same goals, but with different ambitions wrapped around them. For example, most agents only want to go in a roller coaster once or twice, while fans want to go as many times as possible. To this end, the goals can be defined and edited separate from the ambitions.

The desires for a goal can be added, removed and edited freely. Although the same desires may also be used in multiple goals, we chose not to support this in the editor because the desire definitions are almost trivial. Desires will be displayed with a type and a number to be able to distinguish between them while editing.

Desires. When adding a desire to a goal, the user first chooses which type of desire. These can be desires for the agent to be in a particular state or for an agent's attribute to be higher than, lower than or equal to a particular value. One can also express a desire for the agent to be in a particular type of space (e.g. the baggage claims area of an airport) or to simply leave the world. States, attributes or spaces can be chosen from the semantic library. This means that whenever e.g. new attributes are added to entities in the semantic library, these are immediately accessible in the crowd editors.

Ambitions. To set the goal of an ambition, the user only needs to drag and drop the appropriate goal from the list on the left side of the editor. Alternatively, the user may create a new goal that is automatically inserted into the ambition by using the context menu.

The urgency valuation function is split up in two parts. The first is a function that assigns an urgency to the ambition (simply the 'urgency function'), the other is a set of rules (conditions) that determine if the ambition should be considered. The urgency function of the ambition can only be defined by entering the function as text. The format of this function is very similar to mathematical expressions in most programming languages.

The conditions of the ambitions are again similarly defined as the desires of goals. One of the condition types uses a similar function format as the urgency function in order to access the attributes of the agent. The only difference is that the condition function requires the result to be a boolean instead of a number. Both functions are stored as-is, although only if their format is valid will they result in a function that returns a non-null value. In addition, the user can also put an occurance condition for the ambition and a condition on whether or not another ambition is already achieved. For the latter, the user simply drags and drops an ambition and chooses whether this one should or should not be achieved first.

Demographics. The definition of a demographic for our agent model makes the editing rather straightforward. Ambitions can be added to the demographic using drag and drop. Once the ambition is placed in the demographic, the probability can be spec-

ified (see also the screenshot in Figure 4). The demographic conditions also work the same as the ambition conditions.

However, each demographic may also override the agent attribute distributions by virtue of the specific agent attributes *determination* and *distractibility* (as introduced in Section 4) and *preferred movement speed* (these are not to be confused with the semantic attributes). A simple checkbox lets the user indicate that the demographic overrides the attribute distribution of the crowd. Once that is checked, the user can create a tree that defines these distributions. The tree itself only specifies the age group and gender of the agent, but the user can decide in which order. Because of the tree structure, the distributions of the attribute do not need to be independent. Aside from a probability for the agent to 'select' it, each node in the tree may contain a distribution for the other three agent attributes. These distributions only hold for agents that 'selected' that path in the tree, so if the distributions are placed in the root node, all agents with the demographic will use that distribution.

Demographic Slice. The way demographics are put into slices is the same as ambitions are put into demographics. The user only needs to drag and drop the demographics from the left column onto the indicated spots. The coverage percentage of each demographic is also inserted in the same way, albeit with values that range from 0 to 100. If a demographic has conditions the user can indicate will be able to indicate how to handle unsatisfied conditions. The conditions that limit the coverage of the slice are handled in the same manner as the other conditions in this editor.

The Crowd. Most of the editing of the crowd itself works exactly the same as the editing of its components. The list of slices is reminiscent of the list of demographics and the list of ambitions in the demographic slice and demographic definitions respectively. The default agent attribute editor is the same as the one used in a demographic.

For a single demographic, an extra spot has been reserved at the bottom. This demographic is the 'default' one, and need only be set if none of the slices covers the entire crowd. Finally, at the top there is room to enter the label of the crowd.

5.2.2. Crowd Socket Editor

Because of the simplicity of the crowd socket for our agent model, this editor is much smaller than the one for the crowd profile. There are two main content collections that need to be filled: *crowd sources* and *ambition inserters*.

The file resulting from this editor will also contain the data found in the output of the crowd profile editor. We could have just kept references to the crowd profile files, but that would mean the files could not be moved once made. It also has the benefit that only one file needs to be loaded when actually inserting the crowd in the environment.

Crowd Sources. A crowd source is a combination of a crowd profile, a spawn space and spawn trigger. The space can be entered by typing the name of the space, or by browsing through a loaded database to prevent typing errors. In order to enter the spawn rate, the user first has to select what type of distribution the spawn rate should have. After that, only one or two values are necessary to instantiate the spawn rate.

The definition of conditions works similar to all other conditions used in the Crowd Profile editor. However in this case there are two sets of conditions. The first set will prevent the spawn trigger from updating, the second set will prevent it from notifying the spawn space that an agent should be spawned. This split allows for a greater variability in the crowd sources. For example, a (small) revolving door will deposit an agent in the environment every 2 seconds. When the door is broken (and cannot revolve), no agent will be spawned. However when a broken door is fixed, the revolution of the door continues where it left off, instead of continuing wherever it would have been if it was never broken. In other words, the crowd source of the revolving door should have an update condition that the door is not broken, with possibly a spawn condition to prevent the environment from becoming overfull. The latter can be interpreted as agents staying in the 'wedges' of the revolving door, leaving as they come, before they even enter the environment, because they found it too full.

To load the crowd, the user must browse to the file generated by the crowd profile editor. Alternatively, the file can be loaded by dragging the file onto the indicated location.

Ambition Inserters. The editing tab for ambition inserters is similar to the format of the crowd profile editor. On the left side are lists with ambitions and goals, which are automatically populated with those found in any crowd used in the crowd sources. On the right side are tabs for those ambitions and goals, as well as a tab with all inserters.

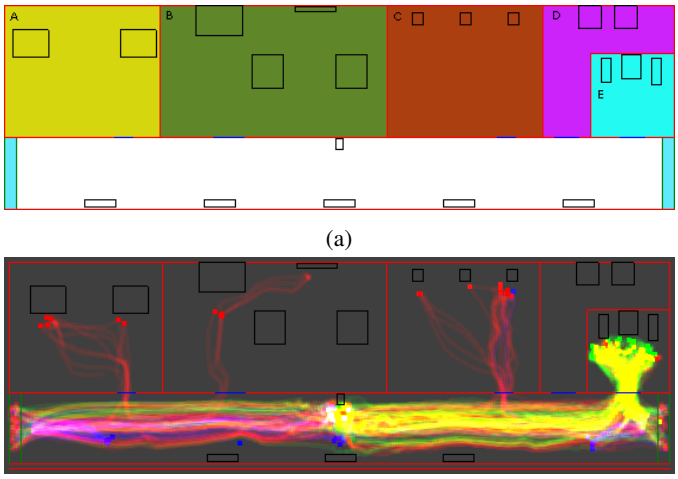
Each ambition inserter requires, aside from an ambition and insertion strength, a reference to a type of action and the entity type of the actor. This pair represents the actual inserter; whenever an entity of the given actor type performs the given action, the given ambition will be attempted to be inserted. The user can also indicate how many times the insertion should be attempted.

