

Virtual Crowds: Methods, Simulation, and Control

Synthesis Lectures on Computer Graphics and Animation

Editor

Brian A. Barsky, *University of California, Berkeley*

Interactive Shape Design

Marie-Paule Cani, Takeo Igarashi and Geoff Wyvill
2008

Real-Time Massive Model Rendering

Sung-eui Yoon, Enrico Gobbetti, David Kasik, and Dinesh Manocha
2008

Virtual Crowds: Methods, Simulation and Control

Nuria Pelechano, Jan. Allbeck, and Norman I. Badler
2008

High Dynamic Range Video

Karol Myszkowski, Rafal Mantiuk, and Grzegorz Krawczyk
2008

GPU-Based Techniques For Global Illumination Effects

L'aszl'o Szirmay-Kalos, L'aszl'o Sz'esci and Mateu Sbert
2008

High Dynamic Range Imaging Reconstruction

Asla Sa, Paulo Carvalho, and Luiz Velho IMPA, Brazil
2007

High Fidelity Haptic Rendering

Miguel A. Otaduy, Ming C. Lin
2006

A Blossoming Development of Splines

Stephen Mann
2006

Copyright © 2008 by Morgan & Claypool

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopy, recording, or any other except for brief quotations in printed reviews, without the prior permission of the publisher.

Virtual Crowds: Methods, Simulation, and Control
Nuria Pelechano, Jan M. Allbeck, and Norman I. Badler
www.morganclaypool.com

ISBN: 9781598296419 paperback

ISBN: 9781598296426 ebook

DOI: 10.2200/S00123ED1V01Y200808CGR008

A Publication in the Morgan & Claypool Publishers series

SYNTHESIS LECTURES ON COMPUTER GRAPHICS AND ANIMATION #8

Lecture #8

Series Editor and Affiliation: Brian A. Barsky, University of California–Berkeley

Series ISSN

ISSN 1933-8996 print

ISSN 1933-9003 electronic

Virtual Crowds: Methods, Simulation, and Control

Nuria Pelechano

Universitat Politècnica de Catalunya

Jan M. Allbeck and Norman I. Badler

University of Pennsylvania

SYNTHESIS LECTURES ON COMPUTER GRAPHICS AND ANIMATION #8



MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

There are many applications of computer animation and simulation where it is necessary to model virtual crowds of autonomous agents. Some of these applications include site planning, education, entertainment, training, and human factors analysis for building evacuation. Other applications include simulations of scenarios where masses of people gather, flow, and disperse, such as transportation centers, sporting events, and concerts. Most crowd simulations include only basic locomotive behaviors possibly coupled with a few stochastic actions. Our goal in this survey is to establish a baseline of techniques and requirements for simulating large-scale virtual human populations. Sometimes, these populations might be mutually engaged in a common activity such as evacuation from a building or area; other times they may be going about their individual and personal agenda of work, play, leisure, travel, or spectator. Computational methods to model one set of requirements may not mesh well with good approaches to another. By including both crowd and individual goals and constraints into a comprehensive computational model, we expect to simulate the visual texture and contextual behaviors of groups of seemingly sentient beings.

KEYWORDS

Animated characters, autonomous agents, CAROSA, collision avoidance, computer animation, crowd simulation, evacuation studies, HiDAC, human behaviors, navigation planning, parameterized actions, pedestrians, presence, psychological factors, roles, social forces, virtual environments

Dedication

To our families

Acknowledgments

We thank the many individuals and organizations that have supported this work with their donations, intellect, talent, and funding. We are grateful to Autodesk, nVIDIA, and Cal3D for their continued development of software and hardware that enrich computer graphics in general and the Center for Human Modeling and Simulation (HMS) at the University of Pennsylvania in particular. We thank all of the members of HMS for their support of these efforts and individually recognize Jeff Wajcs, Grace Fong, Funda Durupinar, and Catherine Stocker for their contributions. Additionally, we appreciate the guidance provided by Joan Pelechano, Mel Slater, and Ali Malkawi. The perceptive comments of the book reviewers are greatly appreciated, which we hope are reflected in better content, coverage, and structure of this work. Thanks also to our publisher, Michael Morgan, who encouraged us to undertake this project and whose enthusiasm kept us close to schedule.

The authors' research described in this volume was partially supported by the National Science Foundation grant IIS-0200983, the Office of Naval Research Virtual Technologies and Environments grant N0001 4-04-1-0259, the U.S. Army Research Office grants N61339-05-C-0081 and MURI W911NF-07-1-0216, the T. C. Chan Center Building Simulation and Energy Studies, a Fulbright scholarship, a Spanish Government grant TIN2007-67982-C02-01, and the School of Engineering and Applied Science at the University of Pennsylvania. Any opinions expressed are entirely those of the authors and do not reflect any official positions of the sponsors.

Contents

1.	Introduction	1
1.1	Terminology	2
1.2	Overview	4
1.2.1	Lessons Learned From the Psychology Literature	11
1.2.2	Main Features in Crowd Simulation Systems	13
2.	Crowd Simulation Methodology Survey.....	15
2.1	Microscopic and Macroscopic Approaches Used to Model Pedestrian Movements.....	15
2.2	Microscopic Models	15
2.2.1	Social Force Models	15
2.2.2	Cellular Automata Models	18
2.2.3	Rule-Based Models	19
2.3	Macroscopic Models.....	22
2.3.1	Regression Models.....	22
2.3.2	Route Choice Models.....	22
2.3.3	Queuing Models.....	22
2.3.4	Gaskinetics	22
2.4	Current Pedestrian Software Systems.....	22
2.5	Summary of Crowd Models	32
2.5.1	Some Limitations of the Current Commercial Software for Crowd Evacuation	34
2.6	Navigation	37
2.6.1	Cell and Portal Graphs	38
2.6.2	Flow Tiles and Potential Field Methods	39
2.6.3	Probabilistic Roadmaps	39
2.7	Environment Modeling	40

3.	Individual Differences in Crowds	43
3.1	Personality and Emotion Models	43
3.2	Physiology	44
3.3	Sociology: Subgroups	44
3.4	Culture, Roles, and Status	45
3.5	Summary	46
4.	Framework (HiDAC + MACES + CAROSA)	47
4.1	Interaction Between Framework Levels and Psychological Models	49
4.2	Parameters Affecting Crowd Behavior	52
5.	HiDAC: Local Motion	57
5.1	Introduction.....	57
5.2	Agents' Speeds and Densities	58
5.2.1	Walking Speeds and Densities When Walking Downstairs	62
5.3	Perception.....	62
5.4	Crossing Portals.....	64
5.5	The HiDAC Model	66
5.5.1	Avoidance Forces	68
5.5.2	Repulsion Forces.....	72
5.5.3	Solution to "Shaking" Problem in High Densities	73
5.5.4	Organized Behavior: Queuing	74
5.5.5	Pushing Behavior.....	74
5.5.6	Falling and Becoming Obstacles	75
5.5.7	Panic Propagation.....	77
6.	MACES: Wayfinding With Communication and Roles.....	81
6.1	Introduction.....	81
6.2	Navigation Algorithm	82
6.2.1	Exploring the Building	83
6.2.2	Communication Affecting Evacuation Times	85
6.2.3	Relevance of Having Trained Leaders vs. Untrained Leaders	87
6.2.4	Importance of Leadership	88
6.2.5	Simulating Psychology Affecting Roles and Navigation	90
6.2.6	Interactive Navigation and Impatient Agents Avoiding Bottlenecks	93
7.	CAROSA: Functional Crowds	97
7.1	Applications with Actions	97
7.2	Parameterized Action Representation	99

7.2.1	Key Fields of the Action Representation.....	99
7.2.2	Key Fields of the Object Representation.....	100
7.2.3	Four Types of Actions.....	101
7.2.4	Application to Crowds	105
7.3	Carosa System Overview.....	106
7.3.1	PAR System	108
7.3.2	Actionary.....	108
7.3.3	Agent Process	108
7.3.4	Processing the Four Action Types	110
8.	Initializing a Scenario.....	113
8.1	Building Modeling	113
8.1.1	Cell and Portal Graph Automatic Generation	114
8.1.2	Generate Cell and Portal Graph for Each Floor	115
8.1.3	Identify Stairs and Link Floors Through New Cells.....	117
8.1.4	Identify and Store Walls	118
8.1.5	Identify and Store Obstacles	119
8.1.6	Precalculating Data for Real-Time Simulation	119
8.2	Layout of Environment	123
8.3	Character Profiles	123
8.4	Creating Groups.....	125
8.5	Constructing Actions	125
8.6	Refining The Simulation	127
8.6.1	Effects of Changes to the Environment	127
8.6.2	Modifying Roles.....	128
8.6.3	Scripting Characters	129
9.	Evaluating Crowds	131
9.1	Feature Comparison	131
9.1.1	Low-Level Features.....	131
9.1.2	Middle-Level Features	132
9.1.3	High-Level Features.....	132
9.1.4	Summary	133
9.2	Comparison to Real-World Data	133
9.2.1	Sensor Data	133
9.2.2	Action Statistics.....	133
9.2.3	Validation Through the Society of Fire Protection Engineers Guide..	134
9.3	User Evaluations.....	135

9.4	<i>Presence</i> in Virtual Worlds	136
9.4.1	Important Egocentric Features.....	137
9.4.2	Experimental Evidence From the Literature.....	139
9.4.3	Pilot Experiment	139
9.4.4	Initial Results and Future Work	142
9.4.5	Conclusions on <i>Presence</i> as a Validation Method.....	144
10.	Summary	147
Appendix A		151
Appendix B		153
Appendix C		159
References		163
Author Biographies		175

CHAPTER 1

Introduction

As we journey through our day, our lives intersect with other people. We see people leaving for work, waiting for trains, meeting with friends, working at their jobs, and engaging in numerous other activities. People create a rich tapestry of activity throughout the day, a *human texture*. We may not always be conscious of this texture, but we would definitely notice if it were missing, and it *is* missing in many computer graphics simulations of 3D environments populated by collections of animated virtual humans.

There are many applications of computer animation and simulation where it is necessary to model virtual crowds of autonomous agents. Some of these applications include site planning, education, entertainment, training, and human factors analysis for building evacuation. Other applications include simulations of scenarios where masses of people gather, flow, and disperse, such as transportation centers, sporting events, and concerts. Most crowd simulations include only basic locomotive behaviors possibly coupled with a few stochastic actions. Our goal in this survey is to establish a baseline of techniques and requirements for simulating large-scale virtual human populations. Sometimes these populations might be mutually engaged in a common activity, such as evacuation from a building or area; other times they may be going about their individual and personal agenda of work, play, leisure, travel, or spectator. Computational methods for modeling one set of requirements may not mesh well with good approaches to another. By including both crowd and individual goals and constraints into a comprehensive computational model, we expect to simulate the visual texture and contextual behaviors of groups of seemingly sentient beings.

The structure of this exposition includes surveys of existing computational crowd motion models, descriptive models originating in data analysis and urban planning, and functional models of human behavior. We stop short of exploring the details of individual human motion animation techniques, preferring instead to focus on the collective structure, motion, and control of groups and the differentiation and individualism of roles within groups. Within this matrix, however, one can readily embed various computational methods for animating more personal details of individuals adapted to the spatial context and task execution desired.

2 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

1.1 TERMINOLOGY

A wide variety of terms appear in the literature that refer to humans “inhabiting” virtual worlds. *Avatars* are characters that represent and are controlled directly by a real person. The term *avatar* is often confused with characters that take their motivations and behaviors from computer programs and simulators. These computer-driven characters or *virtual humans* may also be called *digital humans*, *autonomous agents*, or *humanoids*. We use these terms interchangeably. Although finer distinctions are plausible, they are not meaningful here: sometimes “autonomous” is equated to “undirected” behaviors, but we prefer to consider all action choices as under some sort of computational control. Different virtual human control mechanisms exist, and we will differentiate their qualities and characteristics shortly.

Existing terminology used to describe multiple beings can be even more confusing. *Artificial life*, “*boids*,” and *multiagent systems* simulate more than one (autonomous) character. Other terms used include *crowds*, *pedestrians*, *groups*, and *populations*. It is not always clear what is meant by these terms or the functionality of the individuals that these systems represent. Here we will briefly examine some of the terminology associated with crowds.

WordNet definitions ([Fellbaum 1998](#)):

- *crowd*: a large number of things or people considered together
- *pedestrian*: a person who travels by foot
- *populace*: people in general considered as a whole
- *population*: the people who inhabit a territory or state.

Crowd, *populace*, and *population* all inherit from the *group* hypernym. However, *crowd* is also a *gathering* and a *social group*, whereas *populace* and *population* are people. Hyponyms for *crowd* include:

- *army*: a large number of people united for some specific purpose
- *crush, jam, press*: a dense crowd of people
- *drove, horde, swarm*: a moving crowd
- *huddle*: a disorganized and densely packed crowd
- *mob, rabble, rout*: a disorderly crowd of people
- *lynch mob*: a mob that kills a person for some presumed offense without legal authority
- *phalanx*: any closely ranked crowd of people
- *troop, flock*: an orderly crowd.

We are not claiming that WordNet is necessarily the best source for clarification, but we found it to be a useful jumping off point. Looking at the psychosocial literature on crowds yields,

for example, Figure 1.1, showing a taxonomy of crowds as described by Brown (1954). Here Brown first makes a division into active (mobs) and passive (audiences) crowds. He then further breaks down these divisions according to their purpose or feeling. For example, an intentional, recreational audience may be watching a basketball game.

From a computational perspective, the question is: do all crowd simulators really simulate crowds? Systems that animate more than a few agents seem to be regarded as crowd simulators by the research community, but not all are. Terzopoulos and colleagues (Shao and Terzopoulos 2005; Yu and Terzopoulos 2007) have called their work simulations of “autonomous pedestrians”. As they simulate commuters in a train station, this seems an apt term. One of the simulations shown by Treuille et al. (2006), “continuum crowds,” depicts agents on city streets; these might also be deemed pedestrians. The seminal work by Reynolds (1987) on “flocks, herds, and schools”—generically called boids—seems to be very appropriately named. Much of the work in crowd simulations has been for evacuation (Helbing et al. 2000; Pelechano and Badler 2006). Figure 1.1 shows that these events would be categorized as escapes and may be further broken into events with and without panic. Here escapes are a type of mob which, in turn, is a type of crowd. For the remainder of this exposition, we will use the common word *crowd* to describe simulations of modest numbers of characters, though for some of the discussions, terms such as pedestrians or populace may be more appropriate.

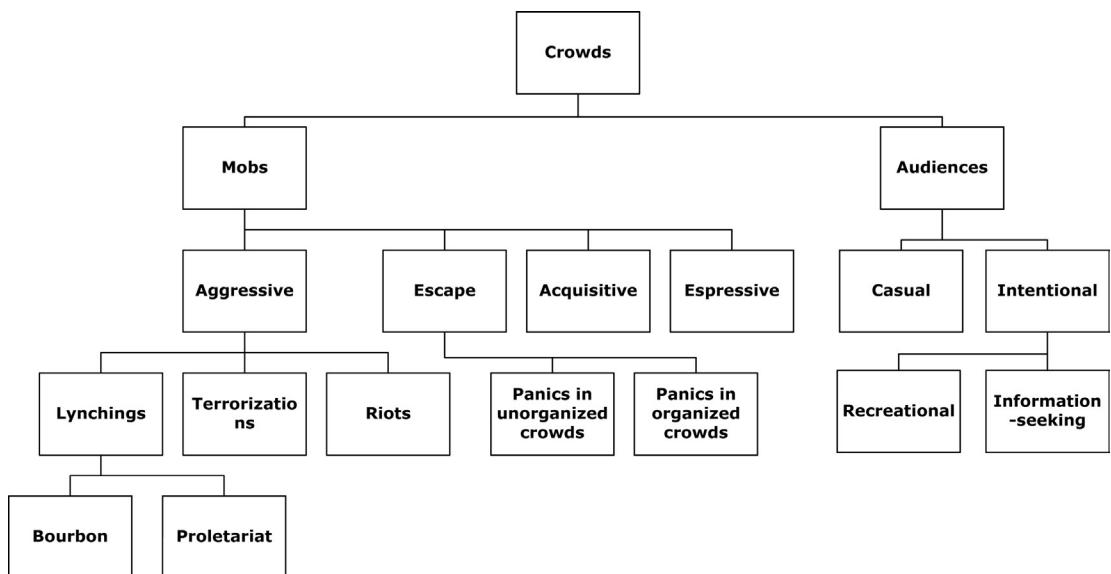


FIGURE 1.1: Mass phenomena from Brown (1954).



FIGURE 1.2: Examples of crowd behavior simulated with our system. On the left is a drill evacuation simulation and on the right is a cocktail party simulation, where virtual agents perform several actions to interact with other agents (Pelechano, Stocker et al. 2008).

