Authoring Multi-Actor Behaviors in Crowds with Diverse Personalities

Mubbasir Kapadia, Alexander Shoulson, Funda Durupinar, and Norman I. Badler

Abstract Multi-actor simulation is critical to cinematic content creation, disaster and security simulation, and interactive entertainment. A key challenge is providing an appropriate interface for authoring high-fidelity virtual actors with feature-rich control mechanisms capable of complex interactions with the environment and other actors. In this chapter, we present work that addresses the problem of behavior authoring at three levels: Individual and group interactions are conducted in an event-centric manner using parameterized behavior trees, social crowd dynamics are captured using the OCEAN personality model, and a centralized automated planner is used to enforce global narrative constraints on the scale of the entire simulation. We demonstrate the benefits and limitations of each of these approaches and propose the need for a single unifying construct capable of authoring functional, purposeful, autonomous actors which conform to a global narrative in an interactive simulation.

1 Introduction

Multi-actor simulation is a critical component of cinematic content creation, disaster and security simulation, and interactive entertainment. Depending on the application, a simulation may involve two or three actors interacting in complex ways, a group of actors participating in an event, or a large crowd with hundreds and thou-

Mubbasir Kapadia
University of Pennsylvania e-mail: mubbasir@seas.upenn.edu

Alexander Shoulson
University of Pennsylvania e-mail: shoulson@seas.upenn.edu

Funda Durupinar
University of Pennsylvania e-mail: fundad@seas.upenn.edu

Norman I. Badler
University of Pennsylvania e-mail: badler@seas.upenn.edu

sands of actors. For example, a user may want to author huge armies in movies, the repercussions of a car accident in a busy city street, the reactions of a crowd to a disturbance, a virtual marketplace with buyers and vendors haggling for prices, and thieves that are on the lookout for stealing opportunities. The existing baseline for multi-actor simulations consists of numerous relatively independent walking pedestrians. While their visual appearances may be quite varied, their behavioral repertoire is not, and their interactions are generally limited to attention control and collision avoidance. The next generation of interactive virtual world applications require functional, purposeful, heterogeneous actors with individual personalities and desires, while exhibiting complex group interactions, and conforming to global narrative constraints.

Readily authoring such complex multi-actor situations is an open problem. Existing techniques are often a bottleneck in the production process, requiring the author to either manually script every detail in an inflexible way or to provide a higher level description that lacks appropriate control to ensure correct or interesting behavior. The challenge is to provide a method of authoring that is intuitive, simple, automatic, yet has enough expressive power to control details at the appropriate level of abstraction. In this chapter, we present work that addresses the problem of behavior authoring at three levels. Our goal is to explain these levels and construct feasible and authorable computational models of when and how they interact.

First, we present a method to capture social crowd dynamics by mapping low-level simulation parameters to the OCEAN personality model. Each personality trait is associated with nominal behaviors – facilitating a plausible mapping of personality traits to existing behavior types. We validate our mapping by conducting a user study which assesses the perception of personality traits in a variety of crowd simulations demonstrating these behaviors [1].

Second, we describe a framework for authoring background characters using an event-centric control model, which shifts behavior authoring from writing complex reactive agents to defining particular activities. Interactions between groups of actors are defined using parameterized behavior trees, and a centralized *Group Coordinator* dispatches events to agents based on their situational and locational context, while satisfying a global distribution of events that is user specified [2, 3, 4].

Third, we present a multi-actor planning framework for generating complicated behaviors between interacting actors in a user-authored scenario. Users define the state and action space of actors and *specialize* existing actor definitions to add variety and purpose to their simulation. Actors with dependent goals are grouped together into a set of independent composite domains. For each of these domains, a multi-actor planner generates a trajectory of actions for all actors to meet the desired behavior. We author and demonstrate a simulation of more than one hundred pedestrians and vehicles in a busy city street and inject heterogeneity and drama into our simulation using specializations [5].

The rest of this document is articulated as follows. Section 2 reviews prior work in behavior authoring for interactive virtual characters. Section 3 describes the use of the OCEAN personality model to capture social crowd dynamics. Section 4 presents an event-centric paradigm for authoring multi-actor interactions, and Section 5 pro-

poses the use of domain-independent planning for behavior generation. Finally, Section 6 discusses the comparative benefits and limitations of each of these approaches, and proposes the need for a single unifying construct capable of authoring functional, purposeful, autonomous actors which conform to a global narrative in an interactive simulation.

2 Related Work

Behavioral animation in crowds has been studied extensively from many different perspectives [6] which can be broadly classified into three overlapping categories: (1) steering based approaches, (2) cognitively based approaches and, (3) narrative driven approaches. Many implementations blend aspects of these three categories – steering-based models in particular are often used in concert with one of the two other approaches. However, since each model has a very different approach to agent control and motivation it is difficult to evenly incorporate all three.

Steering based approaches. These techniques focus on agent movement with a focus on collision avoidance and trajectory planning. Centralized techniques [7, 8, 9, 10] focus on the system as a whole, modeling flow characteristics rather than individual pedestrians. Particle based approaches [11, 12] simulate agents using particle dynamics. Social force based approaches [13, 14, 15, 16] simulates physical as well as psychological forces between steering agents. Cellular Automata models [17, 18, 19] simulate agents defined as mathematical idealizations for physical systems in which space and time are discretized. Rule-based approaches [20, 21, 22] use carefully designed conditions and heuristics to define agent behavior. Data-driven methods [23, 24] use real world data to derive steering choices. The works of [25, 26, 27] use predictions in the space-time domain to perform steering in environments populated with dynamic threats. Local field methods [28, 29] uses egocentric fields to model agent affordances and recent work [30] demonstrates a synthetic vision-based approach for steering.

Cognitively based approaches. These techniques populate virtual worlds with rich individual agents which sense the environment and other agents, and act based on personalized desires, motivations, and other attributes such as mood and emotions. Agent decision-making is simulated using a wide variety of cognitively based models such as decision networks [31], neural networks [32], partially-observable markov decision problems [33], fuzzy logic [34], hierarchical state machines [35], scripts [36, 37, 38], and planners [39]. These models capture domain specific knowledge, effectual actions, and personal agent goals to simulate functional, purposeful autonomous agents [40]. Several studies represent individual differences through psychological states [41, 16]. The OCEAN personality model [42] and the OCC emotion model [43] are commonly used in the simulation of autonomous agents. Such models aid to improve believability of embodied conversational char-

acters [44, 45] as well as agents in a crowd [46].

Narrative driven approaches. These systems orchestrate the behavior of actors in a scene from a global scope, dictating actions to participants based on the needs of the scenario constraints rather than the individual agents' motivations. Drama Managers [47] are used weave a story around the actions of a player and the principal actors in the environment. Director-based systems such as Facade [48], Thespian [49], Mimesis [50], and others act upon a small number of high-dimensional agents representing principal characters in the simulation. These systems can be controlled by a planning approach [51], using actions and preconditions as a dynamic script for the intended plot. Smart Events [52] externalize behavior logic to authored events that occur in the environment. Unlike cognitively-driven simulations, virtual actors respond to impulses sent by a central controller responsible for enforcing the constraints of a global narrative system.

3 The Impact of the OCEAN Personality Model on the Perception of Crowds

Personality is the sum of a persons behavioral, temperamental, emotional, and mental traits. A popular model that describes personality is the Five Factor, or OCEAN (openness, conscientiousness, extroversion, agreeableness, and neuroticism) model. The personality space is composed of these five orthogonal dimensions.

- *Openness* describes a dimension of personality that portrays the imaginative and creative aspect of human character. Appreciation of art, inclination towards going through new experiences and curiosity are characteristics of an open individual.
- *Conscientiousness* determines the extent to which an individual is organized, tidy and careful.
- *Extroversion* is related to the social aspect of human character.
- *Agreeableness* is a measure of friendliness, generosity and the tendency to get along with other people.
- *Neuroticism* refers to emotional instability and the tendency to experience negative emotions. Neurotic people tend to be too sensitive and they are prone to mood swings.

Each factor is bipolar and composed of several traits, which are essentially the adjectives that are used to describe people [53]. Some of the relevant adjectives describing each of the personality factors for each pole are given in Table 1.

We have mapped these trait terms to the low-level behavior parameters in the HiDAC (High-Density Autonomous Crowds) crowd simulation system. HiDAC models individual differences by assigning each person different psychological and physiological traits. Users normally set these parameters to model the non-uniformity and diversity of a crowd. Our approach frees users of the tedious task

O+	Curious, alert, informed, perceptive
O-	Simple, narrow, ignorant
C+	Persistent, orderly, predictable, dependable, prompt
C-	Messy, careless, rude, changeable
E+	Social, active, assertive, dominant, energetic
E-	Distant, unsocial, lethargic, vigorless, shy
A+	Cooperative, tolerant, patient, kind
A-	Bossy, negative, contrary, stubborn, harsh
N+	Oversensitive, fearful, dependent, submissive, unconfident
N-	Calm, independent, confident

Table 1 Trait-descriptive adjectives

of low-level parameter tuning by combining all these behaviors in distinct personality factors.

By incorporating a standard personality model to a high-density crowd simulation, our approach creates plausible variations in the crowd and enables novice users to dictate these variations. A crowd consists of subgroups with different personalities. Variations in the characteristics of subgroups influence emergent crowd behavior. The user can add any number of groups with shared personality traits and can edit these characteristics during the course of an animation.

In order to verify the plausibility of our mapping we have conducted tests that evaluate users' perception of the personality traits in the generated animations. We created several animations to examine how modifying the personality parameters of subgroups affects global crowd behavior. The animations exhibit the emergent behaviors of agents in scenarios in which the settings assigned according to the OCEAN model drive crowds behavior. In order to validate our system, we determined the correspondence between our mapping and the users perception of these trait terms in the videos. The results indicate a high correlation between our parameters and the participants perception of them.

3.1 Personality-to-Behavior Mapping

A crowd is composed of subgroups with different personalities. Variations in the characteristics of the subgroups influence emergent crowd behavior. The user can add any number of groups with shared personality traits and can edit these characteristics during the course of an animation. An agent's personality π is a five-dimensional vector, where each dimension is represented by a personality factor, ψ_i. The distribution of the personality factors in a group of individuals is modeled by a Gaussian distribution function N with mean μ_i and standard deviation σ_i:

$$\pi = < \psi_O, \psi_C, \psi_E, \psi_A, \psi_N > \tag{1}$$

$$\psi_i = N(\mu_i, \sigma_i^2),\ for\ i \in \{O,C,E,A,N\}, \tag{2}$$

where $\mu \in [0,1]$ and $\sigma \in [-0.1,0.1]$.