6. Results

To demonstrate the viability and merits of the proposed approach, three cases have been created, wherein different environments were populated with the same crowd. Section 6.1 introduces the crowd template used in all three cases and describes the setup of the basic environment, which the two first cases, discussed in Sections 6.2 and 6.3, expand upon. Section 6.4, in turn, describes the third case, which is the application of the same crowd template on a different, much larger and more complex environment. Please refer to the accompanying media of this article for several videos on the simulations performed for these three cases.

6.1. Basic Setup

In this section, we introduce a basic environment, the crowd to be used throughout the section, including its environment



(a)

(b)

Figure 5: Floor plan (a) and heatmap (b) of the basic setup. In the heatmap, different colors are used for different age groups: blue traces are elderly, red traces are adults, and green traces are kids. The rooms shown are: (A) a bank, (B) a hotel, (C) a cafe, (D) a hidden garden with plum trees, and (E) an arcade hall. The cyan areas in the hallways are spawn spaces.

specific settings, and the initial results of populating the former with the latter.

6.1.1. The Environment

Layout and Semantics. The environment represents a section of an airport before customs. It is a hallway with benches on one side, on which agents can rest, and several establishments on the other side, including a bank (A), hotel (B), cafe (C), a hidden garden with plum trees (D), and an arcade hall (E). An ATM is also present in the hallway, at the entrance of the hotel. This floor plan can be seen in Figure 5a.

The agents enter from the cyan spawn spaces at the left and right sides of the environment, and can exit there as well. They can also 'exit' into the hotel, but in order to do so, they have to check in first. Checking in can be done at the front desk of the hotel, but requires a credit voucher from the bank. In order to use the lounge chairs in the hotel lobby, an agent must be checked in as well. To use the services of the arcade hall and the coffee shop, money is required, which can be retrieved at the ATM or at the bank. The plums from the trees are free. The arcades increase the enjoyment of an agent, having drink sates thirst and slightly increases the need to go to the toilet, and eating plums greatly increases that need. The cafe and the bank are only usable by adults and elderly.

Crowd Socket. There is only a minimal configuration required for this environment. The cyan spawn spaces at the ends of the hallway are specified, along with their spawn rate. Furthermore, the specific semantic entity that represents the agent is defined. Finally, the maximum number of agents in the environment is set to 50, because with any more agents the doorways might become clogged very fast.

6.1.2. The Crowd Profile

Throughout the three cases in this section we will use exactly the same crowd profile. In doing so, we wish to make our point that virtual environments enriched with different semantics are able to induce rather different behaviors on the same crowd. In other words, you don't need to 're-program' the agents to have them act or react to different environments or stimuli.

The distribution of the agents' genders is set to be equal, but in the age group distribution, adults are more likely than elderly and children: approximately 15% will be children, 15% will be elderly, and the other 70% will be adults. The average walking speed decreases with the age, but their determination and distractibility are equally distributed.

Every agent has a small ambition to leave the environment, such that they will never be standing still, as well as the ambition to go to the toilet when they need to. The latter increases an agent's urgency when it 'fills up'. Most children (80%) will want to do something fun before they leave, while only 10% of the adults have that ambition, and none of the elderly. All adults and elderly have the ambition to rest when they get tired, but elderly get tired much faster than adults. Half of the adults and three-quarters of the elderly enter the environment in search of a drink, and half of the adults who do not want a drink aim to check in at the hotel. All these specifications, regardless of how likely and sensible they are, serve to clearly illustrate how elaborate a crowd description can be.

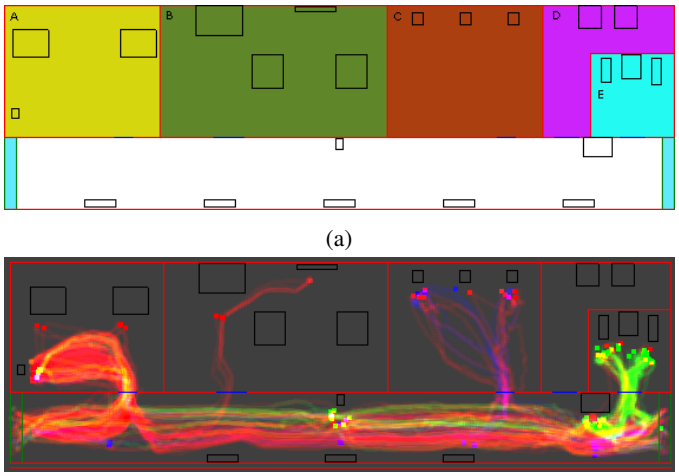
6.1.3. Analysis

A heatmap of the simulation of this basic setup can be seen in Figure 5b. The easiest observation to make from this map is that the agents make no use of the available plums at all. This is exactly what is expected though, as the effect of the plums does not positively influence any of the ambitions that the agents have. The agents will also never exhibit the ambition to relieve themselves, as there is nothing in the environment that they can use for that purpose.

Another general observation is that the agents will approximately equally distribute themselves over the various objects giving the same service. As discussed in Section 4.2, the choice of actions that agents choose to perform is based on the expected 'cost' when performing those actions. Because the calculation of the expected cost is rather crude, it is not necessarily the object closest to the entrance that gets used the most.

The heatmap also shows several agents that at some point sat on the benches. Because the average life time of an agent in the environment is low, only elderly agents have rested on the benches, adults left before they got tired. The hotel lounge chairs are unused in this sample. Because the benches are spread out over the environment, agents will find it much easier to just sit on a bench, than going through the process of checking in.

Our agents take into account only one ambition at a time. Thus agents will not combine the ambition to check in to the hotel with the ambition to rest; whatever is deemed more urgent will be achieved first. Something similar could be observed with regard to the coffee shop and arcade hall. If an agent



(a)

(b)

Figure 6: Floor plan (a) and heatmap (b) for Case 1. The heatmap uses the same color scheme as the heatmap in Figure 5. The rooms shown are: (A) a bank, (B) a hotel, (C) a cafe, (D) a hidden garden with plum trees, and (E) an arcade hall. The cyan areas in the hallways are spawn spaces.

wants to visit and make use of both establishments, the agent will need to use the ATM twice (or go to the bank once) to get enough money. However, agents will never get more money if they believe they have enough to start satisfying their current ambition. Thus an agent may visit the ATM before going to the coffee shop, after which it needs to go to the ATM again before going to the arcade hall.