To provide an overall structure for the presentation, we use the context of three of our own simulation systems that have been joined into a single framework. HiDAC was constructed for simulating high-density crowds, i.e., *crushes* or *presses*, but is parameterizable and can also simulate *huddles*, *mobs*, *escapes*, and *flocks*, for example. HiDAC combines a social forces model with a rule-based model to yield interesting emergent behaviors (Section 5.5). MACES builds on HiDAC to simulate additional behaviors, including evacuation, wayfinding with interagent communication and certain personal roles (Chapter 6). CAROSA builds on the foundations of HiDAC and MACES, but moves the focus away from *escapes* by incorporating a richer role and action representation that allows the depiction of functional heterogeneous crowds (Chapter 7). CAROSA might be used to simulate *audiences*, but would better be described as a system for simulating a populace or population of individuals such as the daily activities of inhabitants of an office building or neighborhood, or perhaps even an entire city (Figure 1.2).

1.2 OVERVIEW

Our intention here is to provide insight into crowd simulation software design choices and their resultant characteristics and features. We begin by providing a survey of crowd simulation research.

It can be difficult to compare virtual human technologies. Table 1.1 shows an attempt to create a few features and points along a scale for each of the features. *Appearance* fairly straightforwardly describes aspects of the visual qualities of characters. *Function* reflects the capabilities of

the virtual humans. What movements and behaviors are possible and how accurate are they? Are the behaviors a visualization of the state of the character in terms of injuries and psychology, for example? Ultimately, can the behaviors of the virtual human coordinate with other virtual humans to create teams? *Time* reflects efficiency in computation. Is the computation too heavy for even interactive manipulations? Can the motions be synthesized in real time? Can more than one character be simulated in real time, and can members of the crowd be coordinated at a viable frame rate? *Autonomy* indicates the level at which the character can control itself. Does the character creator have to specify every moment of the character's behavior, or can it make its own decisions? Can the autonomous character aid in the decision making of other virtual humans and, therefore, act as a leader? *Individuality* indicates to what level characters can be differentiated from one another in terms of the other features. Do all of the characters look and behave the same? Can observers recognize cultural distinctions and personality types? Ultimately, can specific individuals be recognized?

Table 1.1 was originally created with one or a few virtual humans in mind. Here we are more interested in larger numbers. In terms of *appearance*, we would certainly desire variability for realistic scenes. This is also true of *functionality*. We would not want to see every character in a scene walking identically, for example. Recent work by McDonnell et al. (2007, 2008) (Figure 1.3) has explored the amount of differentiation needed in these features for crowds. The *time* feature already

TABLE 1.1: Comparative virtual humans originally published by Allbeck and Badler (2002)

<i>Appearance</i>	2D drawings > 3D wireframe > 3D polyhedra > curved surfaces > freeform deformations > accurate surfaces > muscles, fat > biomechanics > clothing, equipment > physiological effects (perspiration, irritation, injury)
<i>Function</i>	cartoon > jointed skeleton > joint limits > strength limits > fatigue > hazards > injury > skills > effects of loads and stressors > psychological models > cognitive models > roles > teaming
<i>Time</i> (time to create movement at the next frame)	off-line animation > interactive manipulation > real-time motion playback > parameterized motion synthesis > multiple agents > crowds > coordinated teams
<i>Autonomy</i>	drawing > scripting > interacting > reacting > making decisions > communicating > intending > taking initiative > leading
<i>Individuality</i>	generic character > hand-crafted character > cultural distinctions > sex and age > personality > psychological-physiological profiles > specific individual

6 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

included how many characters can be simulated per frame. *Autonomy* may be extended to include mechanisms for characters to coordinate, collaborate, and even compete with other agents. The characters would themselves decide when a task requires additional personnel and schedule appropriately. *Individuality* makes crowd scenes more interesting and realistic. Also important is the formation of groups. People have affiliations with others that change over time and even during the course of a day. In scenarios with larger numbers of virtual humans, these dynamic changes become important for realism.

Crowd simulation researchers tend to focus on furthering development in one or perhaps a couple of these areas. McDonnell and colleagues have focused mainly on overall appearance (McDonnell et al. 2007, 2008), but this relates to time in that frame rate is affected by the resolution of the characters (McDonnell et al. 2005). Ahn et al. (2006) examined using not just the level of detail in character models but also the level of detail in motion to increase the number of characters that can be simulated in real time. Many research groups have looked to increase the level of function of crowds particularly in the area of navigation and collision avoidance (Reynolds 1999; Helbing et al. 2005; Treuille et al. 2006; Pelechano et al. 2007; Sud, Andersen, et al. 2007). Collaboration and coordination within crowd simulations has been explored, but much more work is needed (Shao and Terzopoulos 2005; Pelechano and Badler 2006; Yu and Terzopoulos 2007). Individuality has been the focus not only of many autonomous agents researchers, but also of a few crowd simulation research groups (Musse and Thalmann 2000; Pelechano et al. 2005; Pelechano and Badler 2006; Durupinar et al. 2008).

Animating virtual crowds is often mediated by local rules (Musse and Thalmann 2001), forces (Helbing et al. 2000), or flows (Chenney 2004). The goal is usually either to achieve real-time simulation for very large crowds, where each individual's behavior is not important as long as the overall crowd movement looks realistic, or to focus on individual behaviors using complex cognitive models (but achieving real time only for smaller crowds). Much effort has been put into improving



FIGURE 1.3: Perception of crowd variety (McDonnell et al. 2008) C 2008 ACM, Inc. Reprinted by permission.

the behavioral realism of each of these approaches; however, none of those models can realistically animate the complexity and function of population movements in a large building or a city. Current work in crowd motion has focused on realistically animating moderate and high-density crowds for real-time applications, where agents are endowed with psychological elements that will drive not only their high-level decision making, but also their reactive behavior (pushing, moving faster, being impatient, etc.) (Pelechano et al. 2007). The task of specifying and animating large collections of functional individuals is just beginning to be addressed.

On the global navigation level, most approaches either deal with simple environments or assume that agents have complete knowledge of the environment and move toward their goal as individuals (without interacting with other agents). Other work has focused on realistically simulating how communication affects the behavior of autonomous agents (Cassell et al. 1999) and on how combining different personalities and situations affects an agents' navigation and the way it interacts with other agents and the environment (Pelechano and Badler 2006).

To have autonomous agents navigating a virtual environment, it is necessary to endow them with a wayfinding algorithm to obtain a cognitive map of a building or environment. Wayfinding is the process of determining and following a route to some destination (Golledge 1999). It deals with the cognitive component of navigation and, therefore, with the knowledge and the information processing required to move (locomote) from an initial position to a goal position. Initially, the individuals of the crowd may only have partial information about the internal building structure, but as they explore it and communicate with other individuals in the crowd, they will be able to find paths toward their goals or exits.

Wayfinding is defined as a spatial problem-solving process with three subprocesses: decision making, decision execution, and information processing. To carry out wayfinding, each agent would need:

- a cognitive map: a mental model of the space
- an orientation: its current position within the cognitive map
- exploration: processes to learn the features of the space (doors, walls, hazards, etc).
- navigation: the process of making its way through the environment.

There have been several cognitive agent architectures proposed to generate human-like behavior. Cognitive architecture features include a broad set of possible requirements, such as perception, memory, attention, planning, reasoning, problem solving, learning, emotions, and mood. Agent architectures motivated from a cognitive perspective generally consist of a knowledge representation, algorithms that learn, and modules that plan actions based on that knowledge (Wray et al. 1999; Yu and Terzopoulos 2007). Tu and colleagues have worked on behavioral animation for creating artificial life, where virtual agents are endowed with synthetic vision and perception of the environment (Tu

8 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

and Terzopoulos 1994; Funge et al. 1999). However, the complexity of these systems often makes them difficult to scale up and port to new scenarios.

Rule-based schemes are fast enough for use with dozens of agents. Reynolds (1987) described the first use of a distributed behavioral model to produce a flocking behavior. Brogan and Hodges (1997, 2002) used particle systems and dynamics for modeling the motion of groups with significant physics. Helbing et al. (2000, 2002) described a microscopic (personal) approach to simulate pedestrian movement based on a social force model. This approach uses social analogs of physical forces and solves Newton's equations of motion for each individual; forces considered include repulsive interactions, friction forces, dissipation, and fluctuations. Helbing's model treats individuals as particles, and those forces appear from the interaction between particles when their density is great enough that particles bump into each other. More recently, Lerner et al. (2007) introduced a novel approach for rule-based simulation based on examples, in which tracking data from real crowds is used to create a database of examples that is subsequently used to drive the simulated agents' behavior (Figure 1.4). Other data-driven approaches use learning algorithms based on locally weighted linear regression to simulate crowd behavior (Lee et al. 2007).

These traditional crowd simulators ignore the differences between individuals and treat everyone as having the same simple behavior, but there are other models that represent each individual as being controlled by rules based on physical laws or behavioral models showing individualism (Braun et al. 2003). In a multiagent crowd system, the agents are autonomous, typically hetero-

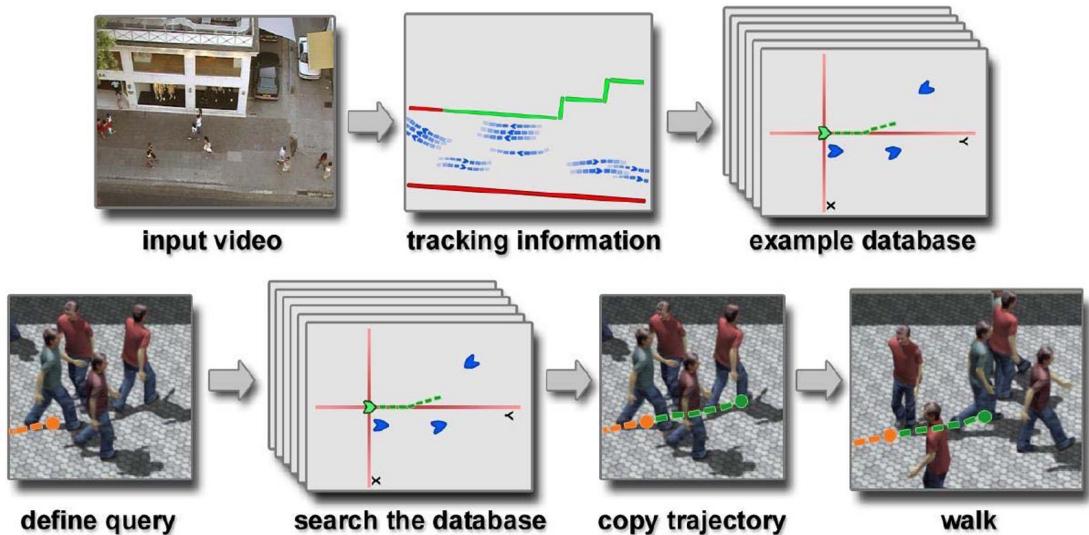


FIGURE 1.4: Crowds by example (Lerner et al. 2007).

geneous, and their concern is with coordinating intelligent behaviors among themselves, that is, how these agents can coordinate their knowledge, goals, skills, and plans to take action and to solve problems. Some of these applications include crowd behavioral models used in the training of military personnel (Weaver et al. 2001) and crowd motion simulations to support architectural design for both everyday use (Bouvier and Guilloteau 1996) as well as emergency evacuation conditions (Musse and Thalmann 2000, 2001; Still 2000).

Hierarchical schemes have been proposed to address scalability (Farenc et al. 2000). In particular, Musse and Thalmann (2001) provide crowds with different levels of autonomy for hierarchical crowd behaviors, but complex individual behaviors have not been shown.

Complex behaviors can be facilitated by using an action representation that holds the semantics of the actions and the objects that participate in the actions (Bindiganavale et al. 2000). Representing this information in a general way such that it is applicable to many scenarios enables the writing of agent controllers that are more general and require less custom work for new domains. By adding additional information about the agents being simulated, for example, their roles, and linking this information to appropriate actions, *functional* crowds (crowds that perform actions and interact with the environment in meaningful ways) can be generated. Additionally, different types of actions can be modeled to enrich a simulation with emergent behaviors. Actions can be categorized in different ways based on their origin. Some actions are planned or scheduled to achieve certain goals (deliberative actions). Others are reactions to events in the environment (reactive actions). Some are based on needs such as hunger or energy and may be planned for or woven into existing plans (opportunistic actions). Finally, some actions seem somewhat random, but often, they have underlying probabilities (aleatoric or stochastic actions). When coupled with agent descriptions, these actions are capable of providing agents with meaningful behavior options closer to the real texture of human activities.

In multiagent systems, each agent needs to sense the environment to perceive changes and react to them. Perception is often simulated by casting a set of rays and finding their intersection with obstacles around the object (Pan et al. 2005; Shao and Terzopoulos 2005). Massive SW (Massive Software Inc. 2005) (used, for example, in the *Lord of the Rings* movies) has also developed a crowd simulation system with a vision-based behavior. Individual agents do a low-resolution render of the scene from their own point of view and use computations based on that image to guide their actions.

To reduce the complexity of controlling all the agents in the crowd while detailed behaviors are still guaranteed, several systems have attached information to the environment (Farenc et al. 1999; Thomas and Donikian 2000; Tecchia et al. 2001; Pelechano and Badler 2006). Our approaches also embed information, such as shortest paths, in the environment. Individual agents will have differential types of access to that information, and they will use it in different ways, depending

10 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL



FIGURE 1.5: Treuille et al.'s (2006) continuum crowds C 2006 ACM, Inc. Reprinted by permission.

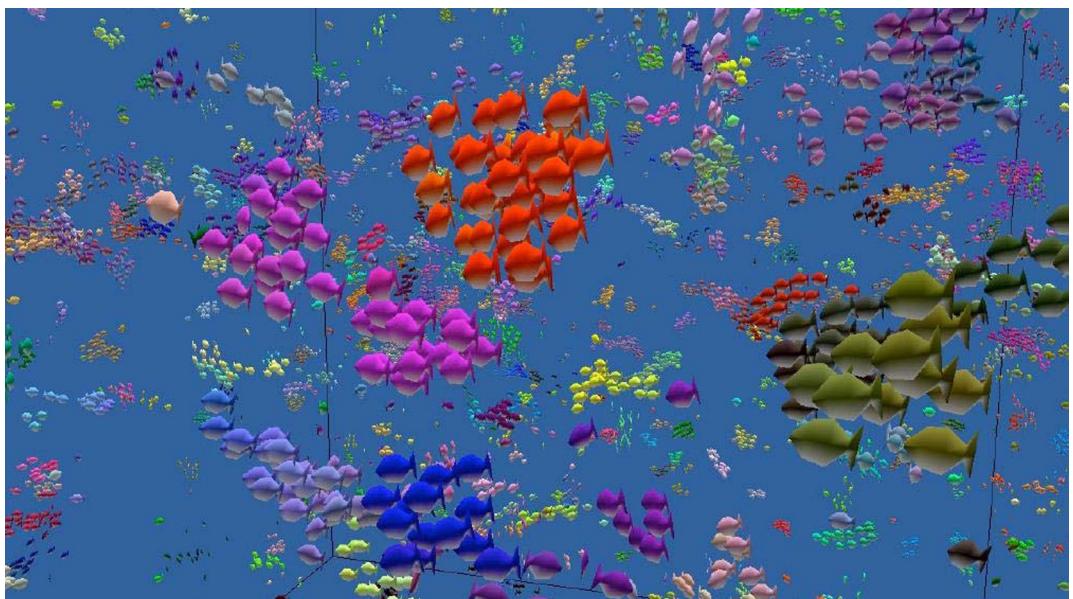


FIGURE 1.6: Ten thousand schooling fish at 60 fps in PSCrowd (Reynolds 2006) C 2006 ACM, Inc. Reprinted by permission.

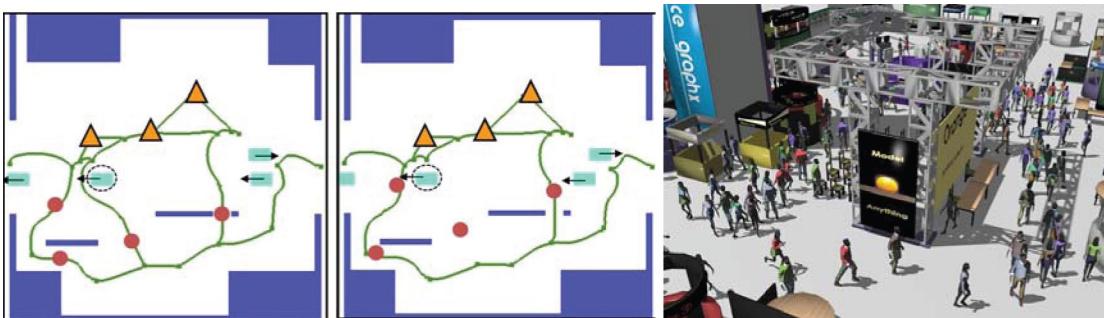


FIGURE 1.7: Adaptive Elastic ROadmaps (AERO), where the green curves represent links of the reacting deforming roadmap, and an example of a 3D animated crowd (Sud, Gayle, et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

on their individual behavior at any given moment. In general, the semantic markup of the environment — which parts of the 3D model have action or behavior significance for the occupants — is a significant obligation of the scenario author. Later, we will describe one such framework within the context of an object representation. The automated creation of such markup data remains an open challenge to crowd modelers.