An individual's overall behavior β is a combination of different behaviors. Each behavior is a function of personality as:

$$\beta = (\beta_1, \beta_2, \ldots, \beta_n) \tag{3}$$

$$\beta_j = f(n),\ for\ j = 1, \ldots, n \tag{4}$$

$$\tag{5}$$

Since each factor is bipolar, ψ can take both positive and negative values. For instance, a value of 1 for extroversion means that the individual has extroverted character; whereas a value of -1 means that the individual is highly introverted.

By analyzing the meaning and usage of each low-level parameter and built-in behavior in the HiDAC model, we characterize these by the adjectives that are used to describe personalities. Thus, we devise a mapping between the agents' personality factors (adjectives) and the HiDAC parameters, as shown in Table 2. A positive factor takes values in the range $[0.5,1]$, whereas a negative factor takes values in the range $[0,0.5)$. A factor given without any sign indicates that both poles apply to that behavior. For instance E+ for a behavior means that only extroversion is related to that behavior; introversion is not applicable. As indicated in Table 2, a behavior can be defined by more than one personality dimension. The more adjectives of a certain factor defined for a behavior, the stronger is the impact of that factor on that behavior. We assign a weight to the factor's impact on a specific behavior. The sum of the weights for a specific type of behavior is 1. In order to understand how the mapping from a personality dimension to a specific type of behavior is performed, we explain four representative mappings. The remaining ones are mathematically similar.

Right preference. When the crowd is dispersed, individuals tend to look for avoidance from far away and they prefer to move towards the right hand side of the obstacle they are about to face. This behavior shows the individual's level of conformity to the rules A disagreeable or non-conscientious agent makes a right or left preference with equal probability, while the probability of choosing the right side increases with increase in values of agreeableness and conscientiousness. Given $P_i(Right) \propto A, C$ and $\beta_i^{Right} \in \{0,1\}$, right preference $P_i(Right)$ is computed as follows

$$P_i(Right) = \begin{cases} 0.5 & \text{if } \psi_i^A < 0\ or\ \psi_i^C < 0 \\ \omega_{AR}\psi_i^A + \omega_{CR}\psi_i^C & \text{otherwise} \end{cases} \tag{6}$$

$$\beta_i^{Right} = \begin{cases} 1 \text{ if } P_i(Right) \geq 0.5 \\ 0 \text{ otherwise} \end{cases} \tag{7}$$

Personal space. Personal space determines the territory in which an individual feels comfortable. Agents try to preserve their personal space when they approach other agents and when other agents approach from behind. However, these two values are

Leadership	Dominant, assertive, bossy, dependable, confident, unconfident, submissive, dependent, social, unsocial	E, A-, C+, N
Trained/not trained	Informed, ignorant	O
Communication	Social, unsocial	E
Panic	Oversensitive, fearful, calm, orderly, predictable	N, C+
Impatience	Rude, assertive, patient, stubborn, tolerant, orderly	E+, C, A
Pushing	Rude, kind, harsh, assertive, shy	A, E
Right preference	Cooperative, predictable, negative, contrary, changeable	A, C
Avoidance /personal space	Social, distant	E
Waiting radius	Tolerant, patient, negative	A
Waiting timer	Kind, patient, negative	A
Exploring environment	Curious, narrow	O
Walking speed	Energetic, lethargic, vigorless	E
Gesturing	Social, unsocial, shy, energetic, lethargic	E

Table 2 Low-level parameters vs. trait-descriptive adjectives

not the same. According to the research on Western cultures, the average personal space of an individual is found to be 0.7 meters in front and 0.4 meters behind [54]. Given $\beta_i^{PersonalSpace} \propto^{-1} E$ and $\beta_i^{PersonalSpace} \in \{0.5, 0.7, 0.8\}$, the personal space of an agent i with respect to an agent j is computed as follows

$$\beta_{i,j}^{PersonalSpace} = \begin{cases} 0.8\, f(i,j) \text{ if } \psi_i^E \in [0, \frac{1}{3}) \\ 0.7\, f(i,j) \text{ if } \psi_i^E \in [\frac{1}{3}, \frac{2}{3}] \\ 0.5\, f(i,j) \text{ if } \psi_i^E \in (\frac{2}{3}, 1] \end{cases} \tag{8}$$

$$f(i,j) = \begin{cases} 1 & \text{if } i \textit{ is behind } j \\ \frac{0.4}{0.7} & \text{otherwise} \end{cases} \tag{9}$$

Waiting radius. In an organized situation, individuals tend to wait for space available before moving. This waiting space is called the waiting radius and it depends on the kindness and consideration of an individual, i.e., the agreeableness dimension. Given $\beta_i^{WaitingRadius} \propto A$ and $\beta_i^{WaitingRadius} \in \{0.25, 0.45, 0.65\}$. , the waiting radius is computed as follows

$$\beta_{i,j}^{WaitingRadius} = \begin{cases} 0.25 \text{ if } \psi_i^A \in [0, \frac{1}{3}) \\ 0.45 \text{ if } \psi_i^A \in [\frac{1}{3}, \frac{2}{3}] \\ 0.65 \text{ if } \psi_i^A \in (\frac{2}{3}, 1] \end{cases} \tag{10}$$

Walking speed. The maximum walking speed is determined by an individual's energy level. As extroverts tend to be more energetic while introverts are more lethargic, this parameter is controlled by the extroversion trait. Given $\beta_i^{WalkingSpeed} \propto E$ and $\beta_i^{WalkingSpeed} \in [1, 2]$, the walking speed is computed as follows

$$\beta_i^{WalkingSpeed} = \psi_i^E + 1, \tag{11}$$

3.2 User Studies on Personality

In order to evaluate if the suggested mappings are correctly perceived, we conducted user studies. We created several animations to see how global crowd behavior is affected by modifying the personality parameters of subgroups. Some of these animations can be found at `http://cg.cis.upenn.edu/hms/research/Ocean/`.

3.2.1 Experiment Design

We created 15 videos presenting the emergent behaviors of people in various scenarios where the crowds' behavior is driven by the settings assigned through the OCEAN model. We performed the mapping from HiDAC parameters to OCEAN factors by using trait-descriptive adjectives. We determined the correspondence between our mapping and the users' perception of these trait terms in the videos in order to validate our system. 70 subjects (21 female, 49 male, ages 18-30) participated in the experiment. We showed the videos to the participants through a projected display and asked them to fill out a questionnaire consisting of 123 questions– about eight questions per video. The videos were shown one by one; after each video, participants were given some time to answer the questions related to the video. The participants did not have any prior knowledge about the experiment. Questions assessed how much a person agreed with statements such as "I think the people in this video are kind." or "I think the people with black suits are calm." We asked questions that included the adjectives describing each OCEAN factor instead of asking directly about the factors because we assumed that the general public might be unfamiliar with the OCEAN model. Participants chose answers on a scale from 0 to 10, where 0 = totally disagree, 5 = neither agree nor disagree, and 10 = totally agree. We omitted the antonyms from the list of adjectives for the sake of conciseness. The remaining adjectives were *assertive*,*calm*, *changeable*, *contrary*, *cooperative*, *curious*, *distant*, *energetic*, *harsh*, *ignorant*, *kind*, *orderly*, *patient*, *predictable*, *rude*, *shy*, *social*, *stubborn*, and *tolerant*.

3.2.2 Sample Scenarios

A sample scenario testing the impact of openness took place in a museum setting as one of the key factors determining openness is the belief in the importance of art. Figure 1 (a) shows a screenshot from the sample animation. We tested the adjectives curiosity and ignorance with this scenario. There were three groups of people, with

openness values of 0, 0.5, and 1. We mapped the number of tasks that each agent must perform to openness, with each task requiring looking at a painting. The least open agents (with blue hair) left the museum first, followed by the agents with openness values of 0.5 (with black hair). The most open agents (with red hair) stayed the longest.

In order to test whether the personalities of people creating congestion are distinguished, we showed the participants two videos of same duration and asked them to compare the characteristics of the agents in each video. Each video consisted of two groups of people moving through each other. The first video showed people with high agreeableness and conscientiousness values($\mu = 0.9$ and $\sigma = 0.1$ for both traits), whereas the second video showed people with low agreeableness and conscientiousness values($\mu = 0.1$ and $\sigma = 0.1$ for both traits). In the first video, groups managed to cross each other while in the second video congestion occurred after a fixed period of time. Such behaviors emerged since agreeable and conscientious individuals are more patient; they dont push each other and are always predictable, as they prefer to move on the right side. Figure 1 (b) shows how congestion occurred due to low conscientiousness and agreeableness. People were stuck at the center and refused to let other people move. They were also *stubborn*, *negative*, and not *cooperative*.

Another video assessed how extroverts and introverts were perceived according to their distribution around a point of attraction. Figure 1 (c) shows a screenshot from the video in which the agents in blue suits are extroverted ($\mu = 0.9$ and $\sigma = 0.1$) and those in grey suits are introverted ($\mu = 0.1$ and $\sigma = 0.1$). At the end of the animation, introverts were left out of the ring structure around the object of attraction. Because extroverts are faster, they approached the attraction point in less time. In addition, when other agents blocked their way, they tended to push them to reach their goal. The figure also shows the difference between the personal spaces of extroverts and introverts. This animation tested the adjectives, *social*, *distant*, *assertive*, *energetic*, and *shy*.

Figure 1 (d) shows a screenshot from the animation demonstrating the effect of neuroticism, non-conscientiousness and disagreeableness on panic behavior. Five of the 13 agents had neuroticism values of $\mu = 0.9$ and $\sigma = 0.1$, conscientiousness values of $\mu = 0.1$ and $\sigma = 0.1$ and agreeableness values of $\mu = 0.1$ and $\sigma = 0.1$. The other agents, which are psychologically stable, have neuroticism values of $\mu = 0.1$ and $\sigma = 0.1$, conscientiousness values of $\mu = 0.9$ and $\sigma = 0.1$ and agreeableness values of $\mu = 0.9$ and $\sigma = 0.1$. The agents in black suits are neurotic, less conscientious, and disagreeable. The figure shows that they tend to panic more, push other agents, force their way through the crowd, and rush to the door. They are not *predictable*, *cooperative*, *patient*, or *calm* but they are *rude*, *changeable*, *negative*, and *stubborn*.