As long as one does not try to follow the individual agents, the system creates a fairly realistic 'background crowd'. Of course, if more detailed behavior is desired, the semantics and/or crowd definition will need to be extended accordingly.

6.2. Case 1: Hurdles as preconditions

For this first case, we slightly adapted the basic environment described above, in order to highlight the strength of the semantics in the world and in its objects. The same crowd is used, and the modifications to the environment were so minor that the crowd socket can remain the same. We first describe the environment changes, after which we discuss the results of having the crowd move through the new environment.

6.2.1. The Environment

This environment is mostly the same as the environment from Section 6.1. However, in order to use the bank counters, a ticket is required from the bank's front desk in the bottom-left corner of the bank. A similar change has been applied to the arcade hall: in order to use the arcade machines, coins are needed. Money can be traded in for coins at the machine near the entrance, outside of the arcade hall. An updated floor plan can be seen in Figure 6a.

6.2.2. Analysis

Since the same crowd is used in this environment, we still see mostly the same behavior. However, as can be seen from

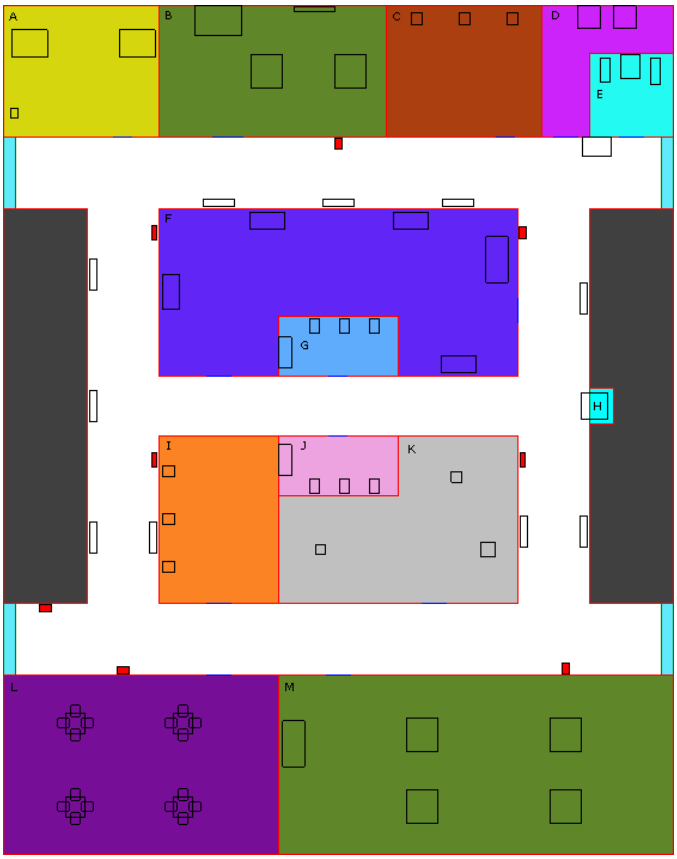


Figure 7: Floor plan for the extended environment of Case 2. This environment contains ads, which give agents ambitions that are not specified in their profile. These ads are marked in red. The rooms shown are: (A) a bank, (B) a hotel, (C) a cafe, (D) a hidden garden with plum trees, (E) an arcade hall, (F) a toilet (G for males, J for females), (I) a juice shop, (K) an art gallery, (L) a burger restaurant and (M) another hotel. The cyan areas in the hallways are spawn spaces.

the heatmap in Figure 6b, there are some significant differences. The major difference is that a disproportionate number of agents is using the front desk of the bank. Because getting a ticket is very easy, agents will (usually wrongly) assume that using the bank is faster than using the ATM. There are some agents that still use the bank after they got a ticket, but those are the same agents as the ones that checked in at the hotel a few moments later.

It is also apparent that agents often walk back and forth between the coin machine and the arcade cabinets. Agents are not able to satisfy their ambition to have 'enough' fun with just one or two plays, and their coins will usually run out before the ambition has been satisfied. Because we made our agents opt for instant satisfaction, rather than finding the optimal route to satisfy all ambitions, they will not hoard enough coins in order to keep playing until they are done. Instead, only one batch of coins is bought at a time, and a new batch is only bought when the current batch runs out.