Other systems use different methods to reduce complexity. To achieve a real-time simulation of very large crowds, Treuille et al. (2006) used a dynamic potential field to integrate global navigation with moving obstacles and people. This approach is not agent based, which means that we cannot have individual goals for each pedestrian; instead, goals are common to all the crowd members. In this system, motion is calculated as particle energy minimization (Figure 1.5).

Reynolds (2006) achieved real-time rule-based simulation of up to 15,000 individuals (boids) by using spatial hashing to represent a partitioning of space with a scalable multiprocessor approach (Figure 1.6).

To carry out collision avoidance while simulating each agent's navigation toward its goal at interactive rates, several methods have been introduced. Lamarche and Donikian (2004) combined topological precomputation of the geometry to enable real-time global path planning for each individual of the crowd, considering visibility and reactive behaviors. The model suffers from agents getting stuck in local minima. Sud, Gayle, et al. (2007) used adaptive elastic roadmaps (Figure 1.7) to simulate large crowds of agents at interactive rates in dynamic environments with each agent having different goals.

1.2.1 Lessons Learned From the Psychology Literature

To realistically simulate crowds with a heterogeneous behavior, it is necessary to examine the psychology literature to endow our agents with psychological factors possessed by humans that will

12 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

affect their overall behavior and movement. By using this information, we may produce crowd animation that is data driven and “prevalidated” by being based on known factors and effects. While our implementation may still need to be evaluated, this at least ensures that our foundation has an established basis.

One of the most studied situations in the psychology literature is that of crowd *evacuation*. A mass exodus from large and complex spaces, such as a public building, a cruise ship, or an unfamiliar city center, is usually hindered by a lack of knowledge of the detailed internal connectivity of the space’s rooms, passageways, or roads. In such circumstances, occupants may not be aware of the existence of suitable paths for circulation or, in the case of an emergency situation, the most appropriate paths of escape. This is a well-known problem (Sime 1984): building occupants usually decide to make use of familiar exits, which are often the way in which they entered the building. Emergency exits or exits not normally used for circulation are often ignored. When an emergency occurs, such as when a fire may be blocking some of these known paths and where smoke may also be obscuring vision, the problem can be fatally aggravated.

Spatial awareness and orientation within buildings is made possible by the five senses. The perceived space depends on the individual’s ability and psychological condition. Most people react to time pressure by an increase in the speed of their actions, as well as by subjectively choosing information. In general, the evacuation of a building due to imminent danger is accompanied by considerable physical and psychological stress. Since rising stress levels have the effect of diminishing the full functioning of one’s senses, this leads to a general reduction in awareness, especially the ability to orient oneself quickly in rooms and other surrounding areas (Waldau et al. 2003).

Decision skills in emergency situations are influenced by several factors such as the uncertainty of changes that might occur to the environment and the time pressure under which decisions have to be taken. If the agents have not been properly trained for these situations, they are likely to feel stressed and might reach the point where they find themselves incapable of making the right decision. “Stress occurs when there is a substantial imbalance between environmental demand and the response capability of the focal organism” (McGrath 1970). Time pressure is given by the difference between the amount of time available and the amount of required time to make a decision (Rastegary and Landy 1993).

In contrast, trained individuals such as firefighters deal with a dynamically changing environment and choose the best sequence of actions based on their prior experience, current perceptions, and knowledge of the environment. Their decision-making process is biased, and it is based on the importance that they assign to each evaluative dimension, such as saving lives, keeping fire from spreading, minimizing risks for themselves or the rest of the team, etc. A specific decision could sometimes lead to failure, whereas the same decision at a different moment could be the best solution.

In an evacuation situation caused by a fire, the main sources of stress could be too much or too little information coming at one time (several people in the same room making different deci-

sions and shouting different information about blocked rooms), complex and dynamically changing situations that result in uncertainty, and time pressure.

People are ultimately individuals, and even in everyday life, their behaviors vary. Researchers in psychology, physiology, sociology, and other related disciplines have been studying these personality differences for hundreds of years. This has resulted in many generalizations and proposed models. For example, one of the most accepted models of personality is the OCEAN (openness, conscientiousness, extraversion, agreeableness, neuroticism) or Five Factor model ([Wiggins 1996](#)). This model describes personality in terms of the amount of each factor present. Work in nonverbal communications has helped link these traits to behaviors ([Burgoon et al. 1989](#); [Knapp and Hall 1992](#)). For example, introverts tend to desire more personal space than extroverts ([Knapp and Hall 1992](#)). This information can be used to parameterize crowd simulations that will more closely resemble real gatherings ([Durupinar et al. 2008](#)) (Figure 1.8).

1.2.2 Main Features in Crowd Simulation Systems

There are a range of features that characterize crowd simulation systems. In this section, we introduce a number of them and then provide a brief explanation of our current system within this context.

Crowd size. This feature refers to the number of individuals that the system can simulate in real time. Some applications for building design purposes need very large crowds to measure both the overall flow rate in different parts of the environment and percentage of people that can leave the environment in a given amount of time. Other systems focus on simulating realistic human behavior within a crowd. They usually work with smaller groups and study the interaction both between individuals and with the environment.

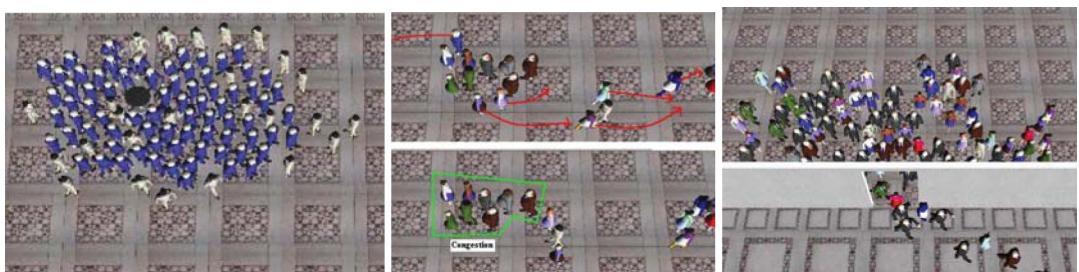


FIGURE 1.8: Different crowd behaviors based on the OCEAN model. From left to right: Ring formation where extroverts (blue suits) are inside and introverts are outside, people with low conscientiousness and agreeableness value cause congestion, and finally, neurotic and nonconscientious agents (in black suits) show panic behavior ([Durupinar et al. 2008](#)).

14 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

Goal. This feature refers to whether individuals have one main task, such as walk toward one exit, or whether they have several routes to follow. Systems that focus more on individual behaviors may also have some subgoals within the simulation, such as going through via points, helping others to an exit or to performing some specific actions.

Type of hazards. Some evacuation systems simulate drills, and therefore, there is no explicit hazard. Others simulate only fire and the way it propagates, while the most complex ones also simulate smoke propagation, and the way the level of toxicity affects the individual's performance. Other nonevacuation simulations may not need to have overt hazards because their main focus is on everyday situations, such as people navigating in a train station during rush hour.

Individuality/role. Systems that are concerned with simulating realistic human behavior within a crowd implement microscopic approaches where individuals have different decision-making processes, depending on their internal characteristics. These characteristics are generally given by a set of parameters whose values are assigned in a probabilistic manner or by the scenario author.

Communication/signaling/alarms. This feature deals with whether there is some kind of interaction between individuals and the environment. Some systems use alarms to start an evacuation process. Others implement simple signaling methods or instructions given by the firemen to indicate the evacuation routes. Nonevacuation scenarios use agent-to-agent signaling methods or, in some cases, virtual perception, along with overarching goals to drive the simulation.

Behavior method. At a lower level, an agent's directional movement choices can be implemented using a number of techniques such as: rule-based models, physical models, cellular automata, and finite state machines. At a higher level, behavioral choices can be scripted, represented by rules or automata, programmed into decision networks, planned, or driven stochastically.

Spatial structure. This feature refers to whether the space is represented by a continuous model (i.e., a 2D plane with real coordinates), or whether it is subdivided into some sort of grid (i.e., the space is divided into squares or hexagons).

Hierarchical systems. Some systems implement different levels of behavior (scripted, autonomous). Also, the crowd could be given as a multilayer architecture where behaviors are associated to individuals, groups, or the entire crowd. This allows for a wider variety of behaviors.

Environment type. This includes features such as home, train station, ship, multiple story building, or outdoor environment. The behaviors simulated should be appropriate to this context.

• • • •

CHAPTER 2

Crowd Simulation Methodology Survey

There exist many relevant crowd simulation methodologies. Here we present several systems that have been developed for animation or evacuation dynamics purposes. They all share the requirement for pedestrian movement models, but differ in techniques for generating motion paths. First we shall examine microscopic methods, then macroscopic methods, and finally detail several examples.

2.1 MICROSCOPIC AND MACROSCOPIC APPROACHES USED TO MODEL PEDESTRIAN MOVEMENTS

Many pedestrian simulation models have been developed over the years in a variety of disciplines including computer graphics, robotics, and evacuation dynamics. These can be grouped into two main methodologies: macroscopic and microscopic. Macroscopic models focus on the system as a whole (characteristics of the flow rather than individual pedestrians), while microscopic models study the behavior and decisions of individual pedestrians and their interaction with other pedestrians in the crowd.

Microscopic models describe the space–time behavior of individual pedestrians. There are two subcategories: social force models and cellular automata (CA) models. The difference between them is in the discretization of space and time. Social force models (Helbing et al.) describe pedestrian behavior microscopically by social fields (virtual “physical” forces) induced by the social behavior of the individuals. In the CA approach, the space under study is represented by a uniform grid of cells with local states depending on a set of rules that describe the behavior of the pedestrians (Chenney 2004). These rules compute the state of a particular cell as a function of its previous state and the states of the adjacent cells. Microscopic models are more interesting from the point of view of animating virtual crowds of agents with realistic autonomous behaviors; therefore, we will explain those methods in greater detail.

2.2 MICROSCOPIC MODELS

2.2.1 Social Force Models

The social force model is a microscopic approach for simulating pedestrian motion. Given “virtual” social forces analogous to real forces such as repulsive interaction, friction forces, dissipation, and

16 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

fluctuations, it solves Newton's equations of motion for each individual. This model can be successfully applied to simulate real-world pedestrian movement scenarios.

Relative to other models, social force models describe pedestrian behavior more realistically. However, they are designed to be as simple as possible. Every agent is represented in the locomotion plane by a circle with its own diameter and the model describes continuous coordinates, velocities, and interactions with other objects. Each force parameter has a natural interpretation, is individual for each pedestrian, and is often chosen randomly within some empirically found or otherwise plausible interval. Social forces model human crowd behavior with a mixture of sociopsychological and physical factors. The most important empirically derived social forces model is Helbing's model.

Helbing's model. Pedestrians $1 \leq i \leq N$ of mass m_i like to move with a certain desired speed v_i^0 in a certain direction e_i^0 , and they tend to adapt their instantaneous velocity \mathbf{v}_i within a certain time interval τ_i . At the same time, the individuals try to keep a distance from other individuals j and from the walls w using interaction forces \mathbf{f}_{ij} and \mathbf{f}_{iw} . The change of velocity in time t is given by the acceleration equation:

$$m_i \frac{d\mathbf{v}_i}{dt} = m_i \frac{v_i^0(t) \mathbf{e}_i^0(t) - \mathbf{v}_i(t)}{\tau_i} + \sum_{j(\neq i)} \mathbf{f}_{ij} + \sum_w \mathbf{f}_{iw},$$

while the change of position $r_i(t)$ is given by the velocity $\mathbf{v}_i(t = dr_i/dt)$. This model describes the psychological tendency of two pedestrians i and j to stay away from each other by a repulsive interaction force $A_i \exp[(r_{ij} - d_{ij})/B_i] n_{ij}$, where A_i and B_i are constants. The distance between the pedestrians' centers of mass is $d_{ij} = ||r_i - r_j||$, and $n_{ij} = (n1_{ij}, n2_{ij}) = (r_i - r_j)/d_{ij}$ is the normalized vector pointing from pedestrian j to pedestrian i . The pedestrians touch each other if their separation distance d_{ij} is smaller than the sum $r_{ij} = (r_i + r_j)$ of their radii r_i and r_j . If this is the case, then two additional forces are assumed inspired by granular interactions, which are essential for understanding the particular effects in panicking crowds: a "body force" $k(r_{ij} - d_{ij})n_{ij}$ counteracting body compression and a "sliding friction force" $\kappa(r_{ij} - d_{ij})\Delta v t_{ij}$ impeding relative tangential motion, if pedestrian i comes close to j . The tangential direction is $t_{ij} = (-n2_{ij}, n1_{ij})$ and $\Delta v t_{ij} = (v_j - v_i)t_{ij}$ is the tangential velocity difference. The weights k and κ represent large constants. This formulation yields:

$$f_{ij} = \left\{ A_i e^{(r_{ij} - d_{ij})/B_i} + kg(r_{ij} - d_{ij}) \right\} n_{ij} + kg(r_{ij} - d_{ij}) \Delta v_{ji}^t t_{ij},$$

where the function $g(x) = x$ if the pedestrians touch each other ($d_{ij} < r_{ij}$) and is otherwise zero.

The interaction with the walls is treated analogously. If d_{iw} is the distance to wall W , n_{iw} denotes the direction perpendicular to it, and t_{iw} the direction tangential to it, the corresponding interaction force with the wall is given by:

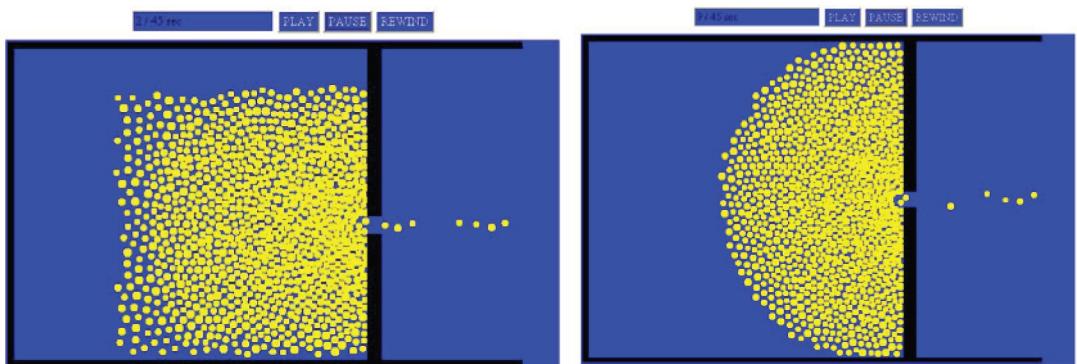


FIGURE 2.1: Helbing’s model simulation with 10,000 individuals. Reprinted by permission from Macmillan Publishers Ltd: Nature ([Helbing et al. 2000](#)), copyright 2000.

$$f_{ij} = \left\{ A_i e^{(r_{ij} - d_{iw})/B_i} + kg(r_{ij} - d_{iw}) \right\} n_{iw} + kg(r_{ij} - d_{iw})(v_i \cdot t_{iw}) t_{iw}.$$

Helbing’s social forces model applies repulsion and tangential forces to simulate the interaction between people and obstacles, which allows for realistic “pushing” behavior and variable flow rates ([Helbing et al. 2000](#)). Helbing’s model was estimated from real data. The main disadvantage of this approach is that agents appear to “shake” or “vibrate” in response to the numerous impinging forces in high-density crowds, which does not correspond to natural human behavior. Figure 2.1 shows a sequence of images from Helbing’s simulation.

There has been a considerable amount of work done using particle simulation approaches for low-density crowds. [Brogan and Hodgins \(1997\)](#) used particle systems and dynamics for modeling

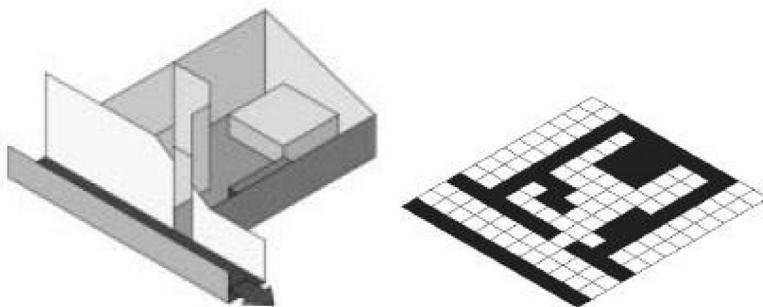


FIGURE 2.2: 3D environment and its corresponding grid of cells ([Klüpfel 2003](#)).

18 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

the motion of groups with significant physics. Musse extended the social forces model to include individualism (Braun et al. 2003).