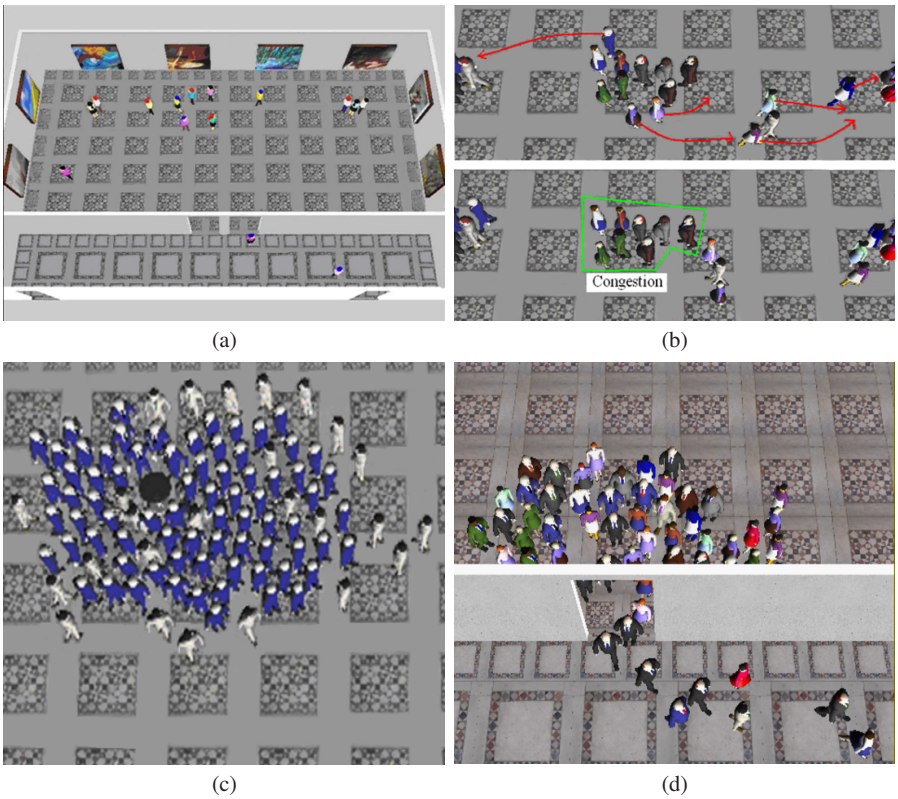


Fig. 1 Snapshots of a crowd simulation authored using our framework: (a) Openness tested in a museum. The most open people (red-heads) stay the longest, whereas the least open people (blue-heads) leave the earliest. (b) People with low conscientiousness and agreeableness values cause congestion.(c) Ring formation where extroverts (blue suits) are inside and introverts are outside. (d) Neurotic, non-conscientious and disagreeable agents (in black suits) show panic behavior.

3.2.3 Analysis

After collecting the participants' answers for all the videos, we first organized the data for the adjectives. Each adjective is classified by its question number, the actual simulation parameter and the participants' answers for the corresponding question. We calculated the Pearson correlation (r) between the simulation parameters and the average of the subjects' answers for each question.

We grouped the relevant adjectives for each OCEAN factor to assess the perception of personality traits. The evaluation process is similar to the evaluation of adjectives; this time considering the questions for all the adjectives corresponding to an OCEAN factor. For instance, as openness is related to curiosity and ignorance, we took into account the adjectives *curious* and *ignorant*. Again, we averaged the

subjects' answers for each question. Then, we computed the correlation with the parameters and the mean throughout all the questions inquiring *curious* and *ignorant*.

We computed the significance of the correlation coefficients as $1 - p$, where p is the two-tailed probability that is calculated considering the sample size and the correlation value. Higher correlation and significance values suggest more accurate user perception.

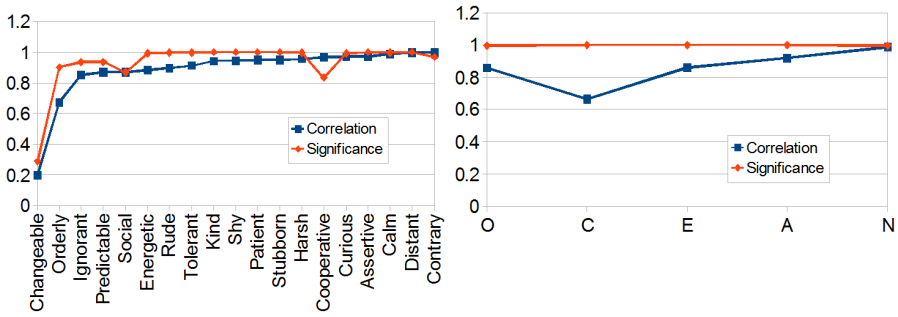
3.2.4 Results

Figure 2(a) depicts the correlation coefficients and significance values for the adjectives. Significance is low (< 0.95) for *changeable*, *orderly*, *ignorant*, *predictable*, *social* and *cooperative*. Low significance is caused by low correlation values for *changeable* and *orderly*. However, although the correlation coefficients are found to be high for *predictable*, *ignorant*, *social* and *cooperative*, low significance can be explained due to small sample size. From the participants' comments, we determined that *changeable* is especially confusing because the participants identified non-conscientious agents as rude but perceived them as persistent in their rudeness.

Orderly is another weakly correlated adjective. Analyzing the results for each video, we found that agents in the evacuation drill scenario were perceived to be orderly although they displayed panic behavior. In these videos, even if the agents pushed each other and moved fast, some kind of order could be observed. This was due to the smooth flow of the crowd during building evacuation. Although people were impatient and rude, the overall crowd behavior appeared orderly. On the other hand, in a scenario showing queuing behavior in front of a water dispenser, the participants could easily distinguish orderly agents from disorderly ones. Orderly agents waited at the end of the queue, whereas disorderly agents rushed to the front. In this scenario, although the main goal was the same for all the agents (drinking water), there were two distinguishable groups that acted differently.

Figure 2 (b) shows the correlation coefficients and their significance for the OCEAN parameters. These values are computed by taking into account all the relevant adjectives for each OCEAN factor. All the coefficients have high significance, with a probability of less than 0.5% of occurring by chance ($p < 0.005$). The significance is high because all the adjectives describing a personality factor are taken into account, achieving sufficiently large sample size.

The correlation coefficient for conscientiousness is comparatively low, showing that the participants correctly perceived only approximately 44% of the traits($r^2 \approx 0.44$). Low correlation values for *orderly* and *changeable* reduce the overall correlation. If we consider only *rude* and *predictable* for conscientiousness, correlation increases by 18.6%. The results suggest that people can observe the politeness aspect in short-term crowd behavior settings more easily than the organizational aspects. This observation also explains why the perception of agreeableness is highly correlated with the actual parameters.



(a) (b)

Fig. 2 (a) The correlation coefficients between the parameters and the subjects answers for the descriptive adjectives (blue), and the significance values for the corresponding correlation coefficients (orange). Significance is low (¡0.95) for changeable, orderly, ignorant, predictable, social, and cooperative.(b)The correlation coefficients between actual parameters and subjects answers for the Ocean factors (blue), and the two-tailed probability values for the corresponding correlation coefficients (orange). All the coefficients have high significance.

Figure 2 also shows that the participants perceived neuroticism the best. In this study, we have only considered the calmness aspect of neuroticism, which is tested in emergency settings and building evacuation scenarios.

4 Coordinating Agent Interactions with Behavior Events

When two actors interact in a virtual world, they must coordinate tasks and exchange information. Managing this complexity is difficult when designing the behavior of each actor in isolation. For sophisticated cooperative or competitive behavior, an actor must constantly communicate and perform actions dependent on both its own and other actors' current state. This call-and-response type of interchange is traditionally authored in pieces across multiple actors, with certain steps anticipating the behavior of another actor involved in the interaction. If a cooperative or competitive behavior involves actors taking on certain roles (such as "leader/follower"), participating actors must negotiate the nature of their participation in the interaction, which further complicates the behavior authoring process.

Consider two agents participating in a transaction involving bargaining over a piece of merchandise. In a localized agent-centric model, the interaction begins when the buyer, B, has a desire for an item that is sold by the seller, S. B approaches S's market stall and displays a greeting animation. S is notified that B played a greeting animation and has to recognizes that B wants to buy something that he (S) has for sale in order to begin the bargaining process. Throughout the interaction, the agents transmit notifications to one another about which animations were played and the current price negotiated. They must regularly receive these notifications and inter-

pret the mode and mood of the interaction, maintaining the state of the conversation and the currently negotiated price, as well as the item in question when deciding on an appropriate response. For a simple sequence of animations, this exhibits a high level of complexity that must be duplicated and maintained in both agents' state. In contrast, suppose a centralized data structure could coordinate these agents instead. The centralized structure selects B and S out of a pool of available agents, instructs B to approach S, dispatches the appropriate animations to each in sequence, and maintains centralized information such as the current mood of the conversation, the item being haggled over, and the current offers from both parties. Moreover, this centralized structure can be invoked immediately to involve the two agents in this kind of interaction, rather than waiting for the whim of actor B to decide that he suddenly wants to purchase an item from S.

This centralized data structure simplifies the process of designing interactions between actors, using a behavior paradigm we call event-centric authoring. Rather than requiring multiple actors to handle the responsibilities of message passing and stimuli response, a behavior event consists as a body of centralized control logic with its own state and unrestricted access to the actor(s) involved in the event. Participating actors temporarily suspend their own autonomy and are controlled entirely by the event structure, which treats them as limbs of a central entity for the duration of its execution. A conversation could be conducted with a central event instructing the two actors to approach one another and then take turns playing the requisite sounds and gesture animations. When the event is completed, or fails, the involved actors resume their own autonomy until co-opted to participate in another event. Events also define roles for their participants that must be filled on instantiation by a particular actor type. For example, an event for a transaction between two actors could may stipulate that the seller be of type "Merchant Actor". Events exist in a system to augment the richness of available behavior, and need not detract from an actor's own individuality. Actors can still retain rich autonomous behaviors and only occasionally be involved in higher-order events.

4.1 Parameterized Behavior Trees

Though they could be designed with any suitable method, we create events using Parameterized Behavior Trees (PBTs) [4], an expansion on standard behavior tree models [55, 56] with a specialized data flow architecture to handle multiple agents and shared state data. PBTs, and behavior trees in general, represent a flexible graphical programming language with explicit goal direction that is easy to visualize for complex behavior structures. They contain an inherent hierarchical structure that makes them easy to visualize and conceptualize at a macroscopic level. In general, a subtree represents a goal at its root, and the means by which that goal can be achieved with its leaves. This allows for implicit documentation within the tree structure, so that certain branches of the tree can be understood by the goals they attempt to accomplish, without the need to necessarily expose any of their children.