Since this environment is only slightly different from the

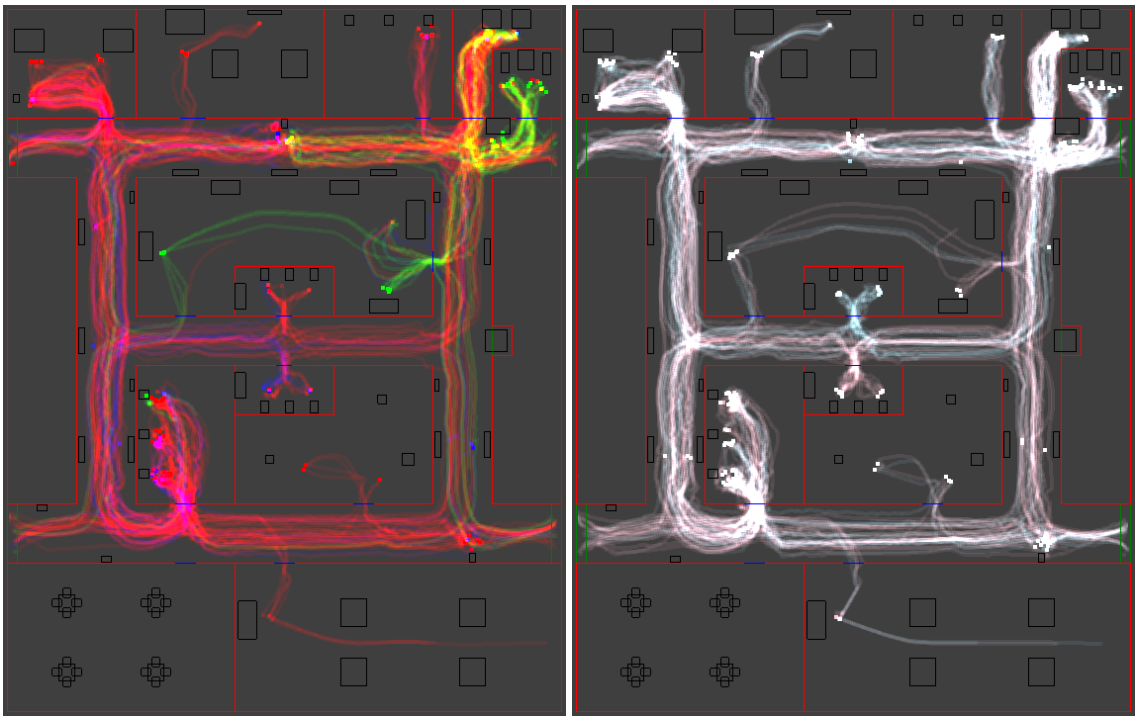


Figure 8: Heatmaps for Case 2. The left map uses the same color scheme as the heatmaps in Figure 5 and 6; in the right map, blue traces are males and pink traces are females.

original environment, having the same crowd run in this simulation gives only a little taste of the strength of semantics for crowd simulation. By changing the way tasks have to be solved, without changing anything to the agents themselves, we show that for deploying semantics, it does not matter how convoluted the path towards the goal of the agents is. It also does not matter in how many ways the agents can fulfil their ambitions, nor what type of objects the agents need to use, nor how those objects can be used.

6.3. Case 2: The power of ads

This second case uses a larger environment, where we introduce ads, which will induce agents to do things that were not specified in ther crowd definition. Section 6.3.1 describes the differences between the new environment and the one used in Case 1, and Section 6.3.2 discusses the resulting crowd behaviour in this new environment.

6.3.1. The Environment

Layout and Semantics. This environment is an extension of the environment used in Section 6.2, as can be seen in the floor map in Figure 7. The top hallway is the same as before, although there are less benches on the bottom wall to accommodate the new hallways. Ads are marked in red.

The central block contains a toy store (F), toilets (G for males, J for females), juice shop (I) and an art gallery (K). The toy store will provide fun for kids as they visit it, while the art gallery does the same for adults and elderly. An ad placed in the left hallway will entice agents to buy something at that toy store, although none of the agents will shop there automatically. The toilets, separated by gender, finally provide the agents with a way to relieve themselves.

Near the entrance of the toy store is a sign with information on the plum trees, thus acting as an ad for this fruit. In the same hallway an ad for drinks is located, which means that kids may also want a drink if they pass by that ad. A fountain can also be found close by (H), although it only has an aesthetic function.

The bottom hallway contains a burger restaurant (L) and another hotel (M). An extra ATM is placed in front of the new hotel, and a generic ad for food is placed in front of the burger restaurant. The same ad is also placed in the left hallway. Agents can enter and leave from the left and right ends of the top an bottom hallways, and an ad for the hotel chain is also located at the exit near the burger restaurant.

With the addition of ads, this environment needs some more configuration before agents can use all of it. For each type of ad entity, we specify what kind of ambition they want the agents to have. For the plum ad, the agents will want to eat at least one plum before leaving the environment. The food ads will make agents want to eat something before they leave; in this case that will have to be either a plum or a burger at the restaurant. The toy store ad and hotel ad are fairly self-explanatory: they will make the agents want to buy something at the toy store and check-in to one of the hotels respectively.

6.3.2. Analysis

Behavioural Evaluation. Now that the environment contains ads, the crowd ends up doing what it would never think of before, as shown on the heatmaps in Figure 8. Since the plum trees are not on the route for anything else, the fact that agents are

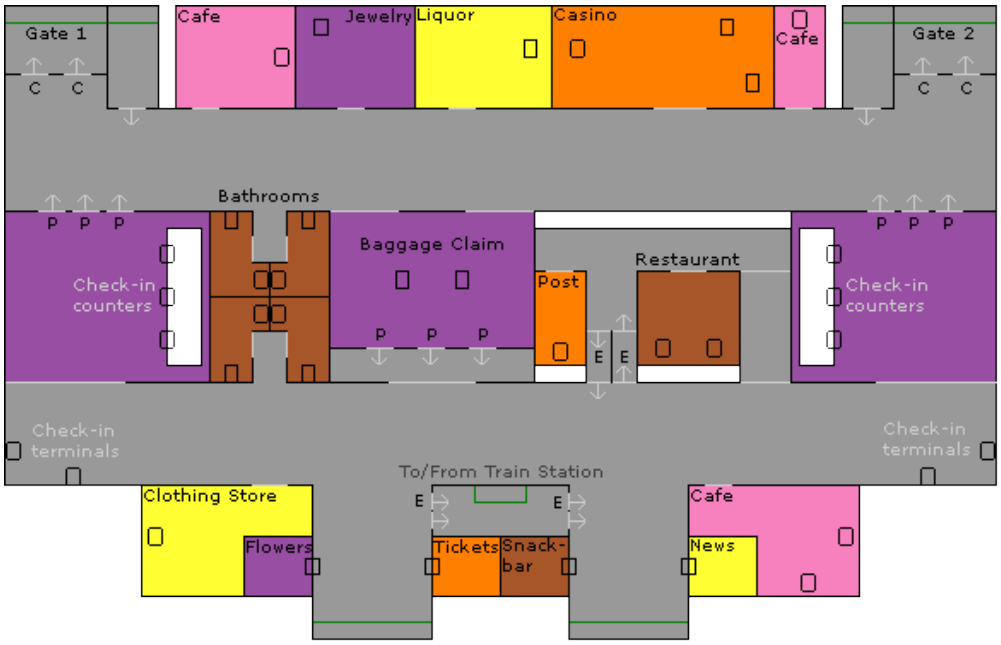


Figure 9: The floor plan of the arrivals and departures section of the airport. *P* indicates that a passage is a passport control; *C*, customs; and *E*, escalators. The arrows indicate one-way passageways. The areas of the hallways behind the dark green lines are spawn spaces.

walking near them gives enough evidence that the plum trees are now being used. The burger restaurant is eerily quiet, but that is because it costs so much: plums will also sate the hunger of the agents, so they would rather eat a few plums than go to the restaurant.

Another interesting observation is that, while the art gallery can be used by elderly, they never actually use it. This has a simple explanation, since, according to the crowd profile (see Section 6.1.2), the elderly agents do not have the ambition to do something fun before leaving the environment and there are no ads that will make them want to visit the gallery either.

In conclusion, now that the environment is significantly larger, the power of semantics is even more visible. Without changing anything in the crowd profile, we can let the crowd roam through this environment with just as much realism as in the smaller environments.