2.2.2 Cellular Automata Models

CA (Dijkstra et al. 2000; Kirchner et al. 2003) is an artificial intelligence approach to simulation modeling defined as mathematical idealizations of physical systems in which space and time are discrete and physical quantities take a finite set of discrete values. A cellular automaton consists of a regular uniform lattice (2D array) with one or more discrete variables at each site (cells) (Figure 2.2). Walls and other fixed obstacles are black, while the white cells are areas that can be occupied by pedestrians.

The state of a cellular automaton is completely specified by the values of the variables at each cell. A cellular automaton evolves in discrete time steps, with the value of the variable at one cell being affected by the values of variables at the neighboring cells. The variables at each cell are updated simultaneously based on the values of the variables in their neighborhood at the previous time step and according to a set of local rules (Wolfram 1983). These rules describe the (intelligent) decision-making behavior of the automata, thus creating and emulating actual behavior. Each automaton evaluates its opportunities on a case-by-case basis. Global emergent group behavior is a result of the interactions of the local rules as each pedestrian examines the available cells in its neighborhood.

CA in general provide a framework for discrete models with locally homogeneous interactions. They are characterized by the fundamental properties (L, S, N, f) shown in Table 2.1.

The assumption of a regular lattice and a uniform neighborhood is compatible with geometries like those in Table 2.1 since the set of states, S , also contains information about whether a cell is accessible or not (e.g., doors or walls between cells).

TABLE 2.1: Definition of a cellular automaton (Weimar 1997)

L	Consists of a regular discrete lattice of cells
$t \rightarrow t + 1$	Evolution takes place in discrete time steps
S	Set of finite states
$F: S^n \rightarrow S$	Each cell evolves according to the same rule (transition function), which depends only on the state of the cell and a finite number of neighboring cells
$N: \forall c \in N, \forall r \in L: r + c \in L$	The neighborhood relation is local and uniform



FIGURE 2.3: Crowd simulation: (left) CA and (right) underlying 2D grid structure (Tecchia et al. 2001) C 2001 ACM, Inc. Reprinted by permission.

A configuration $C_t: L \rightarrow S$ is a function that associates a state with each cell of the lattice. The update function f changes a configuration C_t into a new configuration C_{t+1} :

$$C_{t+1}(r) = f(\{C_t(i) | i \in N(r)\}),$$

where $N(r)$ is the set of neighbors of cell r , $N(r) = \{i \in L | r - i \in N\}$. This definition assumes that f is deterministic, which may not be the case.

CA models (Tecchia et al. 2001; Kirchner et al. 2003; Chenney 2004; Torrens 2007), although fast and simple to implement, do not allow for contact between agents. Floor space is discrete, and individuals can only move to an adjacent free cell. This checkerboard approach offers realistic results for lower density crowds, but unrealistic results when agents in high-density situations are forced into discrete cells. More realistic longer (nonlocal) paths in the grid can be obtained by precomputing paths toward goals and storing them within the grid (Loscos et al. 2003).

2.2.3 Rule-Based Models

Rule-based models (Reynolds 1987, 1999) achieve more realistic human movement for low- and medium-density crowds in a flocking or swarming style, but do not handle contact between individuals and therefore fail to simulate “pushing” behavior. These models usually adopt a conservative approach by avoiding contact and, when densities are high, applying “wait” rules to enforce ordered crowd behavior without the need to calculate collision detection and response. Cognitive models have been used in combination with rule-based models to achieve more realistic behaviors for pedestrian simulation (Shao and Terzopoulos 2005). Different behavioral rules can be applied to the

20 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

crowd, group, or individuals to achieve more believable overall crowd behavior (Thalmann et al. 1999; O’Sullivan et al. 2002).

The most well-known set of local rules to simulate lifelike complex behavior is Reynolds’ “boids” model (1987). It is an elaboration of a particle system with the simulated entities (boids) represented as oriented particles with specific control rules. The aggregate motion of the simulated flock is created by a distributed behavioral model. Each simulated agent is implemented as an independent actor that navigates according to its local perception of the dynamic environment, the laws of simulated physics that rule its motion and a set of behaviors programmed by the animator. The aggregate motion of the simulated flock is the result of the dense interaction of the relatively simple behaviors of the individual simulated boids.

The basic model to simulate generic flocking behavior consists of three simple rules that describe how an individual boid maneuvers based on the positions and velocities of its nearby flockmates:

- Separation: steer to avoid crowding local flockmates
- Alignment: steer toward the average heading of local flockmates
- Cohesion: steer toward the average position of local flockmates

Each boid has access to the whole environment description, but flocking only requires reaction within a specific neighborhood that is given by a distance (from the center of each boid) and an angle (from each boid’s direction of flight). This neighborhood can be considered as a limited perceptual field. Each boid will not only avoid collision with other boids but also with obstacles in the environment.

In addition to the basic three rules used to simulate flocking, Reynolds (1999) introduced the more general concept of steering behaviors and placed flocking within that context. Steering behavior enhances the behaviors already present in the original boids model by building parts for

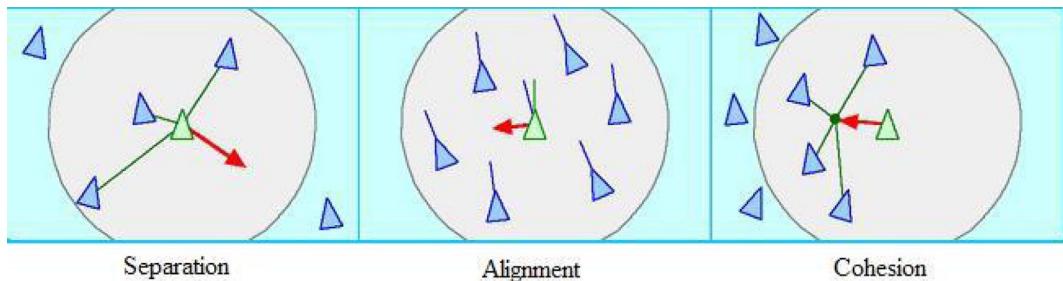


FIGURE 2.4: Reynolds’ boids (Reynolds 1987) C 1987 ACM, Inc. Reprinted by permission.

complex autonomous systems. Each of these new rules defines only a specific reaction on the simulated environment of the autonomous system.

Simple behaviors for individuals and pairs:

- seek and flee
- pursue and evade
- wander
- arrival
- obstacle avoidance
- containment
- wall following
- path following
- flow field following steering behavior

Combined behaviors and groups:

- crowd path following
- leader following
- unaligned collision avoidance
- queuing
- flocking

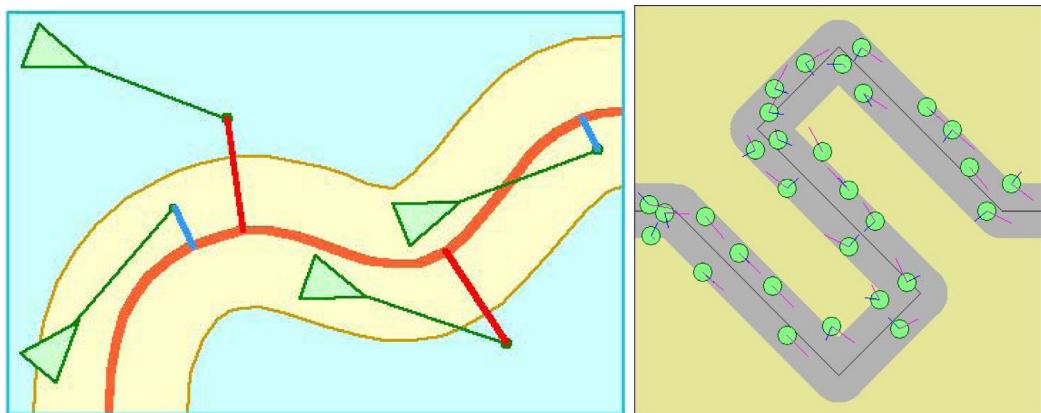


FIGURE 2.5: Example of path following from Reynolds' (1999) steering behaviors.

2.3 MACROSCOPIC MODELS

Macroscopic models abstract away from individual specialized behaviors in favor of a broader view of crowd behaviors as flows. Such models can be of value in computing and predicting traffic needs and capacities for large scale structures such as stadiums or transportation centers.

2.3.1 Regression Models

Regression models use statistically established relations between flow variables to predict pedestrian flow operations under specific circumstances. The characteristics of this flow depend on the infrastructure (stairs, corridors, etc.) (Milazzo et al. 1998).

2.3.2 Route Choice Models

Route choice models describe pedestrian wayfinding based on the concept of utility. Pedestrians choose their destinations to maximize the utility of their trip (comfort, travel time, etc.) (Hoogendoorn 2003).

2.3.3 Queuing Models

Queuing models use Markov chain models (Lovas 1994) to describe how pedestrians move from one node of the network to another. Nodes are usually rooms, and therefore links are usually portals or doors. Markov chain models are defined by a set of states together with transition probabilities. At each extrapolation step, a successor state is selected by either sampling from the transition distribution or identifying the most probable successor. The state transition probabilities are estimated from the relative frequency of transitions between behavior prototypes observed in the training data, taking the closest behavior prototype at each time instant. Only transitions causing state change are considered.

2.3.4 Gaskinetics

Gaskinetics use an analogy with fluid or gas dynamics to describe how crowd density and velocity change over time using partial differential equations (Henderson 1971).

2.4 CURRENT PEDESTRIAN SOFTWARE SYSTEMS

Commercial models of pedestrian traffic also fall into microscopic and macroscopic classes. The former studies the characteristics of individual pedestrians such as speed and interaction with other pedestrians, while the latter is concerned with groups of pedestrians rather than individual characteristics, and their analysis focuses on high-density, large-scale systems. Since we are primarily

interested in simulations where each agent is driven by its own goals and has its own personality and decision-making process, we will focus on microscopic systems.

In computer graphics, research efforts have focused mainly on portraying realistic movements in each human figure, and on the graphical techniques necessary to render huge numbers (many thousands) on commodity graphics display systems. These efforts clearly capitalize on microscopic models, but in addition address realistic body, leg, and torso movements, visual and size variety, and graphical display realism. These topics are important but outside the scope of our discussion.

This section surveys some of the most relevant systems for crowd simulation that have been developed both from industry and academia. [Other surveys on academic and commercial software can be found in papers by [Still \(2000\)](#), [Teknomo \(2002\)](#), and [Kuligowski and Peacock \(2005\)](#).]

Exodus. Exodus was developed by the Fire Safety Engineering Group at the University of Greenwich ([Galea et al. 1993](#)). The system is able to simulate the evacuation of large numbers of individuals from large multifloor buildings. By adopting fluid dynamic models, coupled with discrete virtual reality simulation techniques, the program tracks the trajectories of individuals as they make their way out of the building or are overcome by hazards (e.g., fire and smoke). The output of Exodus includes overall evacuation time, individual waiting and evacuation time, and individual paths ([Galea et al. 1993; Owen et al. 1998](#)) (see Figures 2.6 and 2.7 for some screenshots of the Exodus SW and the 3D offline tool that provides a higher-quality render of the simulation).

Pedroute. Pedroute was originally developed by London Underground Limited and has been used extensively to model crowd parameters in underground networks around the world ([Buckmann and Leather 1994](#)). It is a spatial entropy maximizing model that has been used for station design,

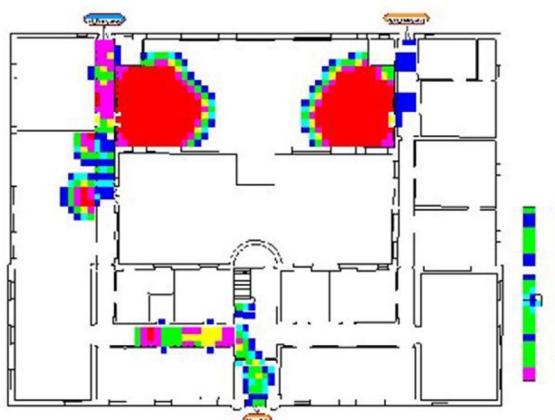


FIGURE 2.6: Density distributions during an evacuation in Exodus ([Galea 1993](#)).

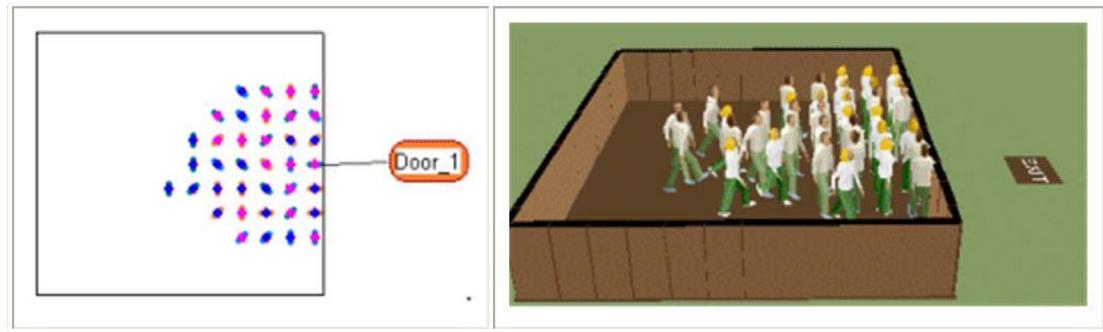


FIGURE 2.7: Simulation of an evacuation with building Exodus and its offline visualization in 3D with vrExodus (Owen 1998).

including the Olympic Railway Station, Sydney (designed to be capable of handling 50,000 passengers an hour). The model can simulate train and passenger movements going through a station or a building. The performance of the building is assessed using service levels, passenger densities, and delays and provides statistics of their journey times, congestion, and the level of service for each segment. Passengers are assigned to routes through the station using a dynamic assignment taking into account bottlenecks and congestion effects. Stations are divided into different blocks (e.g., like the CA models but larger and with continuous movement within them) representing stairs, escalators, platforms, ticket halls, etc., with each block having different speed of flow curves associated with the movement of pedestrians through them. The underlying assumptions and principles used in Pedroute are the same as other spatial interactions/entropy maximizing models and fail to incorporate the individual basic mechanisms underlying pedestrian movements. These programs cannot represent the interaction of each pedestrian with other pedestrians and the external environment, only their macroscopic behavior.

CROSSES. CROSSES (Crowd Simulation System for Emergency Situations) aims to provide a virtual reality tool for training people to effectively respond to urban emergency situations by using a model for generating and simulating a virtual crowd (Ulicny and Thalmann 2001). This multiagent system allows both scripted and autonomous behaviors of the agents (as well as interactions among them) with the virtual environment and with the real human participants. A layered approach is used for controlling agents' behavior, which is based on a combination of rules and finite state machines.

Simulex. Simulex was developed as an evacuation model with the capability of simulating a large number of people in geometrically complex buildings (P. Thompson, Integrated Environ-

mental systems, UK) (Thompson and Marchant 1994, 1995). It is based on interperson distances to specify walking speed of the individuals, and it achieves overtaking, body rotation, sideways stepping, and small degrees of back-stepping. The interperson distance is defined as the distance between the centers of the bodies of two individuals. Human body shape is represented by an elliptical body size defined by one main circle, and two smaller circles bounding each shoulder.

The 2D locomotion space is continuous for pedestrian movement, but discretized to calculate and store a distance map (Figure 2.8). The distance map is used to direct occupants to the closest available exit. The velocity of each individual depends on the distance to the people ahead.

Rampage. Rampage is a particle simulation system to animate explosions and other elementary primitives; it divides human behavior into reflex reactions and decision making based on knowledge obtained from the scene (Bouvier and Cohen 1995). Its principles for human behavior simulations are based on the Boltzmann gas equation.

Egress. AEA Technology started the development of Egress in 1991 (AEA Technology 2002). It is a commercial software tool for crowd simulation. The model employs artificial intelligence techniques to determine how a person would react under a variety of circumstances such as fire and smoke. The output of Egress includes evacuation time analysis, comparison between people evacuation and progression of hazard, and potential structural and procedural improvements. The simulation is based on hexagonal grids. The approach is fundamentally a cellular automaton process in which the transition of people from cell to cell is based on cell occupancy.

Legion. Legion was not designed as a crowd behavioral analysis system but as an investigational tool for the study of large scale interactive systems (Legion International 2003). The computational model oversimplifies the behavioral representation of individuals. First, the model employs only four parameters (goal point, speed, distance to others, and reaction time) and one decision rule

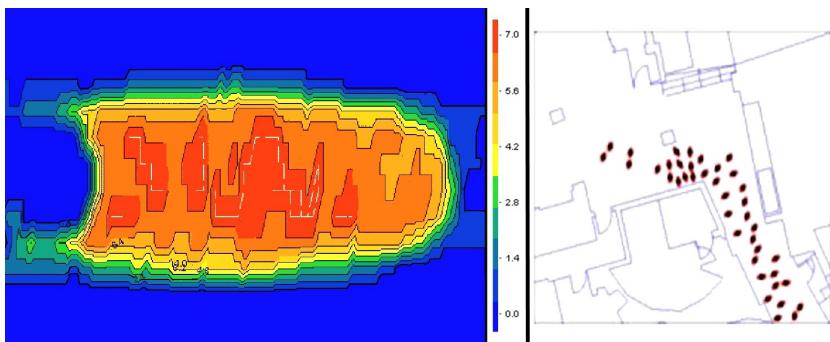


FIGURE 2.8: Crowd density and simulation in Simulex (Still 2000).