The core mechanic of behavior trees in general is the success or failure of each node in the tree. Each node attempts to execute an action and reports success or failure to its parent node. The parent can use that information for selecting the next node to execute. Sequence nodes execute each child in order. If one child of a sequence node fails, that sequence reports failure to its parent and ceases execution. Sequence nodes succeed when all of their children succeed. Conversely, selector nodes cease execution and report success if any one of their children succeeds. A selector node reports failure only if all of its children fail. Figure 3 represents a simple behavior

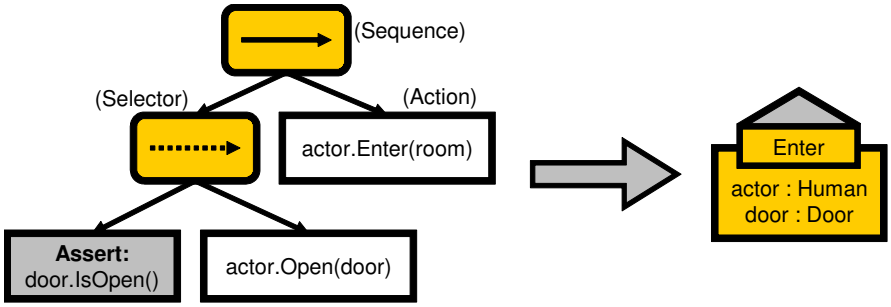


Fig. 3 An encapsulated behavior for entering a room through a door.

tree for opening a door and entering a room. If the assertion succeeds, the selector will propagate success and skip over the door opening action. Otherwise, if the assertion fails, the selector will attempt to perform its next child, which consists of an action opening a door. Assuming one of the two actions is successful, the selector will report success to the root sequence node, which will then execute its next action directing the actor to enter the opened room.

Unlike traditional behavior trees, PBTs are designed with data fields that take on values at runtime and propagate information through the structure of the tree, eventually moving through the tree's action or assertion leaf nodes to the underlying functions those leaves invoke. These data can be targets for low-level character controller, or flags that affect branching decisions within the tree itself. With parameterization, subtrees can be encapsulated for reuse with parameters by multiple PBTs, not unlike a subroutine in a traditional programming language. The right side of Figure 3 shows how the simple door behavior tree can be encapsulated into a generic PBT with the typed parameters "actor" and "door" that take on values at runtime. The single node created by this encapsulation can be accessed by means of a lookup node, which acts as a placeholder for the entire subtree and fills the parameter fields with object references at runtime. Encapsulated subtrees can also act polymorphically, so that different actor types can "implement" a given subtree signature in multiple different ways. When an event invokes an actor's capabilities, or tells an actor to execute a specific subtree with certain parameters, the actual implementation of those actions can still be personalized to the actor or that actor's type.

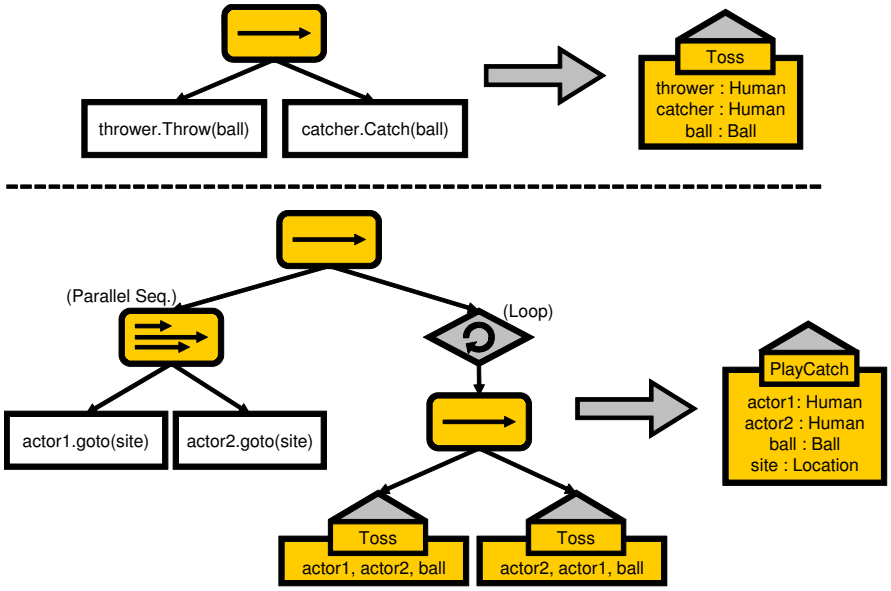


Fig. 4 Composing an event for two actors to play catch.

Parameterized subtrees that take multiple actors as parameters define the logic for behavior events. Figure 4 displays an event for two actors playing a game of catch at different levels of the hierarchy. First we define a simple sub-event taking two actors and animating one tossing a ball to another. This sub-event is encapsulated as a "Toss" tree, with two actor parameters and an object parameter. Next, we define the actual "PlayCatch" event. This event directs the two actors to approach a location (given as a parameter when this tree is instantiated), and loops the actors playing the "Toss" sub-event. Observe that the event reverses the actor roles in the "Toss" subtree so that the ball passes back and forth between actors.

The "PlayCatch" subtree is also encapsulated with parameters, so the "PlayCatch" tree can be instantiated at runtime using entities in the environment for its actor or object roles. At any point during the simulation, a virtual director can select two actors and a location, provide those actors with a ball, and instantiate this event using those objects as parameters. The event will then completely manage the actors' behavior in a centralized manner as they perform the complicated interactions needed for playing a game of catch. This authoring approach avoids issues such as shared ownership of the ball item, since all state information for the event is maintained in the event's own state space, rather than in the state space of one of the actors. Note that the event does not own the ball or the actors involved, but does acquire control over them for the duration of the event. The ball, or any prop, may persist in the world or may be destroyed depending on the virtual director's decision.

In general, we use events for all complex interactive behavior, but actors are also independently controlled by separate PBTs for their autonomous actions. When an actor is not involved in an event, it executes its own internal behavior tree that locally invokes that actor's capabilities and modifies that actor's traits. When an actor is selected to participate in an event, that actor's internal tree is halted, and the actor's traits and capabilities are exposed to be modified and invoked by the PBT contained in the event. In general, events are designed to terminate after the implicit goal of the event is accomplished or fails, while actors' internal trees never terminate.

4.2 Selecting Actors for Events

Since events contain centralized behavior for interactions between actors, we expect scenario authors to design a library of events for a variety of interactions between actors of different types within an environment. Events accept parameters for the actors involved and any other modifiers that may affect the progression of the interaction between those actors, including the location in which it takes place. Since events work only on actor types, they are not designed for any one specific actor within the world. Instead, events are designed to operate generically on any selection of actors matching the interfaces expected by the event. At runtime, over the course of the simulation, we use a global director construct to determine which events should be performed in the world, and what actors should perform them.

When the global director decides that a specific event should be performed at a location in the world (a process we will discuss later), that director must select the actors to participate in that interaction. There are several details that factor into this decision. The first filter is the type of the actor. The structure of the event itself specifies that each of its roles can only be filled by an actor of a given hierarchical type, so the director can only select actors of that type when filling that role. Other filters for actor candidates may be more detailed. The director may select an actor for a role based on its traits, its relationships with other characters, or the history of events in which it has participated. These qualities of an actor may be fixed at initialization, or modifiable at runtime.

If a scenario has many events that select actors based on the actions they have performed during that simulation, then the simulation's actors exhibit a quality we call *progressive differentiation*. That is, actors begin as largely homogeneous characters in the world and are slowly differentiated by one another based on the actions they perform at the behest of the director [3]. This kind of differentiation is important because it matches the perception of a human user in the environment. To a user observing the simulation, actors are already undifferentiated because the user has just encountered them for the first time. The user only learns to distinguish actors' personalities based on the actions they perform and the manner in which they are performed. Much in the same way, the director personifies actors at runtime based on the actions it selects for them, and maintains that history to inform selection of actors for further events. An actor selected to tell a joke to a crowd may

have that history recorded, making the director more likely to select that actor again for another "joke" event, and progressively characterize that actor as a "comedian" character. Because this differentiation experience happens simultaneously for the director and a human user, selective differentiation accomplishes a key goal of facilitating the user's internal narrative for rationalizing the chain of events presented in the scenario.

4.3 Selecting Events to Perform

After the behavior authors for a given scenario create a library of potential events that can be performed in the environment, the centralized director needs a method to pick which events to execute, when, and where in the virtual world. There are many ways a director could be designed to do this, including planning approaches to capture causality in narrative, but we present a simple statistical model suitable for maintaining a variety of actions in the background of a scene. This mostly applies to undifferentiated characters acting as "extras" to add atmosphere to a location [2].

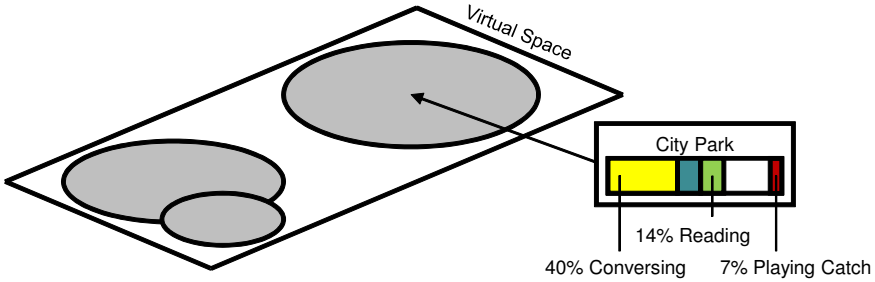


Fig. 5 Assigning regions in a virtual space, each with a desired actor activity distribution.

Areas in the environment are annotated as regions, which keep track of the actors currently present and the events in which they are involved. The scenario designer specifies a distribution of event classes that should be enforced within that given region for the actors present. Event classes may be broad categories such as "conversation", or "playing a game", and members of these classes may be more specific, such as "arguing over politics", or "playing catch with a ball". A desired event distribution specifies what percentage of the actors in that environment should be involved in events of a certain class. For instance, in a park, the author may designate that 15% of all actors in the park should be playing games with one another, while another 25% should be conversing. Figure 5 illustrates the process of assigning regions over a virtual space and specifying the event distribution for each.