Performance Analysis. In Case 1, we could simulate approximately 175 agents without getting a noticeable lag in performance. In Case 2 that same amount of lag occurred at around 100 agents. The main bottleneck for this volume of agents is the process of an agent finding all possible ways to satisfy its ambition. This process is influenced by the amount of actions the agent can take on each entity type. The number of instances of each entity type does not play a role in this process, since the graph is built using the entity types, not all their instances. Therefore enlarging the world and adding many more instances of existing entity types will not slow down this process. Optimizing the performance was outside the scope of this project, however it is clear that some relatively simple improvements could increase performance significantly. For example, reusing graphs between multiple agents, instead of recreating them every time, would decrease the influence of the agent count a lot.

6.4. Case 3: airport case

This final case drastically changes the circumstances, by introducing a completely new and considerably larger environment. In doing so, the original crowd profile from the previous cases can be tested on its portability to such a different environment. In addition, this case illustrates the rest of the functionality of our prototype simulator.

6.4.1. The Environment

Layout and Semantics. Even though the environment is significantly different from that in the previous cases, it still represents part of an airport, this time, the section of the airport for arrivals and departures. This is where agents can check in and board a plane, or disembark, collect baggage and leave the airport. Its floor plan can be seen in Figure 9.

As can be seen on the floor plan, one-way passageways have been introduced, such as passport control and the customs, but also escalators and turnstiles.

The environment contains two boarding gates at both corners in the top, which are composed of two one-way spaces: the outermost spaces are where agents pass the customs and board the plane (the departure space), the innermost spaces are where agents exit the plane (the arrival space).

There are also two check-in areas, the two (outermost) purple spaces on the middle row. That same row contains the baggage claim area, which is the middle purple space. The entrance

in the lower center is an entrance from (and exit to) the train station. The other two entrances lead to the main airport entrance, possibly past the hallways from the previous two cases.

The row of establishments between the gates are: a cafe, a jewelry store, a liquor store, a casino, and another cafe. On the left side of the baggage claim are toilets, on the right side is an elevated section with a post office and a restaurant. This section is accessible by both the escalators (between the post office and the restaurant) and by a set of stairs (on the other side of the restaurant). The establishments at the bottom row include a clotheing shop, a flower kiosk, a train ticket kiosk, a snack bar, a news stand and a third cafe.

The semantics in the environment is as follows.

Miscellaneous Stores Most 'stores' (jewelry, liquor store, clothes store, restaurant, snack bar, flower kiosk, train ticket kiosk and news stand) have one counter, where agents can buy their respective goods (the restaurant only serves food). Each of the counters requires the agent to have a certain amount of money, and set a flag that the agent has bought something there when the exchange was successful.

At the post office, agents can deliver a letter or postcard to be posted. This service also costs money, although only a little amount. The cafes only serve drinks, providing agents a decrease of their thirst.

Casino Since there is no ATM in this environment, the casino is the only place where agents can possibly get more money. The three casino tables all work the same; they require some money, and the return value is based on chance. There is a 20% chance of the agent winning a small amount, and a 2% chance of winning a big amount. All other times the agent loses its inlaid money.

Toilets The toilets work exactly the same as in the previous environment.

Check-In The check-in desks require the agent to have a plane ticket, and have the result that the agent has checked in. Below both check-in areas are two terminals that allow the agents to check-in faster. However these cannot be used if the agent has additional baggage. The agents will only be able to pass the passport controls near the check-in counters once they are actually checked in.

Gates The customs at the gates can be passed whenever an agent is checked-in *and* past passport control. The requirements for agents to be checked in was made to prevent arriving agents from taking another plane (as if this particular airport would not support direct transfers).

Baggage Claim The baggage claim area is where the agents can claim their baggage. If an agent has any baggage to pick up, it can only go past the passport control here after that baggage has indeed been picked up.

Other The escalators can be either on or off. This environment contains no semantics that adjust this state, but it can be toggled during the simulation, if desired (see Figures 10 and 11).

In order to use the exit by the train station, a train ticket is required.

Two types of agent entities simulate people departing and people arriving. Both have the same attributes, although arriving people will have different default values for some of them. There are four numerical attributes present; money, hunger, thirst and toilet need. The value of hunger and thirst will increase gradually over time. All four attributes get a random initial value. Most of the other attributes have to do with inventory control, such as 'bought jewelry', 'has baggage' and 'has plane ticket', but also state control, such as 'is checked in and 'is past passport control'. Arriving people will never have a plane ticket, while only half of the departing people will have one.

Crowd Socket. The 'departing' crowd will be spawning from the main entrances and the train station. The main entrance will spawn a new agent (of type 'departing human') on average every $0.4s$, while the train station will spawn one approximately every second. From the entrances at the gates the 'arriving' crowd will spawn, and will produce one agent (of type 'arriving human') every $0.5s$.

For practical reasons, crowd sources have a condition that will prevent more than 50 agents being in the environment at any one time. Preliminar tests showed that this amount resulted in a reasonably crowded environment.

6.4.2. The Crowd Profile

Two crowd profiles have been made for this environment; one for the crowd arriving at the airport from a plane, and one for the crowd that arrives through other means and may want to depart using a plane. The attribute distribution of the two profiles is the same. The determination is set to 0, so that agents will always try to do what is most urgent. The average (preferred) movement speed is $2ms^{-1}$, with a minimum of $1ms^{-1}$ and maximum of $6ms^{-1}$. The amount of money an agent initially has is random.

The arriving crowd has three demographic bands; one for every arriving agent, and two for men and women, respectively. 30% of all arriving agents will have the ambition to communicate with home, and half of all arrivals will want to check the local new (these groups may overlap). The desired method of leaving the airport splits the arriving crowd in two: 50% of the agents will want to use the train, while the other 50% want to use the front door (to leave by car, bus, taxi, etc.). Agents that want to use the train will also have the more generic ambition to leave sometime, as they may not have enough money to actually buy a train ticket. The latter ambition has a very low urgency though, so it will only be adopted if there is no other possibility. The band 'everyone' contains two sub-bands; one band is for the desired mode of transportation described above, while the other describes which agents are hungry and/or thirsty. 28% will be thirsty people, 17% will be hungry people, and 8% will be hungry and thirsty people. Hungry agents will want to eat