26 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

(based on assumption of the least-effort principle) to represent the complex nature of individual behaviors. Furthermore, all individuals are considered to be the same in terms of size, mobility, and decision-making process, and the model ignores social behaviors such as herding and leader influence (Still 2000).

The Legion model works in 2D continuous space, which gives more realistic paths for the pedestrians than those based on discrete grids (Figure 2.9). Legion claims that occupant movement is in agreement with extensive empirical research through analysis of video footage of crowd movement and behavior. Movements also depend on specific features of the local geometry, input variables specified for each person, an individual's knowledge of the environment, and state of readiness (meaning interaction with signals) (Legion International 2003).

STEPS. STEPS is an agent-based model with coarse grid geometry (CA) (Figure 2.10) (Mott-MacDonald 2003). Each individual occupies one cell at any given time and moves in the desired direction if the next cell is empty. Each occupant has its own characteristics, patience factor, and familiarity behavior.

In STEPS, the fundamental driving mechanism for individual movement is the desire to move at a free walking speed toward the next target point in the shortest amount of time and without collision. The decision process is adhered to by every individual in the model. For each target (exit point), a potential is calculated at each grid cell on the plane. The potential value represents the distance between individual cells and the targets considering the presence of blockages (walls, columns, etc.).

At every time step (0.1 s), each target is scored based on the time of arrival. The patience level modeled into the individual or group is incorporated into the calculation, and a final score is

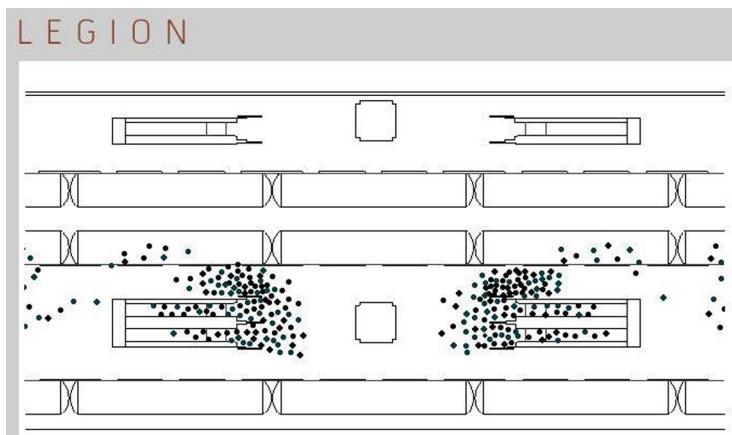


FIGURE 2.9: Legion simulation of a train station (Legion 2003).

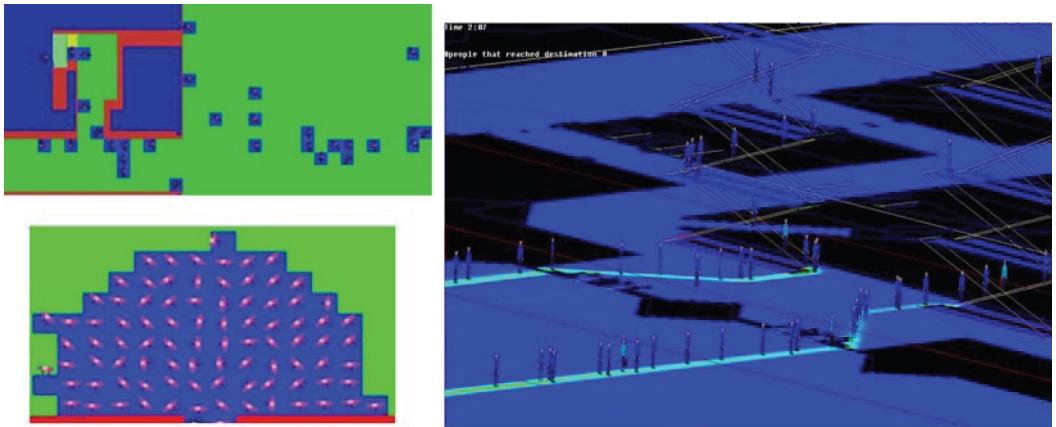


FIGURE 2.10: STEPS pedestrian simulation where we can appreciate the underlying 2D grid. Left image shows the occupied cells in blue, walkable cells in green, and nonwalkable cells in red. On the right image, we can see a simulation with stairs and most common paths in green ([Mott-MacDonald 2003](#)).

derived. Based on the derived final score of the target, the individual located in a cell attempts eight possible directions at every time step to reach the lowest-scored target.

ViCrowd. ViCrowd represents a model to automatically generate human crowds based on group properties instead of individuals ([Musse and Thalmann 1997](#); [Musse et al. 1998](#)). The individuals within a group would follow the groups' specifications instead of their own to satisfy real-time requirements. It is based on Reynolds' flocking system but includes a simple definition of behavioral rules using conditional events and reactions.

A sociological model is used to handle affinities and repulsion effects that can emerge in crowd simulation and create more complex behaviors. In this approach, control is presented through different degrees of autonomy: guided, programmed, and autonomous crowd, ranging from totally interactive to totally autonomous control.

OpenSteer. OpenSteer provides a toolkit of steering behaviors defined in terms of an abstract mobile agent (Figure 2.11) ([Reynolds 1999](#)). It is the C++ implementation of Reynolds' steering model described in Section 2.2.

Massive SW. This system is based on artificial life technology, using a combination of very simple rules with fuzzy logic, developed by Massive Software, Inc. ([Massive Software, Inc. 2005](#)). It is a 3D animation tool to simulate large crowds for the special effects industry (Figure 2.12). Agents are endowed with synthetic vision, hearing, and touch that allow them to react naturally to their environment. In Massive SW, reactions rely on the environment rather than an internal model and the agents respond directly to environmental stimuli, using less storage than modeling the environment internally in each agent. The agents have very simple "brains," since Massive SW

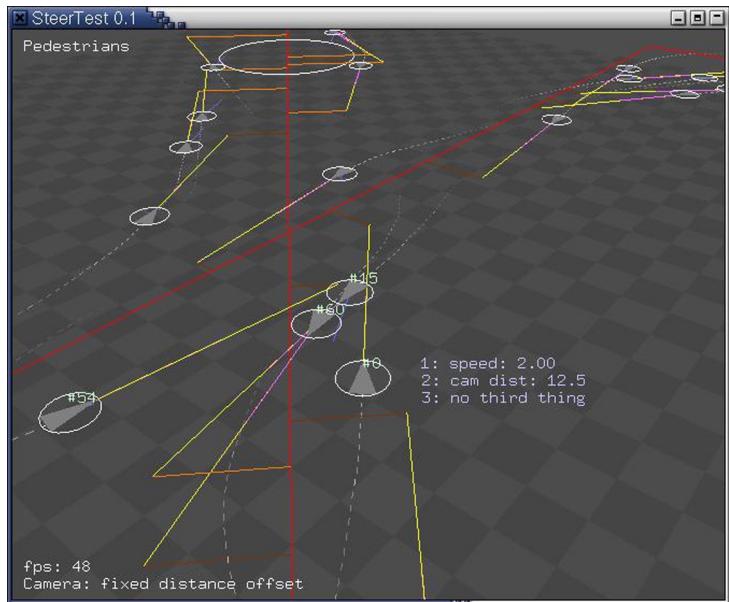


FIGURE 2.11: Screen shot of OpenSteer pedestrians demo (Reynolds 1999).

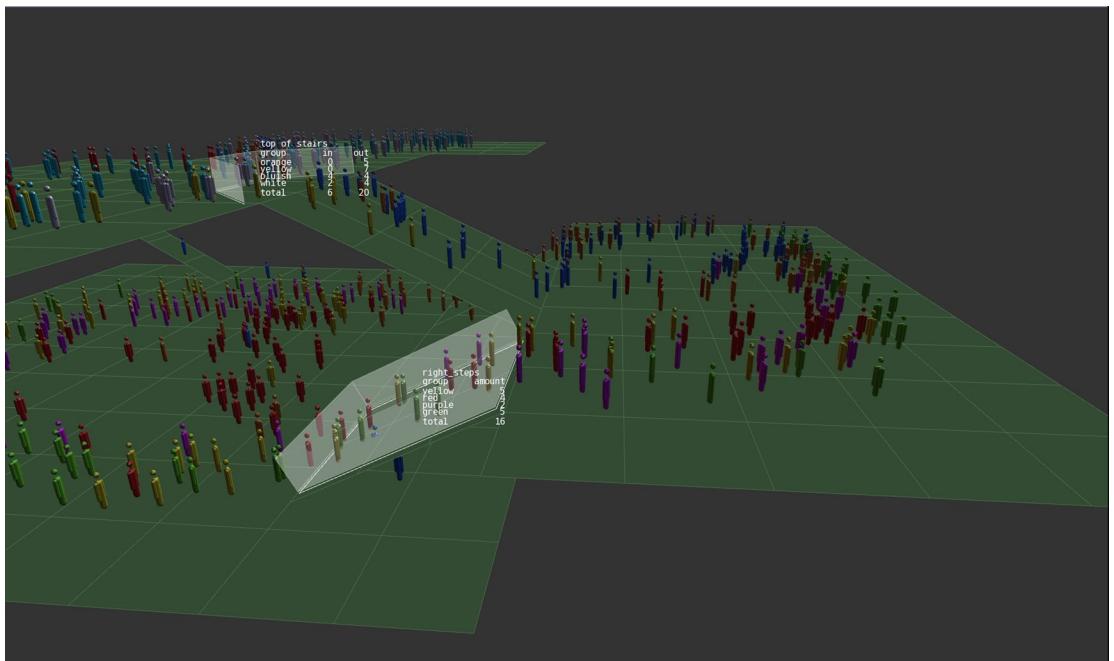


FIGURE 2.12: Evacuation simulation with Massive SW for architectural design and evaluation (Massive Software, Inc. 2008).

has been designed to achieve realistic simulation for short periods (under 5 s) but does not deal with achieving long-term goals and global navigation issues.

Reactive Navigation. The agents of Lamarche and Donikian (2000, 2004) move within complex virtual environments represented with a hierarchical topological structure extracted from the geometry of the virtual environment (Figure 2.13). This structure allows fast path finding as well as an efficient reactive navigation algorithm. To avoid collision while reaching their targets, they use an iterative optimization process. Collision prediction is based on neighborhood computations; it creates long-distance neighborhood relations in sparse crowds and short-distance relations in dense crowds. Collision avoidance is achieved by predicting other agents' positions, and if a collision may occur, then the agent will modify its speed and orientation vector.

Artificial Fishes. Tu and Terzopoulos(1994) developed an artificial life approach to simulate the appearance, motion, and behavior of individual fish in a virtual marine world and also the complex group behaviors observed in real aquatic ecosystems (Figure 2.14). Each fish behaves as an autonomous agent exhibiting fish behavior such as foraging, preying, schooling, courting, and collision avoidance. These simple fish models can learn basic motor functions and perceive the environment.

ACUMEN. ACUMEN is a system for synthesizing and recognizing aggregate movements in a virtual environment with a natural language interface (Allbeck et al. 2002). Its principal components include an interactive interface for aggregate control based on a collection of parameters extending an existing movement quality model, a feature analysis of aggregate motion verbs, recognizers to detect occurrences of features in a collection of simulated entities, and a clustering algorithm that determines subgroups.

The ACUMEN system used the Parameterized Action Representation (Bindiganavale et al. 2000) to capture the semantics of aggregate movement for generation and recognition. Movement is based largely on a particle system-like model of group simulation, using dynamic forces acting on rigid bodies to produce the desired movement (Figure 2.15).

ACUMEN extended the EMOTE features (Chi et al. 2000) to group movement by examining features that characterized verbs that referred to aggregate motions. EMOTE was inspired



FIGURE 2.13: Reactive navigation (Thomas 2000) C 2000 IEEE.



FIGURE 2.14: Artificial fishes (Tu 1994) C 1994 ACM, Inc. Reprinted by permission.

by movement observation science, in particular Laban movement analysis (Maletic 1987) and its “effort and shape” components.

Autonomous Pedestrians. Terzopoulos has been an advocate of an artificial life approach integrating motor, perceptual, behavior, and cognitive components within a model of pedestrians as individuals (Shao and Terzopoulos 2005). The environment is represented through hierarchical data structures that efficiently support perceptual queries from the autonomous pedestrians that drive their behavioral responses and maintain their ability to plan their actions on a local and global level.

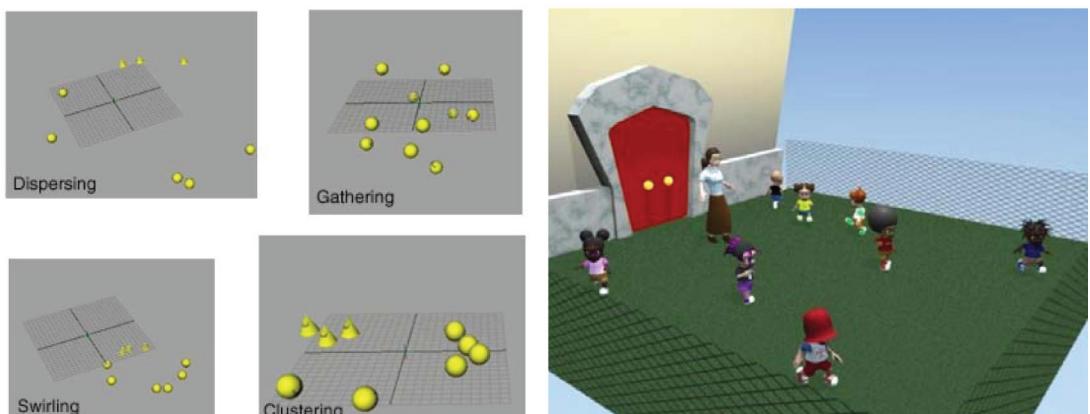


FIGURE 2.15: ACUMEN system applied to spheres and school children (Allbeck 2002) C 2002 ACM, Inc. Reprinted by permission.

Agents perform six basic reactive behavior routines: avoid static obstacle, avoid static obstacle in a complex turn, maintain separation in a moving crowd, avoid oncoming pedestrians, avoid dangerously close pedestrians, and verify new directions relative to obstacles. Agents perform collision avoidance but not response; therefore, if an intersection with another pedestrian is about to happen, which they detect by using a “front safe area,” the agent will stop, try to turn to face away, and wait until space is available around its current position (Figure 2.16).

This work also includes a cognitive model for controlling action selection. This selection mechanism is based on mental state variables that are akin to needs (Shao and Terzopoulos 2007). A probabilistic decision network model has also been incorporated into this work (Yu and Terzopoulos 2007). Decision trees are constructed for different behaviors, and their probabilities are tuned to obtain the desired overall behaviors.

Space Syntax. Urban planners have proposed Visibility Graph Analysis for pedestrian movement (Turner and Penn 2002) (Figure 2.17). This method examines how visually accessible points are within an area to determine a pedestrian’s likely navigation direction. This approach focuses on using the visual field to guide natural movement, without considering other socioeconomic factors (reaching for a destination, meeting with people, picking up something, etc.) or granular physics (moving particles toward a goal, pushing, forming lanes, etc.). To simulate a relatively large crowd, they calculate an “exosomatic visual architecture” where the connections between mutually visible locations within a configuration are prestored in a lookup table.

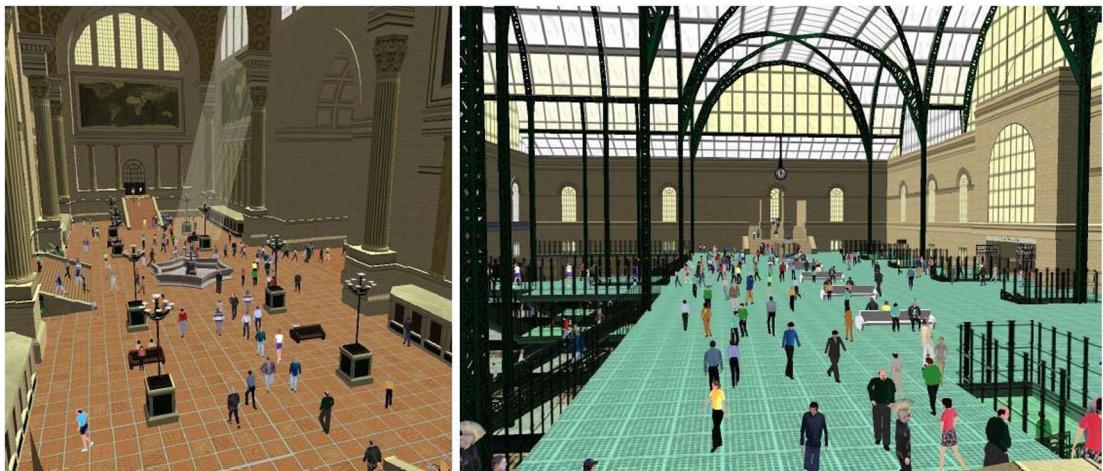


FIGURE 2.16: Autonomous pedestrians (Shao and Terzopoulos 2005) C 2005 ACM, Inc. Reprinted by permission.