We use a specialized director called a *Group Coordinator* to enforce these distributions. At a given point in time, the Group Coordinator inspects a region and

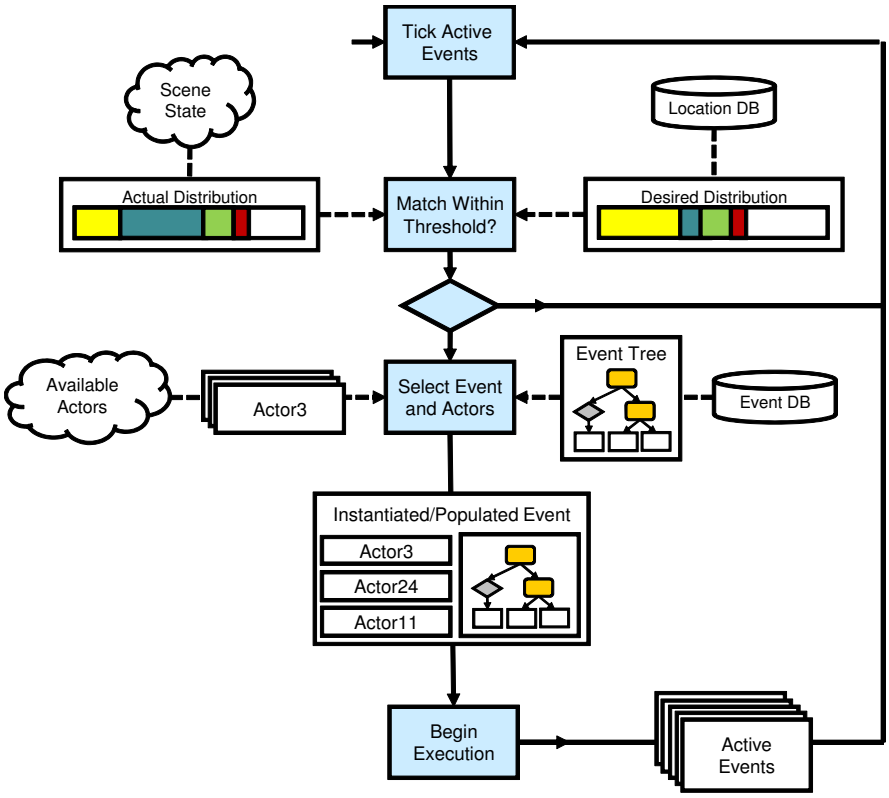


Fig. 6 The Group Coordinator's decision process for maintaining the distribution of events in a region.

determines the distribution density of the events being performed in that region at that instant. If the instantaneous calculated distribution differs significantly from the expected distribution specified by the scenario author, then the Group Coordinator must correct the error. The director accomplishes this by identifying the most under-represented event category, finding an event from the event library in that category, and invoking that new event on some suitable actors in or near that region. So long as there are suitable actors available to perform events in or near that region, the Group Coordinator will ensure that the actual distribution of events being performed will match the ideal distribution specified by the author within some threshold. This decision process is illustrated in Figure 6. The desired event distribution for a region can also be changed over time, reflecting a gradual shift caused by some phenomenon such as a day/night cycle.

To achieve the greatest event variety, events in this framework should not be causally linked. This technique is suitable for ambient activities performed by characters on which the user won't focus, but prominent actors in the environment must account for causality. If the user does begin to focus on one of these background

Fig. 7 A virtual middle eastern marketplace for demonstrating multi-actor interactions.

characters, then that character can be promoted to a principal character and managed by an event dispatch algorithm that does account for causality and differentiation. The goal of this algorithm is a simple interface to greatly affect the perceived variety of the activities being performed in a virtual space. Different regions also allow for different types of activity, where events suitable for a city park would not be appropriate in a movie theater.

We use the behavior trees from this Group Coordinator control process to implement a virtual middle eastern marketplace, as displayed in Figure 7. To accurately implement this setting, actors must be capable of interacting with one another in situations such as haggling, negotiating, and conversation. By default, actors maintain an individual "shopping list" of items they wish to procure from the stalls in the market. An actor's own tree directs that actor to move between stalls looking for items. When a stall has an item that an actor desires, the buyer and seller are placed in a negotiation event that directs them through gestures and movements representing two individuals haggling over an item. After the negotiation event ends, the buyer has either acquired the item or failed, and moves to the next stall on its shopping list. Periodically, actors are also selected by the Group Coordinator to stop and converse with one another, which suspends whatever else they were doing at the time (unless they were involved in a negotiation). This simple system captures the ambience of characters moving in the background and maintaining a baseline of activity in what should be a busy scene.

5 Authoring Complex Multi-Actor Interactions using Domain Independent Planning

This section presents a multi-actor planning framework [5, 57] for generating complicated behaviors between interacting actors in a user-authored scenario. Users define the state and action space of actors and *specialize* existing actor definitions to add variety and purpose to their simulation. Actors with dependent goals are grouped together into a set of independent composite domains. For each of these domains, a multi-actor planner generates a trajectory of actions for all actors to meet the desired behavior. We author and demonstrate a simulation of more than one hun-

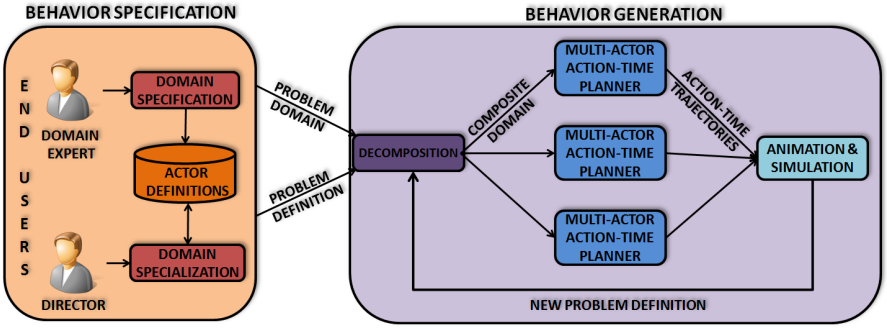


Fig. 8 An overview of the framework.

dred actors (pedestrians and vehicles) in a busy city street and inject heterogeneity and drama into our simulation using specializations.

Figure 8 presents an overview of our framework. First, a domain expert defines the problem domain of the actors in the scenario (domain specification). Next, a director specializes the actors using modifiers, constraints, and behaviors (domain specialization). Actors with dependent goals or constraints enforcing their interaction are grouped into a composite domain, forming a set of independent domains. For each domain, a multi-actor planner generates a trajectory of actions that satisfies the composite goal while optimizing each actors objective. Each searchs results become part of a global plan, which generates the resulting simulation.

5.1 Behavior Specification

Domain Specification. Domain specification is the lowest abstraction level for authoring behaviors. It involves defining the state space and action space for all actors in a scenario. Each actor has a state and can affect the state of itself or others through actions. Different actors in the same scenario might have different domain specifications. For example, we can define a traditional actor to simulate a pedestrian and define the environment as an actor that can be used to trigger global events, such as natural disasters.

1. **The state space**. We represent an actors state space using metrics - physical or abstract properties that are affected by actions. Users can extend metrics by applying operators to existing metrics to provide an intuitive understanding of the simulations properties. We denote the space of metrics for all actors in the scenario as mi.
2. **The action space.** The action space is a set of actions an actor can perform to modify its state and the state of other actors. An action has three properties: preconditions that determine whether the action is possible in a given state, the

Action *actionName* (*parameters*) {
Precondition: *conditions on elements of* $\{m_i\}$ *or* $\{c_i\}$
Effect: *effects on elements of* $\{m_i\}$
Cost Effect: *effects on elements of* $\{c_i\}$
}

Modifier *modifierName* {
Precondition: *conditions on elements of* $\{m_i\}$ *or* $\{c_i\}$
Effect: *effects on elements of* $\{m_i\}$ *or* $\{c_i\}$
}

Constraint *modifierName* {
Precondition: *conditions on elements of* $\{m_i\}$ *or* $\{c_i\}$
Constraint: *conditions on elements of* $\{m_i\}$
}

Behavior *behaviorName* {
Precondition: *conditions on elements of* $\{m_i\}$
Goal: *conditions on elements of* $\{m_i\}$
Objective Function: *values of* $\{w_i\}$
}

Fig. 9 Domain Specification and Specialization. Actions, modifiers, constraints and, behaviors defined using our framework.

actions effect on the state of the actor and target actors, and the cost of executing the action.

3. **Costs.** Costs are a numerical measure of executing an action. Different actions can affect different cost metrics by different amounts. Examples of cost metrics include distance and energy. We denote the space of costs as ci.

Domain Specialization. Users can reuse actor definitions across different scenarios by specializing actors in a state-dependent manner without modifying the original definition. This allows authors to specify and generate vastly different, purposeful simulations intuitively, with minimal specification. We provide three ways to specialize actors: effect modifiers, cost modifiers, and constraints.

1. **Modifiers.** Modifiers specialize the effects and costs of actions in a state-dependent manner. For example, users can place an effect modifier on elderly actors to reduce their normal speed of movement. Cost modifiers indicate what actions are in an actors best interest at a particular state. For example, users can author a cautious actor by increasing the cost of actions that might place the actor in danger (for example, entering a burning building). Here, the notion of danger would be a user-specified metric in the actors state space.
2. **Constraints.** Constraints enforce strict requirements on actors; they can prune the choices of an actor in a particular state. For example, constraints can prevent pedestrians from walking on the road or from disobeying traffic signals. Users can also place constraints on the simulations trajectory to author specific events

(for example, two cars must collide), generate complex interactions between actors, and direct high-level stories.

Behavior State Machine Specification. A behavior state defines an actors current goal and objective function. The goal is a desired state the actor must reach; the objective function is a weighted sum of costs the actor must optimize. Users can define multiple behaviors for an actor that depend on its current state.

5.2 Behavior Generation

The entire problem domain is decomposed into a set of independent composite domains where actors with dependent goals or constraints enforcing their interaction are part of one composite domain. The composite state space is the cartesian product of the states of each actor and the composite action space is the union of the actions of each actor in the composite domain. The composite domain is denoted by Σ.

We define a particular problem instance $\mathtt{P} = (\Sigma, s_0, g, \{o_i\})$ by determining the initial state s_0, composite goal g, and objectives $\{o_i\}$ of each actor in the composite domain. A composite goal can be single or multiple objectives for an actor, common or conflicting objectives between actors, as well as global constraints specified for the entire scene. The composite goal, g, is the logical combination of the goals for all actors in the composite domain. We combine common goals using the $\wedge$ operator, indicating that all actors must satisfy these goals. We combine contradicting goals using the $\vee$ operator, indicating that any actor must satisfy its goal. The problem definition $\mathtt{P} = (\Sigma, s_0, g, \{o_i\})$ becomes the input for the planner.

5.2.1 Multiactor Action-Time Planner

During planning, the heuristic search generates a trajectory of actions for all actors in the composite space that satisfies g while optimizing $\{o_i\}$. This facilitates the generation of complicated interactions between actors, without needing centralized planning across all actors in the scenario. Even though an actors actions affect only the state space of its composite domain, the planner determines an actions possibility by considering the global state space of all actors in the scenario. This ensures collision-free trajectories between two independent plans. So, we can overlay the action trajectories for actors in different groups to generate a complete simulation.

Our planner builds on traditional planning approaches in three ways. First, it works in the composite space of multiple actors with competitive or collaborative goals. Second, it explicitly takes into account that different actions take various amounts of time and that actors actions overlap. Finally, it uses an automatically derived heuristic estimate to speed up the search.

Overview. For the current state, our heuristic planner generates a set of possible transitions. Each transition represents the forward simulation of the actions by one

time step in the composite space in which actors are simultaneously executing actions. The planner chooses a transition by minimizing the sum of the transitions total cost and the heuristic estimate of reaching the composite goal. It computes a transitions cost such that an action optimizes its own objective function. When the planner reaches a state that satisfies g, it returns the generated plan.