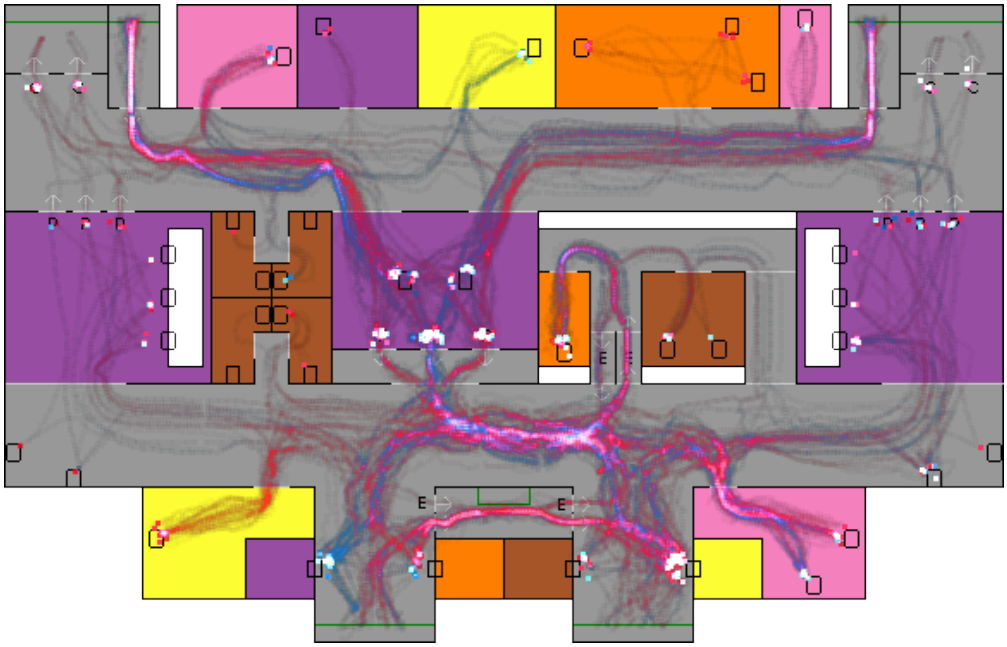


Figure 10: Heatmap for the dedicated crowd of Case 3 with the escalators on. The color scheme for the agents is different this time, as the age group is of little importance in this simulation. Male agents are painted blue, while female agents are painted red.

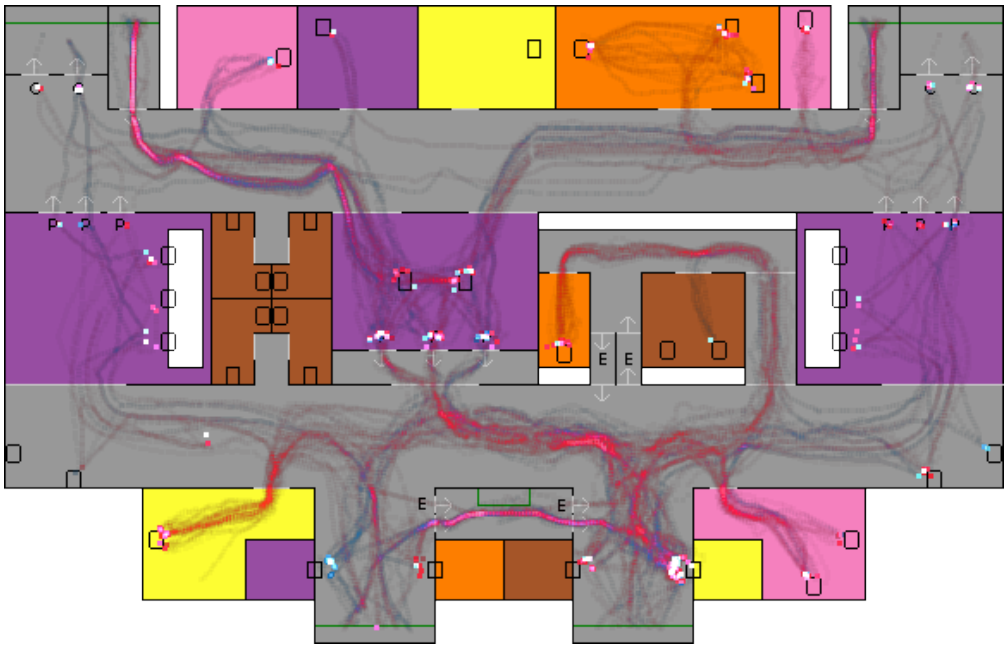


Figure 11: Heatmap for the dedicated crowd of Case 3 with the escalators off. The color scheme is the same as the one used in Figure 10. The exposure time of this heatmap is much shorter than in Figure 10.

something at some point; it does not matter what or how much it fills them up. Agents belonging to the hungry (or thirsty) demographic, will have the ambition to keep their hunger and thirst low. If an agent is in any of these demographics, it will also have the ambition to go to the toilet when necessary.

The two gender-dependent bands have a similar structure. Each contain only one demographic, that covers all of the agents in that band. Of the men, 30% will have the ambition to buy liquor, and 10% will have the ambition to buy some jewelry. All men will have the ambition to buy some flowers. In contrast, 60% of the women will want to buy some clothes, while only 15% have the ambition to buy some jewelry.

The departing crowd is structured similarly as the arriving crowd. It also has three bands, also with the same purpose. The band for women is exactly the same, while the band for men only has one difference; only 30% of all departing men will have the ambition to buy some flowers, instead of all men.

The band for all departing agents is also similarly structured. It still has a sub-band that determines if agents are hungry or thirsty, which has the same distributions as well. However every departing agent will have both the ambition to leave by plane, as well as to leave sometime by any means, where the latter again has a low urgency compared to the former. Because not all agents have a plane ticket, those who do not, would have no way to leave if the generic ambition to leave was not present. Aside from these ambitions to leave, 40% will want to buy a newspaper before that.

6.4.3. Analysis

Behavioural Evaluation. This simulation was run twice. The first run used the dedicated crowd, just described in Section 6.4.2, specifically created for this environment. The second run used the original crowd profile defined in Section 6.1.2, evidently, with an adjusted crowd socket, so that the crowd would only spawn from the front door, and the spawn rate has been increased significantly.

Dedicated Crowd. The first that one notices when looking at the long-exposure heatmaps in Figures 10 and 11 is that the traffic in the baggage claim area is significantly higher than the combined traffic in both check-in areas. This has to do with the conditions set on the crowd sources for the simulation. The conditions require that at most 50 agents are present in the *entire* environment. Once an agent leaves, the spawn spaces are treated on a first-come, first-serve basis. Incidentally, it turns out that the spawn spaces at the gates are updated first, so they often get the first chance to spawn a new agent, after which there will be no more room left for agents to spawn from the front door and the train station.