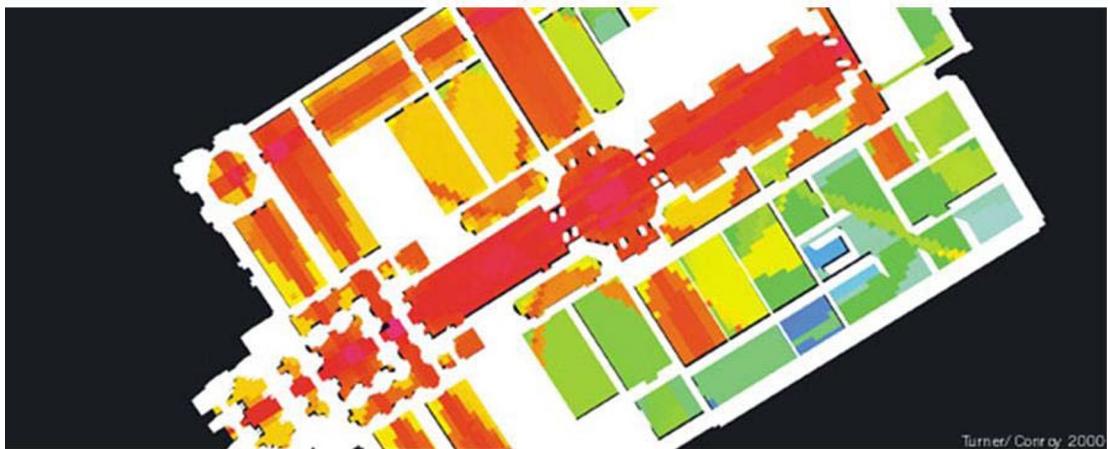


FIGURE 2.17: Visibility graph from Space Syntax ([Turner and Penn 2002](#)).

2.5 SUMMARY OF CROWD MODELS

There has been considerable research and development in crowd or group simulation, especially in the study of evacuation dynamics. Nevertheless, there is still room for improvement.

The main focus of commercial applications is to validate their systems in terms of egress (flow rates, densities, congestion areas, evacuation times, etc.). They use either macroscopic or microscopic approaches. The microscopic models most commonly used in industry applications are particle simulation and CA models. These methods have proven to lack realism when they are applied to 3D virtual humans for animation systems because they either look closer to particles than to real human movement (social forces model and particle simulation in general) or are restricted to checkerboard configurations (CA).

In contrast, research on developing autonomous agents has focused on their ability to navigate large complex virtual environments while avoiding static obstacles and other agents. Most cases, however, ignore the problems that arise when dealing with very high-density crowds. These systems usually apply some sets of rules to avoid collision based on modifying the speed or trajectory of the agents. Consequently, these models are sufficient for medium- and low-density crowds; however, when the crowd is very dense, they yield unnatural emergent behavior such as individuals stopping and waiting for space to clear up. There is no concept of body-to-body contact leading to pushing agents in a crowd or individuals being dragged by the crowd. Further effects such as falling, injury, incapacitation, and others walking over the fallen agent are also ignored. The system HiDAC has as its main goal to deal with all these features that emerge in real high-density crowds ([Pelechano et al. 2007](#); [Sunshine-Hill et al. 2007](#)).

In terms of global navigation, the published systems for crowd simulation assume that agents have complete information about the environment. An agent can access the entire internal structure of the environment and use algorithms such as A* (several techniques have been exploited to achieve real time when performing global path planning for large groups of agents), or else the environment is discretized as a grid that stores potentials or distance maps that the agents will follow locally to reach the goal.

It is essential to provide an agent with the ability to explore partly known environments and learn new features. Another crucial aspect of crowds that is ignored elsewhere is that people have the ability to communicate with others to exchange salient navigational information.

Our work focuses also on improving the realism of the agents' behavior by allowing them to have partial information about the environment and be able to extend their memory (or mental maps) as they explore the environment and communicate with other individuals in the crowd. Agents can also exhibit different behaviors based on different roles (Pelechano and Badler 2006).

Finally, some psychological factors need to be incorporated with the purpose of modifying the overall performance of an individual based on its mental state.

Table 2.2 shows a comparison of some of the most significant models in crowd animation. This emphasizes the main features in multiagent simulations and pedestrian evacuations to compare the contributions of our model (MACES + HiDAC) to others.

2.5.1 Some Limitations of the Current Commercial Software for Crowd Evacuation

Since most of the commercial tools for crowd evacuation are based on the CA model (e.g., STEPS, Exodus, Egress), it is important to understand their movement simulation artifacts. We also describe in this section some of the limitations that current commercial software tools have in terms of simulating human psychology and physiology (Pelechano and Malkawi 2008). Andersen et al. (2005) provide a more detailed discussion regarding the limitations of grid-based pedestrian simulation models.

Grid Size. Using a CA model, and therefore having a discrete grid for the simulation, creates several limitations. Some of the main problems that occur are fixed densities and unrealistic flow rates through portals. Grid size becomes a crucial parameter to calibrate in order to achieve the desired behavior. Individuals will move with their desired velocity unless all of the cells around them are occupied or blocked, which causes the person to wait for the next empty cell in the desired direction of movement.

Having a fixed grid size limits the maximum densities achievable. For example, if the grid size is defined as 0.5 m^2 the maximum density at any time will be 4 persons/m^2 , while the literature

TABLE 2.2: Comparison of different systems for animation of large groups

	COLLISION RESPONSE	PARTICLES SHAKING CORRECTED	BEHAVIOR METHOD
Social forces (Helbing)	Yes	No	Forces
Rule based	No	Not needed	Rules
CA	No	Not needed	CA
Simulex	Yes	No	Distance maps
Egress	No	Not needed	CA
ViCrowd	No	Not needed	Rules + FSM
OpenSteer	No	Not needed	Rules
Legion	No	Not needed	Least-effort
Exodus	No	Not needed	CA
Steps	No	Not needed	CA
Massive SW	No	Not needed	Rules + fuzzy logic
Reactive Navigation	No	Not needed	Rules
Artificial fishes	No	Not needed	Rules
ACUMEN	Yes	No	Particle simulation
Crosses	No	Not needed	Rules+ FSM
Autonomous Pedestrians	No	Not needed	Artificial life approach
Space Syntax	No	Not needed	Visibility graphs
MACES + HiDAC	Yes	Yes	Extended social forces

in crowd behavior reports densities of 7.4 persons/m² where people can still move ([Andersen et al. 2005](#)).

A second issue arises when the grid is not accurately aligned with the geometry. This can lead to the appearance of artifacts where only one person at a time can get through a door in the simulation, when in reality the door size is big enough to fit two people crossing simultaneously. An example of this situation can be observed in Figure 2.18.

Fatigue Factor. Fatigue factor is not included in the simulation. The speed values given in the literature are based on those collected during fire drills and normal situations. During an actual

	COMMUNICATES OR SIGNALS	INDIVIDUALS OR ROLES	REAL TIME	LEARNING	SPATIAL STRUCTURE FOR MOTION
Social forces (Helbing)	No	Some	Yes	No	Cont.
Rule based	No	No	Yes	No	Cont.
CA	No	No	Yes	No	2D grid
Simulex	No	Some	No	No	Cont.
Egress	No	Some	No	No	Hexagonal grid
ViCrowd	No	Yes	Yes	No	Cont.
OpenSteer	No	Some	Yes	No	Cont.
Legion	some	Some	No	No	Cont.
Exodus	No	Some	No	No	2D grid
Steps	No	Some	No	No	2D grid
Massive SW	No	Yes	No	No	Cont.
Reactive Navigation	No	Yes	Yes	No	Cont.
Artificial fishes	No	Yes	Yes	Yes	Cont.
ACUMEN	Yes	Yes	Yes	No	Cont.
Crosses	No	Yes	Yes	No	Cont.
Autonomous Pedestrians	No	Yes	Yes	No	Cont.
Space Syntax	No	No	Yes	No	Cont.
MACES + HiDAC	Yes	Yes	Yes	Yes	Cont.

fire evacuation in a high-rise building, slower speeds when walking downstairs have been reported. This can be the result of fatigue when walking downstairs for long periods (unfit, older, or disabled people). Fatigue motivates people to take rest stops, which can then precipitate bottlenecks.

Speeds in Stairwells. In some reported scenarios (e.g., the WTC attack), the observed speeds during the real evacuation descending stairwells was 0.2 m/s, which is half the slowest speed given by egress when walking downstairs. The reason why these speeds occurred was because there was an ascending counterflow of firefighters that was blocking the downstairs flow. In 1-m-wide stairs, it should be possible to have two flows of people moving in opposite directions, but since one of



FIGURE 2.18: Only one person at a time can fit through the opening of the stairwell, when the actual width is 1 m (Mott-MacDonald 2003).

those flows contains firefighters carrying all their gear, it turns out that one of the flows needs to completely stop to let the other move.

Route Selection. Path finding in grid-based models consists of traversing the centers of squared cells. Distances between centers can be stored before the simulation takes place. The method is usually based on “potential maps,” which identifies a discrete approximation of the shortest path toward the destination and stores this information in the cells to achieve an efficient simulation. The main problem that potential maps have is that they favor 45-degree diagonal movement, and the resulting routes are not always realistic.

Figure 2.19 shows the unnatural paths followed by the people in Exodus (gray paths) compared with some of the real paths that should have been followed if the space was continuous (red dotted lines).

Potential maps computed on grids have the following problems (Andersen et al. 2005): they yield highly unrealistic space utilization, cannot guarantee equivalence on return trips, artificially segregate opposite flows, and distort path length and thus pedestrian travel time.

Uneven Use of Stairwells. Uneven use of stairwells occurs because of familiarity or initial distance to exits, which leads to different utilization of the stairs. Because distances are computed before the simulation and route selection is based on the potential maps, some stairwells may attract more individuals than others (Figure 2.20). This can have a considerable impact on the overall evacuation time. This can be a positive emergent behavior if it matches with what would actually

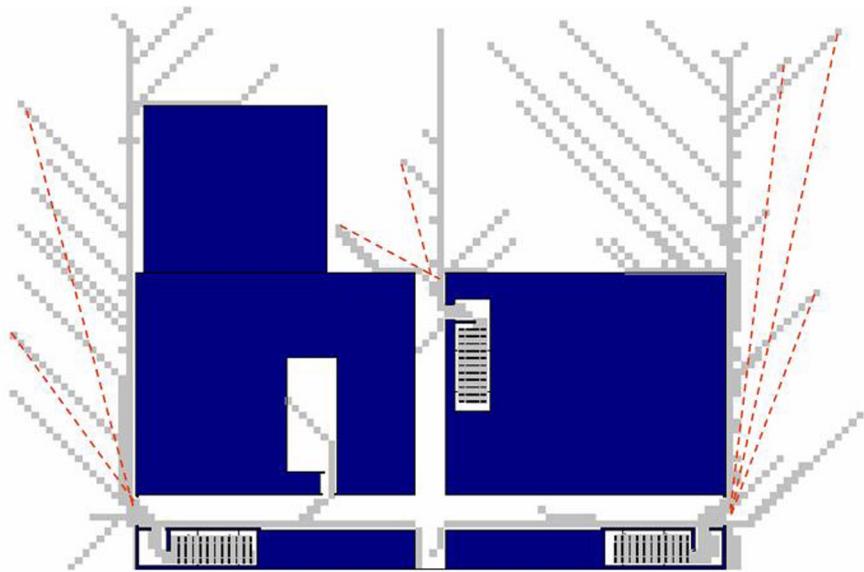


FIGURE 2.19: Paths followed based on potential maps in Exodus ([AEA-Technology 2002](#)).

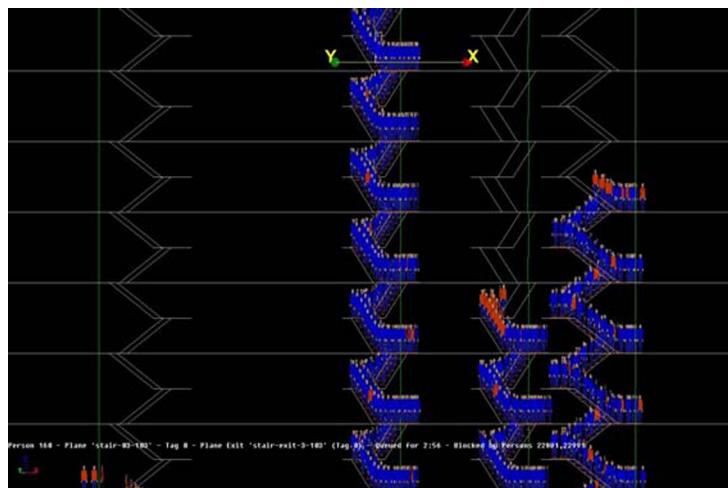


FIGURE 2.20: Uneven utilization of stairwells during route selection ([Mott-MacDonald 2003](#)).

occur in the real building, but how to validate the positive impact of this behavior in the accuracy of the results is unclear.

2.6 NAVIGATION

Coordinating the movement of groups of agents plays an important role to simulate swarms of robots, animals, and pedestrians in computer graphics and civil engineering applications. Most research focuses on techniques for modeling individual behavior of flocks inspired by Reynolds' boids (Reynolds 1987) and Helbing's social forces models (Helbing et al. 2000) when moving in continuous space and CA when dealing with discretized grids.

For continuous space, rule-based models and social forces models can be sufficient for simple environments where agents cannot get locked in local minima; they will have difficulties, however, when simulating larger and more complex environments. To navigate a complex environment, some semantically meaningful geometric representation of the environment is essential. Among the most popular techniques for crowd navigation are cell and portal graphs (CPGs) (Pettre et al. 2005; Lerner et al. 2006; Pelechano and Badler 2006), potential fields (Galea et al. 1993; Thompson and Marchant 1994; Chenney 2004), and roadmaps (Kavraki et al. 1996; Bayazit et al. 2002; Sung et al. 2005).

In CA, navigation can be performed through grid-based search using A* algorithms, potentials, or flow tiles. Computer games have commonly used A* search to generate group motion (Lau and Kuffner 2005). In this approach, the environment is divided into a heterogeneous grid, and the search is based on expanding toward the most promising neighbor of already visited positions. Although A* can find the shortest path to a goal and several improvements have been added to achieve fast solutions, it is still necessary to run the algorithm again to find a new path for each new goal and for each agent in the group. Alternatively, potentials or flow tiles preprocess the required path information and then store it within each cell, so during the simulation, each agent will query the cell for navigation information.

2.6.1 Cell and Portal Graphs

CPGs are often used to abstract the geometry. They were introduced by Teller in 1992 (Figure 2.21). In a CPG, navigation becomes a problem of getting from one node of the graph to another through a sequence of nodes and portals (Pettre et al. 2005; Pelechano and Badler 2006). When employed for indoor scenes, nodes usually represent the rooms defined by their enclosing walls and portals correspond to the doors. On top of that partition, an adjacency graph is built where each portal connects the two rooms on both sides of the door. Outdoor environments can also be represented with CPGs where cells are pedestrian pathways and portals appear between pedestrian pathways and crossings (Lerner et al. 2006).

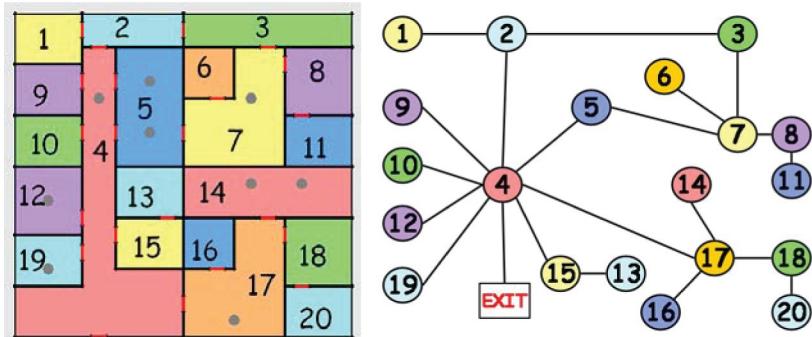


FIGURE 2.21: Floor plan of a building and its corresponding CPG (Pelechano et al. 2005).