Transitions. A transition represents the simultaneous execution of actions chosen by all actors in the composite domain by one time step. A transition is valid and ready to simulate if all actors have a valid action theyre executing or ready to execute. An action is possible if three conditions are met: (1) the actor is currently not executing an action, (2) the preconditions of the action are satisfied and, (3) no constraints prohibit the action.

For a valid transition, all actions are simulated for one time step in a random order. The explicit modeling of time in the action definition results in overlapping actions, partially executed actions (action failure), and actors performing new actions while other actors are still performing their current action. After a transition, an actor might be in one of three states: (1) *Success.* The actor successfully completed the action. (2) *Executing.* The actor partially executed the action at that time step. (3) *Failure.* Other actors actions negated the actions preconditions. For both the success and failure states, the actor must choose a new action in the next time step.

Cost and Heuristic Function. The cost of simulating a transition $\{a_i\}$, at state s, in the time interval $(t,t+1)$ where a_i is the action chosen by actor i in the composite domain is $\sum_i o_i(\{c_j\})$. The heuristic function is used to provide a cost estimate from the current state to the goal state. Our design of a heuristic function is straightforward and efficient. We first relax the preconditions on the actions (all actions are deemed possible at any given instant of time) and do a fast greedy search for a trajectory of actions that takes the planner from the current state to the goal. The sum of the cost of all actions is the heuristic, h for that particular state, s.

5.2.2 The Animation and Simulation Engine

Once weve generated trajectories for the actors, we use a simple steering algorithm to simulate coin-shaped agents to accurately follow paths. Then, the animation system animates models of virtual humans and vehicles along the simulated paths. It animates characters by transitioning between walk, run, and stop animations on the basis of the movement speed. It also employs animations to visualize actors current actions, such as a thief stealing a hot dog.

5.2.3 Behavior Generation Algorithm

The algorithm used to generate multi-actor behaviors is described below:

1. Define Actors, $\texttt{Ac}_i = \langle S_i, A_i, C_i, B_i \rangle$, where S_i is the state space, A_i is the action space, C_i is the set of constraints and modifiers, and B_i is the set of behaviors defined for actor i.
2. Determine Composite Domains, $\texttt{CD}_j = \langle S_j^c, A_j^c, C_j^c, B_j^c \rangle$, where $S_j^c = \{S_1 \times S_2 \times ... S_n\}$ is the composite state space, $A_j^c = \bigcup_{i=1}^{n} \{A_i\}$ is the composite action space, $C_i^c = \{C_i\}$ is the set of specializations, and $B_j^c = \{B_i\}$ is the set of behaviors defined for all actors $i = 1$ to n in the composite domain $\texttt{CD}_j$.
3. For each Composite Domain, $\texttt{CD}_j$
   a. Define Search Domain, $\Sigma = (S^c, A^c, C^c)$.
   b. Determine initial state in the composite space of all agents, $s^0 = \bigcup_{i=1}^{n} s_i^0$
   c. Determine active behaviors, b_i for each actor, i in composite domain, $\texttt{CD}_j$. The active behavior for each actor determines the goal, g_i and the objective function o_i.
   d. The composite goal, g is the logical combination of the goals, $\{g_i\}$ for all actors in the composite domain. Common goals are combined using an $\wedge$ operator, indicating that all actors must satisfy their goal. Contradicting goals are combined using an $\vee$ operator, indicating that any one of the actors must satisfy their goal.
   e. If no behavior is active for actors, **Return**.
   f. Solve for sequence of actions π by performing a search, $\pi = \textbf{Search}(\Sigma, s^0, g, \{o_i\})$, where Σ is the search domain, s^0 is the composite start state, g is the composite goal, and $\{o_i\}$ are the objective functions for each actor.
4. Combine plans for all domains, $\Pi = \pi_1 \cup \pi_2 \cup ... \pi_n$.
5. Execute Global Plan, Π.
6. Determine new states of all actors.
7. Repeat Steps 2-6.

5.3 City Simulation

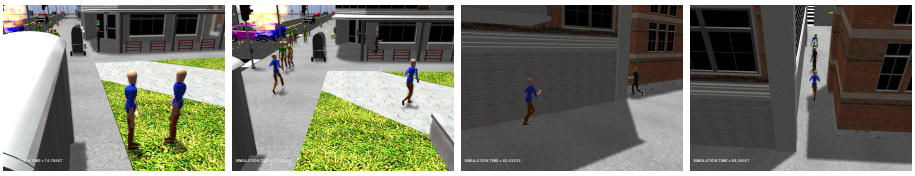
We demonstrate the effectiveness of our framework by authoring a car accident in a busy city street and observing the repercussions of the event on other actors that are part of the simulation, such as the old man and his son, whose behaviors are automatically generated using our framework.

Actor Specification. We first define the state space and action space of three actors in the scenario: (1) a generic pedestrian, (2) a vehicle and, (3) a traffic signal.

1. **Pedestrian:** The state of a pedestrian comprises its position, orientation, speed of movement, mass, and, a collision radius. In addition, pedestrians have the following abstract metrics: hunger, safety, amount of money. These metrics are variables whose values are modified by actions. The *Move* action (Figure 12(a)) kinematically translates an actor and has an associated distance and energy cost.

(a) (b) (c) (d)

Fig. 10 Snapshots of a city simulation authored using our framework: (a) Actors queue up at a hot dog stand while the vendors talk to one another. In the meantime, a thief lies in the shadows waiting for an opportunity to steal the money from the stand. (b) Cars giving right of way to pedestrians. (c) Cautious actors run to a place of safety in the event of an accident. (d) Fire-fighters extinguish the fire while daring actors look on.



(a) (b) (c) (d)

Fig. 11 Interaction between thief and the vendors: (a) The thief steals money from the hot dog stand when the vendors walk away (because of the accident). (b)-(d) The vendors collaboratively work together to surround the thief in the alley and manage to catch him.

The routine `CheckCollisions(..)` returns false if *Move* causes the pedestrian to enter a state of collision. Additional actions (e.g. `Eat`) can be associated with different metrics (e.g. hunger). A pedestrian is given a simple behavior to move towards a specified goal position ((Figure 12(b)) while minimizing distance and energy cost. The goal positions are randomly generated to produce a realistic city simulation with wandering pedestrians. Additionally, the pedestrians monitor the state of a traffic signal which coordinates the movement of pedestrians and vehicles at an intersection ((Figure 12(c)).

2. **Vehicles:** The state and action space of vehicles is defined similarly to simulate their movement. In addition, they have a metric `damage` which increases if a vehicle collides with another vehicle. Vehicles are constrained to stay on the roads, give right of way to pedestrians, and obey the traffic lights.
3. **Traffic Signals:** A traffic signal represents an environment actor that models the simulation of the traffic signals at the intersection. It has a single metric `signal state` which is the current state that the traffic signals at the intersection are in. An action, `ChangeTrafficSignal` (Table 3(a)) determines the state of the traffic signal based upon the current simulation time. The pedestrian and the vehicles query the signal state in order to follow the traffic signals.

Actor Specialization We can easily and intuitively specialize actors using our framework. Fire-fighter actors are specialized pedestrians with lower weights to the safety cost metric and with a common goal to extinguish fires. A grandfather pedes-

```
Action Move(Velocity : v, TStep: dt) {
  Precondition:
   CheckCollisions(self.position + v · dt) == false;
  Effect:
   self.position = self.position + v · dt;
  Cost Effect:
   self.energyCost = ½(self.mass)|v|²;
   self.distanceCost = |v|dt;
}
```

(a)

```
Behavior GoalBehavior {
  Precondition: self.goalPosition ≠ 0;
  Goal: self.goalPosition;
  Objective Function:
   min(self.distanceCost + self.energyCost);
}
```

(b)

```
Constraint PedSignalConstraint {
  Precondition: true;
  Constraint:
   if ((signal.signalState == 0
   ∧CrossingRoad(self.position,C))
   ∨(signal.signalState == 1
   ∧CrossingRoad(self.position,A))
   ∨(trafficSignal.signalState == 2
   ∧CrossingRoad(self.position,B)))
    true;
   else false;
}
```

(c)

Fig. 12 A generic pedestrian with a simple *Move* action (a), a behavior to go to a specified goal position (b), and a constraint to follow the traffic signals (c).

trian is specialized by reducing the walking speed and specializing them to follow their grandson. The objective of the grandson is to escort his grandson at all times and to keep him away from danger (e.g. car accidents, oncoming traffic and other pedestrians) which is achieved by incorporating the safety cost of the grandfather in the grandsons objective function.

Cautious and daring actors are authored by affecting the cost of actions that place them in danger (Table 3(b)). A street vendor is given the behavior of manning his hot dog stand and ensuring that his money is not stolen. A thief is authored with a goal to steal money using a *Steal* action (Table 3 (i),(j)) while minimizing the risk of getting caught (Table 3(j)). A cost modifier assigns a high cost to stealing in the presence of other actors. A reckless vehicle is modeled by introducing a high cost to moving at slower speeds and relaxing the constraints of obeying traffic signals and collisions with other vehicles (Table 3 (f),(g)).

5.4 Results

We populate a city block with pedestrians and vehicles using our framework. Actor specializations provide an easy and intuitive way to add variety and purpose to the virtual world. We observe pedestrians walking along the sidewalks in the city in a goal-oriented manner (satisfying hunger by getting a hot dog, going to the park to meet a friend, stopping to take a look at objects of interest) while obeying constraints and modifications (obey traffic lights, avoid collisions, stay off the streets etc). A video demonstrating the results can be found at `http://cg.cis.upenn.edu/hms/research/MultiActorPlanning/`.

The accident scenario. To add drama to the simulation, we introduce constraints on the trajectory of the entire simulation. First, we introduce a constraint, *AccidentC* (Table 3(e)), that an accident must happen (i.e. two vehicles must collide). A simulation is generated where two reckless vehicles collide with one another, resulting in a fire that stops the traffic at the intersection (Figure 10(d)). Cautious pedestrians who are near the accident run away to a safe distance in panic or walk away calmly (depending on their specialization) while daring actors approach the scene of the accident. The car accident triggers the activation of the behaviors in the fire-fighters who run to the location of the fires. They work together collaboratively to extinguish both fires (a result of the planner working in the composite domain). Upon noticing the accident, the vendor runs to a place of safety (high cost modifier on safety). As soon as the thief notices that the vendor has left his stand, he slowly approaches the stand, steals the money and runs to a place of safety.