With the current setup, the only way of circumventing this 'bias' would be not to use a limit on the environment-wide agent count but on the number of agents spawned per spawn source. Unfortunately, this brings other problems, since the agents coming from the train station and from the entrances could be considered as the same crowd, even though they are generated by different crowd sources. By combining both a total agent limit (for the departing crowd) and a per-crowd-source agent limit (for the arriving crowd), the desired result may still be achieved with regard to the ratio of the agents in the baggage claim and check-in areas. However, a similar problem might subsequently emerge for the distribution of agents emerging from the train station and from the front doors, for which a more elaborate ordering mechanism would be required.

Furthermore, agents will only consider passing through a connection between spaces if the condition has been satisfied. For example, agents will first stop at the passport control before passing through it, even if their objective does not explicitly require them to have their passport checked. They will also never attempt to pass through a connection that only allows passage in the opposite direction.

Original Crowd. When the crowd profile specified for Cases 1 and Case 2 is used (see Figure 12), it is evident that only a few of the available services are used, where only the bathrooms are used intentionally. The other services, such as the check-in desk and train ticket counter, are only used because the agents want to leave by any means.

Normally, one would expect agents to never use the toilets on the 'transit-side' of the airport. However because the *expected* cost that agents use to choose their action only takes into account euclidean distances, the desired toilet is sometimes 'on the other side'. For the same reason, the left gate is occasionally used as an exit. Because the gate is so close to the toilets, the noise in the action cost function will sometimes make an agent think it is 'cheaper' to board a plane than to exit via the front door if they want to leave the environment.

One can also see a lot of agents that appear to stand still at the entrances. However those are agents that spawned, and immediately exited. Because in this environment only their desire to go to the toilet when necessary can be fulfilled, those who do not need to go, have only the ambition 'leave' by any means left. And since they are already in a proper position to leave, they do so right away. When having the spawn rate at approximately one agents per $0.12s$ (for both entrances), at most 15 agents were walking around at any time.

Performance Analysis. In Case 1 and Case 2 (Section 6.2 and 6.3), our prototype simulator could easily handle 100-175 agents at a time before taking more than a few milliseconds to update all agents. In Case 3, however, at around 35 agents the simulator was experiencing lag, especially when a new agent was spawned.

Adding a barrier in the middle of the environment, in the form of passport controls, increased the number of necessary queries in this final simulation, which causes the drop in performance. If an agent tries to use an object on the other side of the barrier, it first builds the graph to use that object. After finding a possible path, it finds that it needs to pass one of the passport controls, which leads to another graph being built. If that does not succeed the agent will try to find another path, not using the passport controls, which is impossible. Subsequently the next instance of the desired object (or the desired object for the next ambition) is checked, which could very well be on the

Figure 12: Heatmap for the crowd profile from Cases 1 and 2, spawn in this environment. The color scheme used paints the agents by age group again.

other side of the barrier as well. Thus the entire chain of queries is done again, and doomed to fail if all desired objects are on the other side of the impassable barrier. Implementing this detection and pruning algorithms would likely bring a substantial optimization, however that was outside the scope of this project.

The framework running all these cases was completely written in C#. The tests were run on an Intel® Core™ 2 Duo CPU E6850 @ 3.00GHz running Windows 7.

7. Conclusions

Many crowd simulation solutions often target domain-specific environments, using special-purpose methods that are hard to re-configure or re-use in other contexts. We proposed *semantic crowds*, a novel approach designed from the outset with crowd re-use and re-configuration in mind. It integrates several existing research concepts and results within a structured model that allows one to easily define crowd templates in a very flexible and powerful way, and re-use them with minimal modifications for virtually any environment, in which the objects available are spontaneously used in a meaningful manner. This is achieved by (i) having the objects in a virtual environment be extended with *semantics*, beyond just pure geometry, and (ii) having each agent query the environment to find whatever objects are deemed suitable to fulfil its desires.

We briefly described our prototype system, including an interactive crowd editor that provides high-level editing parameters for defining crowd templates. These consist of a *crowd profile*, describing all its environment-independent aspects (e.g. its demographics), and a *crowd socket*, defining how it is supposed to be 'attached' to each particular environment (e.g. its spawn spaces and rate). We showed the power and portability of semantic crowds by illustrating how one same crowd profile flexibly adapts its behavior to quite differently configured environments, containing a wide variety of objects with distinct semantics.

We believe the power of this semantic crowd model will become increasingly apparent as more experiments and extensions are performed on this basis, in order to refine or enrich crowd behavior and overal appearance. An interesting challenge, for example, is to explore better methods of controlling and balancing agent's (short and long term) ambitions, as well as their dynamics under the influence of ambition inserters. Fortunately, this modular approach to reusable crowds can easily combine with many custom implementations of agent behavior model and motion planner, possibly providing more fine-grained control or more realistic and detailed paths.

We can, therefore, expect that in the near future new, even richer, crowd models will profitably be developed that further build upon these high-level semantic features.

Acknowledgements

We thank Fernando Marson and Soraia Musse for their valuable contribution to our research on the generation of semantic virtual environments, previously used in a short version of this article [35]. We also thank Roland Geraerts [34] and Atlas Cook IV, for kindly providing us with their path planner library and promptly assisting us with its integration in our prototype. Finally, we thank the anonymous reviewers for their constructive comments.

References

[1] T. Tutenel, R. Bidarra, R. M. Smelik, K. J. de Kraker, The role of semantics in games and simulations, Computers in Entertainment 6 (4) (2008) 1–35.

[2] J. Kessing, T. Tutenel, R. Bidarra, Services in game worlds: A semantic approach to improve object interaction, in: ICEC '09: Proceedings of the 8th International Conference on Entertainment Computing, LNCS vol. 5709, Springer-Verlag, 2009, pp. 276–281.

[3] M. Luck, R. Aylett, Applying artificial intelligence to virtual reality: Intelligent virtual environments, Applied Artificial Intelligence 14 (1) (2000) 3–32. doi:10.1080/088395100117142.

[4] N. Farenc, R. Boulic, D. Thalmann, An informed environment dedicated to the simulation of virtual humans in urban context, Proceedings of Eurographics'99 18 (3) (1999) 309–318.

[5] K. Otto, F. U. Berlin, Towards semantic virtual environments, in: Workshop Towards Semantic Virtual Environments, 2005, pp. 47–56.