2.6.2 Flow Tiles and Potential Field Methods

In potential field methods, the environment is usually discretized into a regular grid. Then a potential is associated with each cell, which corresponds to the sum of a repulsive potential generated by obstacles in the environment and attractive potential generated by the goal. Therefore, gradient methods can be applied to find a path from any origin in the environment to a goal position. The method has some problems, such as local minima where the individuals could get stuck and never reach the goal (Thompson and Marchant 1994; Owen et al. 1998; Loscos et al. 2003).

Flow tiles offer a similar approach (Chenney 2004) where tiling can be constructed to meet a wide variety of external and internal configurations. Each flow tile contains a small precomputed vector field, and concatenation of multiple tiles can produce large flows.

Dynamic potential fields (Treuille et al. 2006) have been used to integrate global navigation with moving obstacles and people, efficiently solving the motion of large crowds without the need for explicit collision avoidance (Figure 2.22).

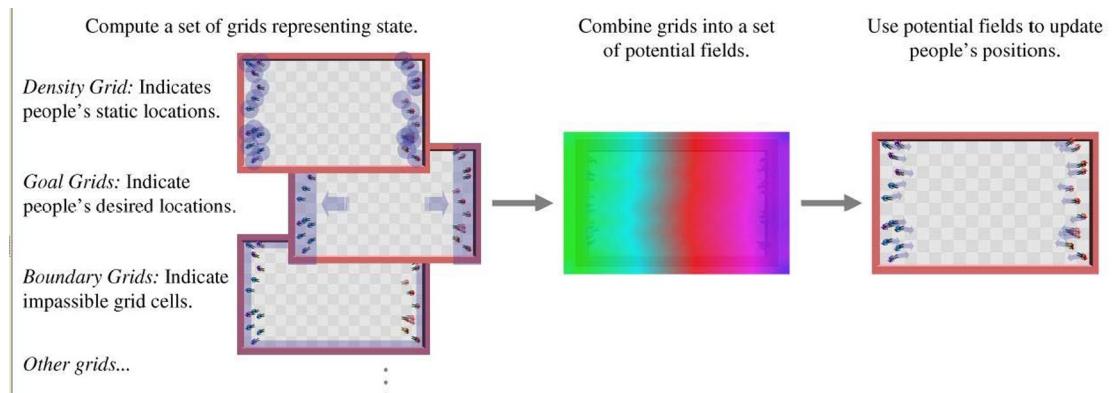


FIGURE 2.22: Continuum crowds general algorithm overview (Treuille et al. 2006) C 2006 ACM, Inc. Reprinted by permission.

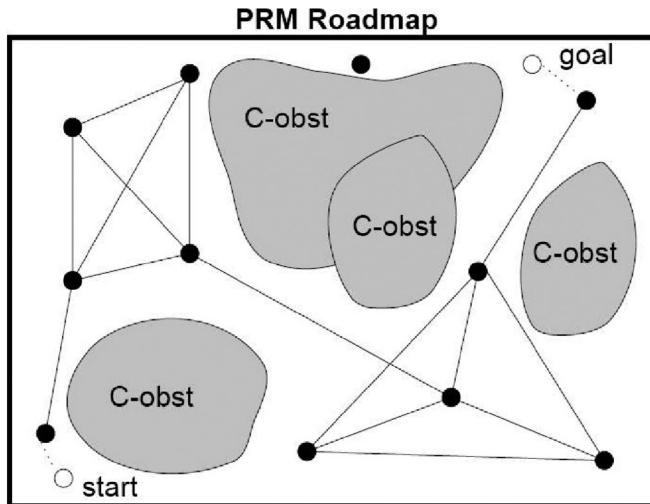


FIGURE 2.23: Example of a PRM (Bayazit et al. 2002) © 2002 IEEE.

2.6.3 Probabilistic Roadmaps

Probabilistic roadmaps (PRMs) have been widely used in robotics and navigation for autonomous agents. The basic idea of a PRM (Kavraki et al. 1996) consists of computing a very simplified representation of the free space by sampling configurations at random. Then the sampled configurations are tested for collision, and each collision-free configuration is retained as a “milestone.” Each milestone is linked by straight paths to its k -nearest neighbors. Finally, the collision-free links will form the PRM (Figure 2.23).

PRMs have been used to generate navigation paths for large groups of autonomous agents. Bayazit et al. (2002) simulate crowds with various group behaviors like homing, shepherding, and exploring combined with PRMs to drive the characters toward a goal or to explore a scene; their main motivation is to expand flocking behaviors by endowing the agents with some global information about the environment. Lien et al. (2005) extend that work by introducing multiple shepherds and allowing them to coordinate without communication. They also incorporated dynamic roadmaps by modifying edge weights as an implicit means of communication between flock members (Bayazit et al. 2002).

2.7 ENVIRONMENT MODELING

Haumont et al. (2003) presented an algorithm for volumetric cell-and-portal generation for indoor scenes based on an adaptation of the 3D watershed algorithm. The watershed is created using a

flooding analogy in the distance field space. Flooding starts from local minima, and each minimum produces a region (room). Portals appear where regions meet during the growth. The algorithm automatically classifies each room as a cell and the openings (doors and windows) as portals, generating the CPG of any indoor environment.

Pettré et al. (2005) introduced navigation graphs for multilayered and uneven terrain based on some motion planning methods from robotics (Hait et al. 2002; Chenney 2004). In this approach, the space is divided into free space and obstacles to be avoided. A Voronoï diagram of the free space is calculated and then collision-free convex cells are built along the diagram. The navigation graph is obtained from the adjacency graph of the cells. The novelty of this work is in extending the basic navigation graph to multilayered terrain by classifying some free-space areas as obstacles based on the slope of the terrain (Figure 2.24).

Lerner et al. (2006) presented a method to efficiently create CPGs for both interior and exterior environments. The algorithm input is a set of half edges in 2D that can be extracted from the geometry. They use a two-pass algorithm: the first step creates an initial partition and then the second step refines it. Their heuristic strives to create small portals as a means for generating an effective partition. The method supports incremental changes of the model by locally recomputing and updating the partition.

Shao and Terzopoulos (2005) represent virtual environments by a hierarchical collection of maps: (a) a topological map, which represents the connections between different parts of the

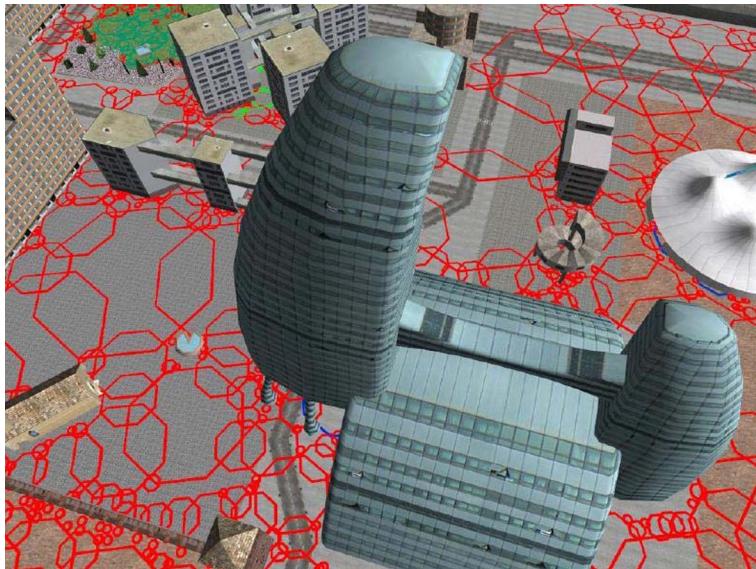


FIGURE 2.24: Navigation graphs (Pettré et al. 2005).

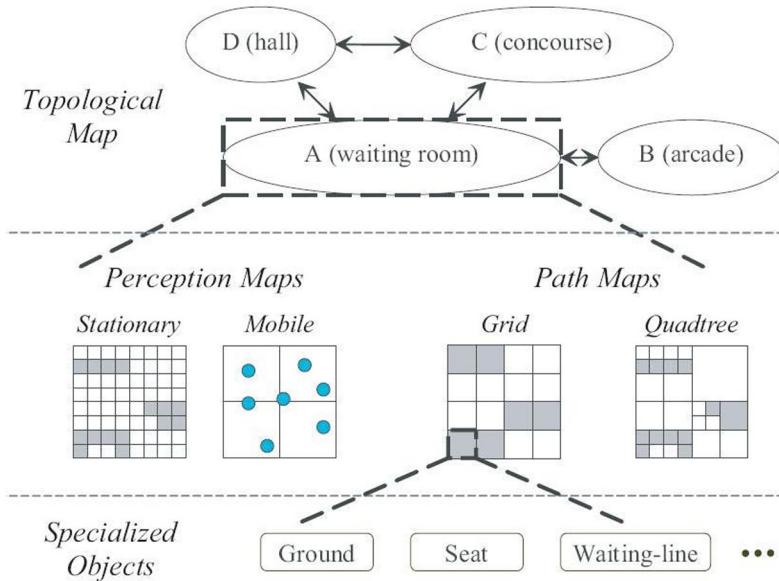


FIGURE 2.25: Hierarchical representation of a building (Shao and Terzopoulos 2005) C 2005 ACM, Inc. Reprinted by permission.

virtual world; (b) perception maps, which provide information regarding perceptual queries; and (c) path maps, which enable online path planning for navigation (Figure 2.25). The topological map contains nodes corresponding to the environmental regions and edges representing accessibility between regions. The path maps include a quad-tree map, which supports global, long-range path planning, and a grid map, which supports short-range path planning.

CHAPTER 3

Individual Differences in Crowds

Real populations and crowds are not homogeneous: they are composed of individuals who differ in a variety of ways. Naturally, their external appearances vary, which can be accomplished by using different graphical models and materials. Members of real crowds also differ in their behaviors. These behavioral variations stem from a range of factors, from roles to personality to environment context. We will outline a few in this chapter.

3.1 PERSONALITY AND EMOTION MODELS

A major focus of the autonomous agents research community is the modeling of personality (Rousseau and Hayes-Roth 1996; Moffat 1997; Trappl and Petta 1997; Ball and Breese 2000) and emotions (Ekman and Friesen 1977; Collier 1985; El-Nasr et al. 1999; Lester et al. 2000; Schroder 2001; Marsella and Gratch 2002; Gratch and Marsella 2004; Silverman et al. 2006). Including these aspects of individual differences is meant to create more believable characters with natural behavioral variations. Mapping these traits to their behavioral effects can be difficult, time-consuming, and scenario-specific. Many implementations only use one or two dimensions of the corresponding psychological models (Ortony et al. 1988; Wiggins 1996) and craft the mapping to behaviors for limited scenarios.

Emotions and mood affect many behaviors and channels of nonverbal communication. The effect of emotions on facial expressions is well-known and well-studied (Ekman and Friesen 1977), but other channels are effected as well. Lewis (1998) indicates that tense moods cause postures that are rigid and upright or slightly leaning forward. Extreme inhibition tends to cause withdrawal movements and general motor unrest. When depressed, movements are slower, fewer, and hesitating. By contrast, elation causes fast, expansive, emphatic, spontaneous movements. The embodied agents research community has studied emotion and mood more than any of the other cognitive processes (Cassell et al. 2000).

In terms of spatial relations between individuals, introverts generally prefer greater interpersonal distances. Aggressive and violence-prone (not agreeable) individuals tend to need even greater interpersonal distances to feel comfortable. Introverts also tend to resist visual interaction. People

44 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

who are more neurotic and introverted have more restrained and rigid behavior and display more uncoordinated, random movements (Burgoon et al. 1989). More details on a mapping of personality traits to individuals in crowds can be found in the work of Durupinar et al. (2008).

Like the other processes described below, the modeling of personality may lead to more consistent characters, and because personality is a pattern of behavior (longer temporal extent), it should lead to more consistent behavior from situation to situation. This may help observers of the character to develop a deeper sense of “knowing” the character.

3.2 PHYSIOLOGY

An individual’s physiology and needs will also impact its decision-making and external behavior (Maslow 1943). If we examine a crowded shopping mall or an office building, it would not be unusual to find at least a few people eating or drinking. Likewise, we might find someone dozing on a bench or even in an office. Implementations of some of these needs in the form of energy and health levels are common in many video games, but fundamentally emphasized in *The Sims* (Wright 2008). The basis of game play in *The Sims* is to try to fulfill the characters’ needs each day. The military simulation and autonomous agents research communities have also done work in this area. PMFserv implements character physiology through a series of reservoirs that empty at various rates and are replenished through actions that fulfill each need (Silverman et al. 2006). Generally, physiology changes a person’s priorities. As a level of need increases, so does the priority of actions that might meet the need.

3.3 SOCIOLOGY: SUBGROUPS

Often, crowds are not merely composed of individuals navigating from location to location in isolation. While there are certainly individuals in crowds, smaller groups of people may travel together or stop to chat with one another (Villamil et al. 2003). Consistently depicting these groups impacts both low and high levels of a crowd implementation. On the lower level, forces or rules need to be crafted to pull groups together. These groups may separate to better navigate obstacles in the environment, but should tend to re-form. An implementation should also allow for members to occasionally join and leave the groups. To create a more consistent and natural crowd simulation with subgroups, these subgroups also need to be represented in some form at a higher level. Groups do not tend to be composed of random strangers; they tend to contain family, friends, coworkers, or individuals with some sort of association or common purpose. Creating and storing these associations enable some logical explanation for the existence of groups, rationale for group formation and separation (e.g., dropping the kids off at school rather than at a random location), and apparent social consistency through time.

3.4 CULTURE, ROLES, AND STATUS

It is said that cultural information is a minimum prerequisite for human interaction, and in the absence of such information, communication becomes a trial and error process (Knapp and Hall 1992). Cultural differences can be extensive and do not only include spoken language. First, different cultures have different distances for interacting. In some cultures, standing close and directly in front of a person while speaking is considered either an intimate or a hostile act. In other cultures, not standing close and directly facing a person would be considered rude. There are also different touching behaviors, gestures, and eye gaze patterns (Knapp and Hall 1992).

It is also well-known that there are some similarities across cultures. Studies have shown that six “universal” facial expressions can be distinguished across cultures (Ekman and Friesen 1977). Also, some behaviors have cross-cultural similarities, e.g., coyness, flirting, embarrassment, open-handed greetings, and a lowered posture for showing submission (Knapp and Hall 1992). While culture is a very important component of human behavior and communication, it has been neglected as a focus for the crowd simulation research community, perhaps due to its interpersonal complexity.

Every character should have a role or roles that it is playing, whether it is a professor of astrophysics, a tour guide, or just a man walking down the street. Roles involve expectations, both from the individual playing the role and from those interacting with the individual playing the role. For a character to be consistent, it must meet the expectations of the role it is playing, including performing appropriate actions in appropriate contexts, whether alone or in a group.

Roles are learned, generalized guidelines for behavior. Among other things, a role can stem from an individual’s occupation, kinship, age, sex, prestige, wealth, or associational grouping. In a situation, one participant normally establishes his or her role, and the other participant(s) must either go along or counter with a different role definition. There must be an agreement on the roles to effectively interact (Burgoon et al. 1989).

Roles influence many of the channels of nonverbal communication. Take for example the roles of doctor and mechanic. We have certain expectations about these roles. The appearance of a doctor is expected to be clean and neat, while a mechanic may be very messy. We would also expect the interpersonal distance with a doctor to be smaller and the physical contacts more frequent (when comforting as well as examining). Confusion and alarm might result from a mechanic standing too close or touching too often (even if trying to comfort someone after showing them the bill).

In any interpersonal situation, one person’s status is always at least a little above or below the other person’s (Johnstone 1979), and age is often a component of status. Age and status are reflected in many different display channels. To present consistent character behavior, these channels should all indicate the same age and status.

For example, gestures change and become more subtle with age (Lewis 1998), and people of higher status seem to gesture less frequently (Lewis 1998). Interpersonal distance also changes with age. Distance seems to increase with age, but is always closer with peers than with those that are younger or older. Older people are more likely to touch younger people than vice versa (Burgoon et al. 1989), which is probably a factor of both status and age. People of more dominance are more likely to engage in unwavering, direct looks. People tend to lower their eyes to show deference to authority figures, and submission is often marked by raised eyebrows, which connote deference (Burgoon et al. 1989). Proper posture signals dominance. High-status people are more confident and therefore comfortable in their space.

The agents research community has, to some extent, modeled status. Hayes-Roth et al. (1996) have explored the use of status with embodied agents in the form of a master-slave relationship. They illustrate how the postures and actions of the characters change as the servant becomes the dominant character. Poggi and Pelachaud (2000) model status through facial expressions called *performatives*, which are facial expressions that accompany and add interpersonal relationship information to speech. Musse and Thalmann (1997) included dominance in their crowd simulations.

3.5 SUMMARY

Certainly, there are many factors that influence individual differences. We have presented just a few that might be viable for and have impact on a crowd simulation. Such factors affect the lower level crowd controls (see Table 4.1), but also the higher level decision-making components that we will discuss subsequently. For example, an individual's personality profile might affect their interpersonal space (lower level), but it is also at the heart of their priorities and hence action choices (higher level).