Varying the simulation. We vary the simulation result by introducing other specializations or modifying existing ones. In a first take, we define the objectives of the two vendors to minimize safety cost as well as the cost of being robbed as individuals. When the accident happens, they run to a place of safety while keeping the stand in eyesight. As soon as they see the thief stealing the money, they both chase after him. However, the thief has a head-start and runs away. This is because the planner generates solutions that tries to achieve the objective of each vendor independently. Hence, we observe that in the composite domain of the thief and two vendors, the thief succeeds. In a second take, we modify the objectives of the vendors to minimize the cost of both being robbed (Table 3(h)). The common goal of the vendors implies that the planner searches for a solution that optimizes their combined objectives. As a result, the two vendors cooperate to corner the thief in an alley (Figures 11(a)-(d)).

Performance and Implementation Details. We demonstrate 106 actors in the city simulation, with 15 cars and 91 pedestrians. Based on constraints, goal definitions and spatial locality, the following composite domains are defined: (1) 15 cars and 4 fire-fighters, (2) old man and son and, (3) generic pedestrians grouped together based on spatial locality. Dividing the problem domain into smaller composite domains reduces the branching factor of the search by two orders of magnitude, re-

ducing the search problem to smaller, more feasible searches. The plans for each of these domains is then overlayed to form the complete solution. The performance results are provided in Figure 13. The amortized performance of our behavior generation framework for the results shown in the video is 0.02 seconds per actor per second of simulation generated.

Number of actors	106
Number of composite domains	12
Max # of actors in a composite domain	19
Total generation time	219 sec
Max generation time for one domain	76 sec
Min generation time for one domain	8 sec
Generation time per actor	2.06 sec
Length of output simulation	95 sec
Amortized time per actor per second	0.02 sec

Fig. 13 Performance Results.

6 Discussion

A key challenge in multi-actor simulations is to provide an appropriate interface for authoring high-fidelity virtual actors with feature-rich control mechanisms capable of complex interactions, while satisfying global scenario constraints. This chapter presents work that addresses the problem of behavior authoring at three levels.

Modeling Agent Personality. The OCEAN personality model defines 5 orthogonal axes to intuitively describe the personality of an agent. We describe a mapping of these personality traits to low-level simulation parameters, facilitating the control of agent and group personality in order to observe the emergence of different crowd behaviors. We validate our mapping by conducting a user study which assesses the perception of personality traits in a variety of crowd simulations demonstrating these behaviors. The personality traits provide an intuitive and flexible interface for authors to control social crowd dynamics.

Event Centric Authoring of Multi-Actor Interactions. We describe an event-centric behavior authoring paradigm where a user-authored centralized controller, defined using parameterized behavior trees (PBTs) [4] is used to coordinate behaviors between multiple interacting actors. Actors participating in an event temporarily suspend autonomy and are controlled by the event structure which treats them as limbs of a central entity for the duration of the event. PBTs provide a flexible, graphical programming language for authoring behaviors, and event-centric authoring alleviates the burden or coordinating the interactions between virtual actors.

However, there is greater burden on the author to define events, and to stitch together event sequences to create more complex narratives.

Automated Planning for Simulating Multi-Actor Interactions. The use of domain independent planners automates the behavior generation process where users only need to specify goals and objectives for actors. In order to inject heterogeneity into the simulation, generic actor definitions can be specialized by modifying the cost and effect of actions. Furthermore, the use of composite domains where actors with common or conflicting objectives are part of a single planning domain, facilitates the generation of complex interactions between actors. However, this method has the following limitations:

- The behaviors generated by our framework are heavily dependent on the manner in which actors are grouped together and the weights of the objectives.
- One of the major design choices of this framework was the use of a multi-actor planner which prohibits its use in interactive applications such as games.
- It is non-intuitive to define interactions between multiple actors which is accomplished indirectly by specifying common objectives and global constraints.

Towards A Unified Framework for Authoring Diversity in Personality and Behavior Multi-Actor Simulations. The traits of an actor that define personality, mood, and emotion provide an intuitive and high-level interface for specializing individual actors as well as groups of actors. Incorporating these traits into the behavior authoring process, especially at the event-centric level, facilitates the emergence of different actor interactions as a parametrization of these traits. For example, a behavior to describe a *Protest* event would differ based on the personality and mood of actors participating in the protest. A hostile and extroverted actor is more likely be an active participant while an anxious and nervous actor is more likely to steer away from the scene of the protest. Figure 14 illustrates the use of personality and mood traits in an authored event.

Authoring complex multi-actor interactions becomes prohibitive when designing the behavior of each agent in isolation. In order to cooperate or compete, actors must constantly communicate information and account for the current state of neighboring actors while performing an action. Planning based approaches provide flexibility of automation at higher performance, while event-centric authoring mitigates the need for treating every actor as an individual at greater burden to the author. A promising direction of exploration is to develop a planning based framework that automatically triggers sequences of user-authored events that satisfy global narrative constraints, while conforming to the roles of individual actors in the scenario. A planner can also be used at the discretion of the centralized controller to act alongside the event level as another tool for coordinating small narrative-driven actor interactions.

Events provide the facility to pre-author and tune actor interactions, while planners allow the generation of more dynamic activities in an automatic fashion. Using events, we can build a library of pre-authored collaborative action sequences of narrative and interactive significance. After preparing these actor interactions, we can

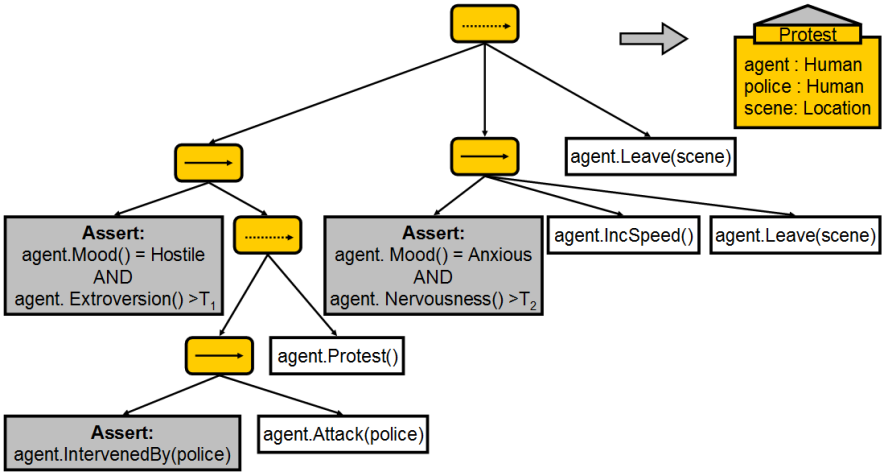


Fig. 14 A *Protest* event that takes into account the personality and mood of actors participating in the event.

apply a multi-actor planner to influence more local behaviors among groups of actors related by activity or domain in addition to selecting and applying the events we have designed. The planner may even be invoked at the discretion of the centralized controller, allowing various degrees of autonomy and control as needed by computational load, actor characteristics, or interactive user actions.

Acknowledgements We would like to thank the following people for their significant contributions in the research projects reported here. Francisco Garcia, Matthew Jones, Robert Mead, and Daniel Garcia helped define a parameterization of behavior trees for use in event-centric authoring. Nuria Pelechano, Jan Allbeck, and Ugur Gudukbay were involved in defining personality parameters for crowd simulations. Shawn Singh, Petros Faloutsos, and Glenn Reinman were instrumental in proposing the use of domain-independent planning for behavior authoring.

Parts of this research were supported by U.S. Army MURI "SUBTLE" and U.S. Army Robotics Collaborative Technology Alliance. The opinions expressed are solely those of the authors and not the sponsors.

References

1. F. Durupinar, N. Pelechano, J. M. Allbeck, U. Güdükbay, and N. I. Badler, "How the ocean personality model affects the perception of crowds," *IEEE Computer Graphics and Applications*, vol. 31, no. 3, pp. 22–31, 2011.
2. A. Shoulson and N. I. Badler, "Event-Centric Control for Background Agents," in *Proceedings of the 4th International Conference on Interactive Digital Storytelling (ICIDS '11)*. Springer, 2011, pp. 193–198.
3. A. Shoulson, D. Garcia, and N. I. Badler, "Selecting Agents for Narrative Roles," in *Proceedings of the 4th Workshop on Intelligent Narrative Technologies (INT4)*. AAAI Press, 2011.
4. A. Shoulson, F. Garcia, M. Jones, R. Mead, and N. I. Badler, "Parameterizing Behavior Trees," in *Proceedings of the 4th International Conference on Motion in Games (MIG '11)*. Springer, 2011, pp. 144–155.
5. M. Kapadia, S. Singh, G. Reinman, and P. Faloutsos, "A behavior-authoring framework for multiactor simulations," *Computer Graphics and Applications, IEEE*, vol. 31, no. 6, pp. 45 –55, nov.-dec. 2011.
6. N. Badler, *Virtual Crowds: Methods, Simulation, and Control (Synthesis Lectures on Computer Graphics and Animation)*. Morgan and Claypool, 2008.
7. L. F. Henderson, "The statistics of crowd fluids." *Nature*, vol. 229, no. 5284, pp. 381–383, February 1971. [Online]. Available: http://dx.doi.org/10.1038/229381a0
8. G. Lovas, "Modeling and simulation of pedestrian traffic flow," in *Transportation Research Record*, 1994, pp. 429–443.
9. J. Milazzo, N. Rouphail, J. Hummer, and D. Allen, "The effect of pedestrians on the capacity of signalized intersections," in *Transportation Research Record*, 1998, pp. 37–46.
10. S. P. Hoogendoorn and S. P. Hoogendoorn, "Pedestrian travel behavior modeling," in *In 10th International Conference on Travel Behavior Research, Lucerne*, 2003, pp. 507–535.
11. C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. ACM, 1987, pp. 25–34.
12. C. Reynolds, "Steering behaviors for autonomous characters," 1999. [Online]. Available: citeseer.ist.psu.edu/reynolds99steering.html
13. D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *PHYSICAL REVIEW E*, vol. 51, p. 4282, 1995. [Online]. Available: doi:10.1103/PhysRevE.51.4282
14. D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic." *Nature*, vol. 407, no. 6803, pp. 487–90, 2000. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/11028994
15. A. Braun, S. R. Musse, L. P. L. d. Oliveira, and B. E. J. Bodmann, "Modeling individual behaviors in crowd simulation," in *CASA '03: Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)*. Washington, DC, USA: IEEE Computer Society, 2003, p. 143.
16. N. Pelechano, J. M. Allbeck, and N. I. Badler, "Controlling individual agents in high-density crowd simulation," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ser. SCA '07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 99–108. [Online]. Available: http://dl.acm.org/citation.cfm?id=1272690.1272705
17. J. Dijkstra, H. J. P. Timmermans, and A. J. Jessurun, "A multi-agent cellular automata system for visualising simulated pedestrian activity," in *in S. Bandini and T. Worsch (Eds.), Theoretical and Practical Issues on Cellular Automata - Proceedings on the 4th International Conference on Cellular Automata for research and Industry*. Springer Verlag, 2000, pp. 29–36.
18. S. Chenney, "Flow tiles," *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 04)*, pp. 233–242, 2004.
19. K. Nishinari, A. Kirchner, A. Namazi, and A. Schadschneider, "Extended floor field ca model for evacuation dynamics," *IEICE Trans Inf and Syst*, vol. E84-D, no. 1, p. 7, 2003. [Online]. Available: http://arxiv.org/abs/cond-mat/0306262