[6] G. Thomas, S. Donikian, Modelling virtual cities dedicated to behavioural animation, Computer Graphics Forum 19 (3) (2000) 71–80. doi:10.1111/1467-8659.00399.

[7] M. Badawi, S. Donikian, The generic description and management of interaction between autonomous agents and objects in an informed virtual environment, Computer Animation and Virtual Worlds 18 (4-5) (2007) 559–569.

[8] M. Kallmann, D. Thalmann, Modeling objects for interaction tasks, in: Proceedings of the Eurographics Workshop on Animation and Simulation, 1998, pp. 73–86.

[9] M. Gutierrez, F. Vexo, D. Thalmann, Semantics-based representation of virtual environments, International journal of computer applications in technology 23 (2) (2005) 229–238.

[10] T. R. Gruber, A translation approach to portable ontology specifications, Knowledge Acquisition 5 (1993) 199–220.

[11] F. Grimaldo, M. Lozano, F. Barber, G. Vigueras, Simulating socially intelligent agents in semantic virtual environments, The Knowledge Engineering Review 23 (04) (2008) 369–388. doi:10.1017/S026988890800009X.

[12] H. Jiang, W. Xu, T. Mao, C. Li, S. Xia, Z. Wang, A semantic environment model for crowd simulation in multilayered complex environment, in: Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology, VRST '09, ACM, New York, NY, USA, 2009, pp. 191–198. doi:10.1145/1643928.1643972.

[13] A. Treuille, S. Cooper, Z. Popović, Continuum crowds, ACM Trans. Graph. 25 (2006) 1160–1168. doi:10.1145/1141911.1142008.

[14] C. W. Reynolds, Flocks, herds and schools: A distributed behavioral model, SIGGRAPH Comput. Graph. 21 (1987) 25–34. doi:10.1145/37402.37406.

[15] X. Tu, D. Terzopoulos, Artificial fishes: physics, locomotion, perception, behavior, in: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, SIGGRAPH '94, ACM, New York, NY, USA, 1994, pp. 43–50. doi:10.1145/192161.192170.

[16] S. R. Musse, D. Thalmann, Hierarchical model for real time simulation of virtual human crowds, IEEE Transactions on Visualization and Computer Graphics 7 (2001) 152–164. doi:10.1109/2945.928167.

[17] B. Ulicny, D. Thalmann, Crowd simulation for interactive virtual environments and VR training systems, Computer Animation and Simulation 2001 (2001) 163–170.

[18] N. Farenc, S. R. Musse, E. Schweiss, M. Kallmann, O. Aune, R. Boulic, D. Thalmann, One step towards virtual human management for urban environment simulation, in: Proceedings of the ECAI workshop on intelligent user interfaces, Vol. 3, Citeseer, 1998.

[19] T. Abaci, J. Ciger, D. Thalmann, Planning with smart objects, in: Proceedings of the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision: WSCG, 2005, pp. 25–28.

[20] M. Kapadia, S. Singh, G. Reinman, P. Faloutsos, A behavior-authoring framework for multiactor simulations, Computer Graphics and Applications, IEEE 31 (6) (2011) 45–55.

[21] J. M. Allbeck, Carosa: A tool for authoring npcs, in: Motion in Games, Springer, 2010, pp. 182–193.

[22] C. Stocker, L. Sun, P. Huang, W. Qin, J. M. Allbeck, N. I. Badler, Smart events and primed agents, in: Intelligent Virtual Agents, Springer, 2010, pp. 15–27.

[23] E. Bouvier, E. Cohen, L. Najman, From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation, Journal of Electronic Imaging 6 (1) (1997) 94–107. doi:10.1117/12.261175.

[24] D. Helbing, P. Molnar, Self-organization phenomena in pedestrian crowds, Arxiv preprint cond-mat/9806152.

[25] D. Helbing, I. Farkas, T. Vicsek, Simulating dynamical features of escape panic, Nature, Vol. 407, pp. 487-490, 2000doi:10.1038/35035023.

[26] A. Braun, S. R. Musse, L. de Oliveira, B. Bodmann, Modeling individual behaviors in crowd simulation, in: Computer Animation and Social Agents, 2003. 16th International Conference on, IEEE, 2003, pp. 143–148. doi:10.1109/CASA.2003.1199317.

[27] N. Pelechano, J. Allbeck, N. Badler, Controlling individual agents in high-density crowd simulation, in: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, Eurographics Association, 2007, pp. 99–108.

[28] J. Kessing, T. Tutenel, R. Bidarra, Designing semantic game worlds, in: PCG 2012: Proceedings of the 3rd workshop on Procedural Content Generation for Games, 2012.

[29] T. Tutenel, R. M. Smelik, K. J. de Kraker, R. Bidarra, Using semantics to improve the design of game worlds, in: Proceedings of AIIDE '09, the 5^{th} Conference on Artificial Intelligence and Interactive Digital Entertainment, Stanford, CA, USA, 2009.

[30] T. Tutenel, R. Bidarra, R. M. Smelik, K. J. de Kraker, A semantic scene description language for procedural layout solving problems, in: Proceedings of AIIDE '10, the 6^{th} Conference on Artificial Intelligence and Interactive Digital Entertainment, Stanford, CA, USA, 2010.

[31] T. Tutenel, R. M. Smelik, R. Lopes, K. J. de Kraker, R. Bidarra, Generating consistent buildings: a semantic approach for integrating procedural techniques, IEEE Transactions on Computational Intelligence and AI in Games 3 (3) (2011) 274–288.

[32] D. de Paiva, R. Vieira, S. R. Musse, Ontology-based crowd simulation for normal life situations, in: Proceedings of Computer Graphics International Conference, IEEE Computer Society, Los Alamitos, CA, USA, 2005, pp. 221–226. doi:10.1109/CGI.2005.1500421.

[33] M. E. Bratman, Intention, Plans, and Practical Reason, Harvard University Press, 1987.

[34] R. Geraerts, Planning short paths with clearance using explicit corridors, in: Proceedings of Robotics and Automation (ICRA), 2010 IEEE International Conference on, IEEE, 2010, pp. 1997–2004.

[35] N. Kraayenbrink, J. Kessing, T. Tutenel, G. de Haan, F. Marson, S. R. Musse, R. Bidarra, Semantic crowds: reusable population for virtual worlds, in: Proceedings of VS-GAMES 2012 - 4th International Conference on Games and Virtual Worlds for Serious Applications, Procedia Computer Science 15C, Genoa, Italy, 2012, pp. 122–139. doi:10.1016/j.procs.2012.10.064.