20. F. Lamarche and S. Donikian, "Crowd of virtual humans: a new approach for real time navigation in complex and structured environments." in *Computer Graphics Forum 23.*, 2004.
21. C. Loscos, D. Marchal, and A. Meyer, "Intuitive crowd behaviour in dense urban environments using local laws," in *TPCG '03: Proceedings of the Theory and Practice of Computer Graphics 2003*. Washington, DC, USA: IEEE Computer Society, 2003, p. 122.
22. J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin, "Interactive navigation of multiple agents in crowded environments," in *SI3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games*. ACM, 2008, pp. 139–147.
23. K. H. Lee, M. G. Choi, Q. Hong, and J. Lee, "Group behavior from video: a data-driven approach to crowd simulation," in *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 109–118.
24. A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Computer Graphics Forum*, vol. 26, no. 3, pp. 655–664, September 2007.
25. S. Paris, J. Pettré, and S. Donikian, "Pedestrian reactive navigation for crowd simulation: a predictive approach," in *EUROGRAPHICS 2007*, vol. 26, 2007, pp. 665–674.
26. J. van den Berg, M. C. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*. IEEE, 2008, pp. 1928–1935.
27. S. Singh, M. Kapadia, W. Hewlett, and P. Faloutsos, "A modular framework for adaptive agent-based steering," in *Proceedings of the 2011 symposium on Interactive 3D graphics and games*, ser. I3D '11. ACM, 2011.
28. M. Kapadia, S. Singh, W. Hewlett, and P. Faloutsos, "Egocentric affordance fields in pedestrian steering," in *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games*. ACM, 2009, pp. 215–223.
29. M. Kapadia, S. Singh, W. Hewlett, G. Reinman, and P. Faloutsos, "Parallelized egocentric fields for autonomous navigation," *The Visual Computer, International Journal of Computer Graphics*, 2012.
30. J. Ondřej, J. Pettré, A.-H. Olivier, and S. Donikian, "A synthetic-vision based steering approach for crowd simulation," in *ACM SIGGRAPH 2010 papers*, ser. SIGGRAPH '10. New York, NY, USA: ACM, 2010, pp. 123:1–123:9. [Online]. Available: http://doi.acm.org/10.1145/1833349.1778860
31. Q. Yu and D. Terzopoulos, "A decision network framework for the behavioral animation of virtual humans," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ser. SCA '07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 119–128. [Online]. Available: http://dl.acm.org/citation.cfm?id=1272690.1272707
32. B. M. Blumberg, "Old tricks, new dogs: ethology and interactive creatures," Ph.D. dissertation, 1997, supervisor-Maes, Pattie.
33. D. V. Pynadath and S. C. Marsella, "Psychsim: Modeling theory of mind with decision-theoretic agents," *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1181—1186, 2005.
34. Massive Software Inc., "Massive: Simulating life," 2010, www.massivesofware.com.
35. E. Menou, "Real-time character animation using multi-layered scripts and spacetime optimization," in *Proceedings of ICVS '01*. London, UK: Springer-Verlag, 2001, pp. 135–144.
36. K. Perlin and A. Goldberg, "Improv: a system for scripting interactive actors in virtual worlds," in *Proceedings of ACM SIGGRAPH*. New York, NY, USA: ACM, 1996, pp. 205–216.
37. H. Vilhjálmsson, N. Cantelmo, J. Cassell, N. E. Chafai, M. Kipp, S. Kopp, M. Mancini, S. Marsella, A. N. Marshall, C. Pelachaud, Z. Ruttkay, K. R. Thórisson, H. Welbergen, and R. J. Werf, "The behavior markup language: Recent developments and challenges," in *Proceedings of IVA '07*, 2007, pp. 99–111.
38. A. B. Loyall, "Believable agents: building interactive personalities," Ph.D. dissertation, Pittsburgh, PA, USA, 1997.

39. J. Funge, X. Tu, and D. Terzopoulos, *Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters*. ACM Press/Addison-Wesley Publishing Co., 1999, pp. 29–38.
40. W. Shao and D. Terzopoulos, "Autonomous pedestrians," *Graph. Models*, vol. 69, pp. 246–274, September 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1323742.1323926
41. J. Allbeck and N. Badler, "Toward representing agent behaviors modified by personality and emotion," 2002.
42. J. Wiggins, *The five-factor model of personality: theoretical perspectives*. Guilford Press, 1996. [Online]. Available: http://books.google.com/books?id=UzXvvAsESjEC
43. A. Ortony, G. Clore, and A. Collins, *The Cognitive Structure of Emotions.* Cambridge: Cambridge University Press, 1988.
44. P. Gebhard, "Alma: a layered model of affect," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, ser. AAMAS '05. New York, NY, USA: ACM, 2005, pp. 29–36. [Online]. Available: http://doi.acm.org/10.1145/1082473.1082478
45. A. Egges, S. Kshirsagar, and N. Magnenat-Thalmann, "Generic personality and emotion simulation for conversational agents: Research articles," *Comput. Animat. Virtual Worlds*, vol. 15, pp. 1–13, March 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1071195.1071196
46. S. J. Guy, S. Kim, M. C. Lin, and D. Manocha, "Simulating heterogeneous crowd behaviors using personality trait theory," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '11. New York, NY, USA: ACM, 2011, pp. 43–52. [Online]. Available: http://doi.acm.org/10.1145/2019406.2019413
47. B. Magerko, J. E. Laird, M. Assanie, A. Kerfoot, and D. Stokes, "Ai characters and directors for interactive computer games," *Artificial Intelligence*, vol. 1001, pp. 877–883, 2004.
48. M. Mateas and A. Stern, *Integrating Plot , Character and Natural Language Processing in the Interactive Drama Faade*, 2003, vol. 2. [Online]. Available: http://www.cs.ucsc.edu/~michaelm/tenurereview/publications/mateas-tidse2003.pdf
49. M. Si, S. C. Marsella, and D. V. Pynadath, *THESPIAN: An Architecture for Interactive Pedagogical Drama*, 2005, pp. 595–602.
50. M. O. Riedl, C. J. Saretto, and R. M. Young, *Managing interaction between users and agents in a multi-agent storytelling environment*. ACM Press, 2003, vol. 34, pp. 186–193. [Online]. Available: http://portal.acm.org/citation.cfm?doid=860575.860694
51. B. Li and M. Riedl, *Creating Customized Game Experiences by Leveraging Human Creative Effort: A Planning Approach*. Springer, 2011, p. 99116. [Online]. Available: http://www.springerlink.com/index/428G8512624879U6.pdf
52. C. Stocker, L. Sun, P. Huang, W. Qin, J. M. Allbeck, and N. I. Badler, "Smart events and primed agents," *Proceedings of the 10th International Conference on Intelligent Virtual Agents (IVA 10)*, vol. 6356, p. 1527, 2010.
53. L. R. Goldberg, "An alternative "description of personality": The big-five factor structure," *Journal of Personality and Social Psychology*, vol. 59, pp. 1216–1229, 1990.
54. E. T. Hall, *The Hidden Dimension*. Anchor Books, 1966.
55. D. Isla, "Handling complexity in the Halo 2 ai," *Game Developers Conference*, 2005. [Online]. Available: http://www.gamasutra.com/gdc2005/features/20050311/isla_01.shtml
56. B. Knalfa, "Introduction to behavior trees," http://www.altdevblogaday.com/2011/02/24/introduction-to-behavior-trees/, 2011.
57. M. Kapadia, S. Singh, G. Reinman, and P. Faloutsos, "Multi-actor Planning for Directable Simulations," in *Proceedings of the 2011 Workshop on Digital Media and Digital Content Management*. IEEE Computer Society, 2011, pp. 111–116.

```
Action ChangeTrafficSignal {
 Precondition:
  true;
 Effect:
  timeMod = currentTime % 100;
  if (timeMode <= 35)
    self.signalState = 0;
  else if (timeMode  <= 70)
    self.signalState = 1;
  else self.signalState = 2;
}
```

(a)

```
CostModifier DaringCM {
 Precondition:
  ∃ a: a.danger > 0 ;
 Cost Effect:
  self.safetyCost =
     MAX_COST - max(a.danger);
}
```

(b)

```
EffectModifier DaringEM {
 Precondition:
  true;
 Effect:
  a = argmax(a.danger) ;
  self.goalPosition = a.position;
}
```

(c)

```
Behavior FireFighterB {
 Precondition:
  ∃ a ∈ Actors: a.fire > 0;
 Goal:
  ∀ a ∈ Actors a.fire = 0;
 Objective Function:
  min(0.3·self.safetyCost
      + self.distanceCost
      + self.energyCost);
}
```

(d)

```
Constraint AccidentC {
 Precondition:
  true;
 Constraint:
  // Two vehicles must collide
  // at some point in time
  ∃ a1,a2 :
  IsAVehicle(a1) ∧
  IsAVehicle(a2) ∧
  Distance(a1,a2) < 5.0;
}
```

(e)

```
EffectModifier RecklessVehicleEM {
 Precondition:
  true;
 Effect:
  self.collisionRadius = MIN;
  self.followSignals = FALSE;
}
```

(f)

```
CostModifier RecklessVehicleCM {
 Precondition:
  true;
 Effect:
  // low cost for traveling
  // at MAX_SPEED
  self.speedCost
    = MAX_SPEED-self.speed;
}
```

(g)

```
Behavior CooperativeVendorB {
 Precondition:
  true;
 Goal:
  self.money >= 100 ∧
  otherVendor.money >= 100;
 Objective Function:
  min(self.stolenCost
    + otherVendor.stolenCost);
}
```

(h)

```
Behavior ThiefB {
 Precondition:
  true;
 Goal:
  self.money >= 100
 Objective Function:
  min(self.distanceCost
     + self.energyCost);
}
```

(i)

```
Action Steal(Actor a, Amount: m){
 Precondition:
  m <= a.money ∧
  DistanceBetween(self,a) < 1.0;
 Effect:
  a.money = a.money - m
  self.money = self.money + m
Cost:
  self.stealCost = m;
  a.stolenCost = m;
}
```

(j)

Table 3 Scripts used to author the city simulation.