Not all of the numerous possible factors need to be implemented to create an effective population. Factors should be prioritized based on their possible impact on the purpose of the simulation. Ideas for individual differences that may have an impact on crowd simulations can be found in the nonverbal communication literature (Burgoon et al. 1989), as well as publications on autonomous agents and animation (Cassell et al. 2000; Allbeck and Badler 2001, 2002, 2003, 2004; Ashida et al. 2001; Badler et al. 2002).



CHAPTER 4

Framework (HiDAC + MACES + CAROSA)

We have developed a framework for high-density multiagent simulation with a bottom-up approach. On the low-level agents move within a room driven by a social forces model with psychological and geometric rules affecting several parameters that will allow for a wide variety of emergent and high-density behaviors. Above the motion level, we need a wayfinding algorithm that will perform navigation in large complex virtual buildings, using communication and roles to allow for different types of behavior and navigation abilities. Both the motion level and the wayfinding with communication and roles can be affected by psychological factors that are initially given as personality parameters for each agent, but also can be modified during the simulation to affect an agent's behavior. CAROSA (Crowds with Aleatoric, Reactive, Opportunistic, and Scheduled Actions) is the upper level that provides semantic information about the environment and agents, including agent roles. It enables functional crowds that perform actions that are appropriate to time and place. This includes both reactive and deliberative actions as well as opportunistic actions and actions that are statistically driven.

Agents move within complex virtual environments with several rooms, corridors, obstacles, stairwells, and doors that can be opened or closed at any time during the simulation (Figure 4.1). To navigate these virtual environments, route selection is carried out through an interactive high-level wayfinding algorithm that dynamically calculates the global path based on the agents' knowledge of the environment (Pelechano and Badler 2006).

To achieve real-time interactive navigation, some relevant information about the environment is precalculated and stored. Among the information stored are paths toward the exits, distances from each door to a destination point, and the position of the attractors that will be used during the local motion to steer the agents. Agents will have access to this information based on their roles. This allows us to represent different levels of knowledge about the environment, but any other information required by the agent will have to be gathered through exploration, learning, and communication with other agents.

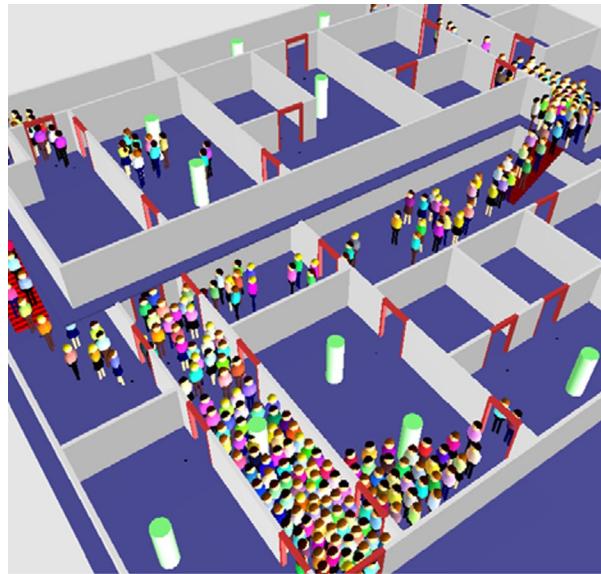


FIGURE 4.1: Example of a complex building (Pelechano 2006).

The navigation process is interactive, meaning that agents are endowed with a decision-making process that will allow them to follow the known route or make new decisions based on changes in the environment and their psychological parameters.

Changes in the environment include a door appearing locked, which makes that path invalid or creates a bottleneck in some part of the desired path. These changes make it more difficult to reach a goal, and therefore, based on the level of impatience assigned to the agent, a decision could be made to take a different route.

Each cell of the building stores the shortest path to each exit. There are two ways in which this information can be interpreted. On the one hand, we can consider that this shortest path stored in the cell corresponds to the path that an agent in that cell would have followed when entering the building and therefore is the only one known. On the other hand, we could consider this shortest path as being the one indicated by the exit signs in a building and therefore would be the ones that everyone would follow in case of emergency.

Agent spatial knowledge is represented by a graph where the nodes are the rooms and the arcs are the portals between rooms. This mental graph that represents the memory of the agent will have more nodes added as it navigates and explores the building. At any time, each agent needs to know which rooms of the building have been fully explored and which others still have portals that lead to rooms that have not been visited yet. The mental graph abstracts away the actual geometry

of the environment. The building geometry is used later to compute locomotion transit times and portal bottlenecks.

Another crucial source of information is communication with other agents. There are two pieces of information shared by the agents every time two or more agents meet in a room: locations of hazards found in the building that are blocking some of the paths and parts of the building that have been fully explored by other agents and found to have no exit through them. This localized sharing of mental models is the key to our algorithm's wayfinding behavior.

Each individual within the crowd will have different behaviors depending on two attributes: leadership and training:

Leaders and trained agents have complete knowledge about the internal building structure that would also help others during the evacuation process. An example of this type of agent would be a firefighter.

Leaders but untrained agents correspond to people that by nature can handle stress better, tend to help others, and will explore the building searching for new paths.

Nonleaders and untrained (followers) represent dependent people who might panic during an emergency situation and reach the point where they are incapable of making their own decisions.

4.1 INTERACTION BETWEEN FRAMEWORK LEVELS AND PSYCHOLOGICAL MODELS

Our crowd simulation model is a multiagent system without a centralized controller. Each agent has its own behavior based on roles and personality variables that represent physiological and psychological factors observed in real people. Agent behaviors are computed at three levels (Figure 4.2):

- CAROSA (high-level behavior): character definitions, object and action semantics, and action selection and control
- MACES (middle-level behavior): navigation, learning, communication between agents, and decision making for wayfinding
- HiDAC (low-level motion): perception and a set of reactive behaviors for collision avoidance, detection, and response in order to move within a room

The parameters describing each agent are stored in CAROSA along with the object and action semantics. Based on schedules, agent parameters, and percepts from HiDAC, CAROSA determines what actions an agent should perform and what objects are needed to participate in the action. It then passes the location where the action is to be performed to MACES so that course can be charted. If a path to the location is not possible, a failure is generated and passed back to CAROSA.

CAROSA passes the action information to HiDAC where ultimately the motion will be displayed. HiDAC returns percepts of the environment and other agents and when necessary failure states.

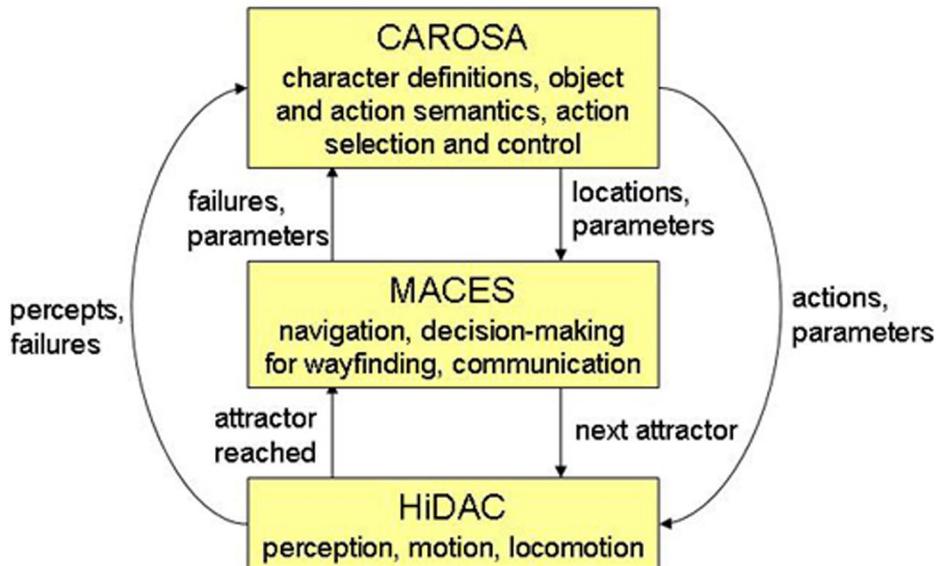


FIGURE 4.2: Framework overview.

CAROSA can also both get and set the agent parameters found in MACES and HiDAC. For example, CAROSA might determine the leaders in a simulation and set this parameter for the agents in MACES. In return, MACES might alert CAROSA when an agent becomes panicked, so that it no longer generates actions for the agent. More information about CAROSA can be found in Chapter 7.

Figure 4.3 shows details of the interaction between MACES and HiDAC. The higher module receives information about bottlenecks and door changes that have been perceived by the agent and makes decisions based on that information and its current knowledge of the environment. Once this level decides the next room to walk to, it sends the next attractor point to the low-level module to carry out the required motion to reach it. When the low-level module reaches the attractor, it queries the higher module for the next attractor in its path toward the destination.

The motion submodule queries the perception submodule about positions and angles of obstacles, crowd density ahead of the agent, and velocity of dynamic obstacles. Based on information perceived and the internal state of the agent (current behavior, panic, impatience, etc.), the motion submodule calculates the velocity and next position of the agent and sends a message to the locomotion submodule to execute the correct feet movements.

Both high- and low-level behaviors are affected by a module representing the psychological and physiological attributes of each agent. The idea of using a psychological model is that agents will operate independently in perceiving the simulated world and in forming their reactions to it. At no point

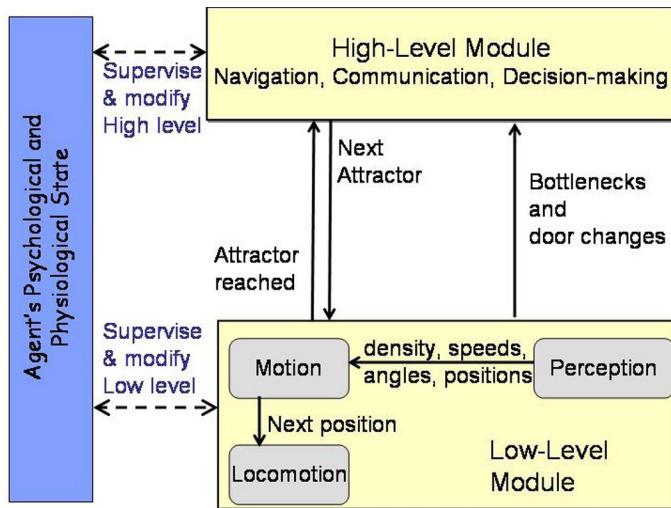


FIGURE 4.3: Architecture overview of HiDAC + MACES with psychological model (Pelechano et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

will they be pre-scribed or programmed via rules or procedures. We only model personality attributes, and individual agents will make their own decisions that lead to the emergent crowd behavior.

The high-level behavior is affected by changes in psychological elements such as panic or impatience, by altering the decision-making process (e.g., an impatient agent will select a different route after perceiving congestion at a door). Other elements such as an agent's memory and orientation abilities can be affected by a high-level behavior (psychological studies show that a person under panic may suffer disorientation). Finally, an agent's psychological state may trigger changes in roles (e.g., a leader changing to follower when its panic level gets very high or a trained agent exhibiting untrained behavior when suffering from disorientation).

The low-level behavior is also affected by changes in the psychological state of the agent, which will trigger modification of the agent's speed, probability to fall, pushing thresholds, etc. The psychological model needs to have as input information about environment events detected by the agent's perception system and information obtained through communication. Then this information will be combined with the agent's current emotional state to modify it if necessary and send back the right input to both low- and high-level modules.

In Figure 4.3, we can observe how the psychological model interacts with the navigation and local motion modules. This psychological module contains information regarding the agent's state and psychological factors that are currently affecting its behavior. This module needs to supervise both high and low levels to detect changes that should alter the internal state of the agent and then apply the corresponding modification at both decision-making and local motion levels.

4.2 PARAMETERS AFFECTING CROWD BEHAVIOR

Table 4.1 shows the parameters that can be input in our system to specify initial conditions for a simulation and psychological and physiological personality attributes for the agents. In the current framework, those parameters are specified by the user through an interface and can be modified during the simulation. It could also be possible to get those values through an API from a high-

TABLE 4.1: Parameters affecting behavior

PARAMETER	TYPE	PROPERTIES
Leadership	Percentage	Percentage of leaders in the crowd (the rest will be dependent individuals)
Trained	Percentage	Percentage of trained (building knowledgeable) individuals among the leaders
Communication	Boolean	Whether agents can communicate
Panic	Percentage	Percentage of people that will exhibit panic when an alarm goes off or a hazard is perceived
Panic propagation	Percentage	Percentage of people with high probability of exhibiting panic behavior when perceiving other agents in panic
Impatience	Percentage	Percentage of people that will avoid bottlenecks when other paths are available
Falling	Percentage	Percentage of people with high probability to lose equilibrium under severe pushing (representing physical abilities)
Pushing Threshold	Percentage of people with {min, medium, max}	Percentages for each distance allowed from other agents for which repulsion forces will not apply
Right preference	Percentage	Percentage of people that will tend to move towards the right when facing opposite flow
Avoidance	Percentage of people for {min, medium, max}	Percentage for each magnitude indicating how abruptly a person will try to avoid others by walking around instead of forming lines during normal conditions.

level psychological model that would drive the internal emotional state of the agents (Pelechano et al. 2005).

The current interface allows the user to create either the entire population at once and have each parameter being distributed among the entire crowd according to the percentage assigned or, if the user desires to have more control over the individual parameter of the agents, then smaller groups of agents can be created with specific personality attributes. For example, if the user wants a population of 40 agents, where 50% are leaders with maximum pushing threshold and the other 50% are dependent agents with minimum pushing threshold, then the user should first create a segment of 20 agents, with 100% leadership and 100% maximum pushing threshold. Next, add another segment of 20 agents with 0% leaders and 100% minimum pushing threshold.

Figure 4.4 shows the interface used to create the segments/population. As we can see, the interface allows the user to specify the size of the segment and the percentages of each parameter that will affect that group of agents. The user can create as many segments as desired.

Leadership. This specifies the percentage of agents in the crowd that tend to be leaders and take decisions in terms of global navigation when they find themselves blocked due to a hazard or a locked door. The rest of the individuals are considered dependent people, which means that in the situation of not knowing where to go, they would rather follow others than explore the environment by themselves.

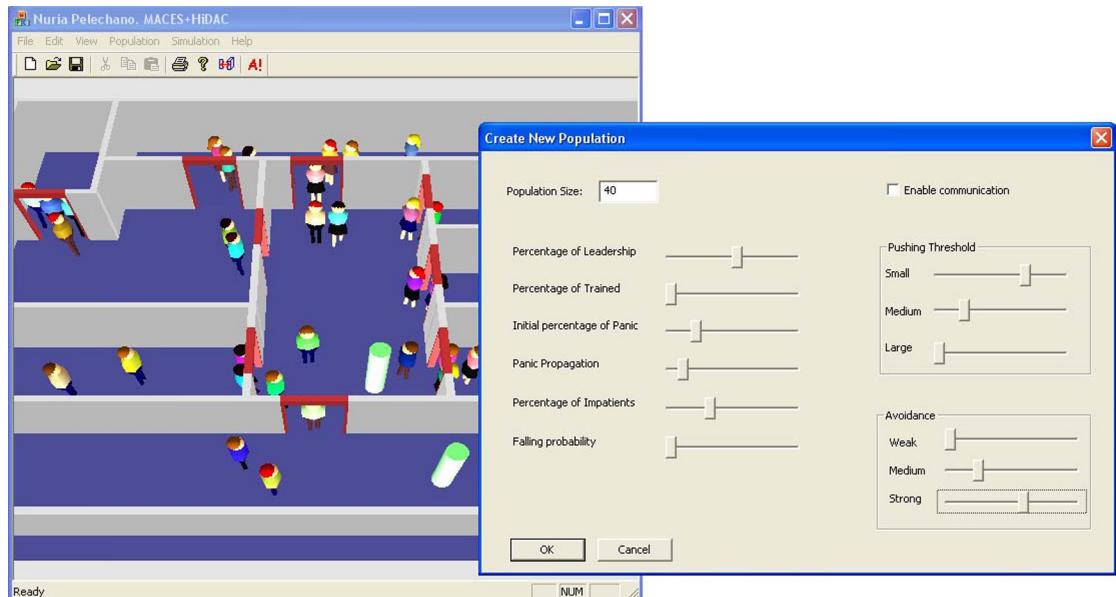


FIGURE 4.4: MACES + HiDAC interface (Pelechano 2006).

54 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

Trained. Among the leaders population, there will be a percentage that will have complete knowledge about the internal connectivity of the building, and therefore, if the shortest path being followed becomes invalid, they will immediately know an alternative solution. Basically, their internal mental map corresponds to the cell and portal graph representing the environment, while the rest of the agents will only have a subgraph of it at any given time, which will expand as they explore and communicate with other agents.

Communication. This can be set to true or false based on whether we want the agents to have the ability to communicate or not during the simulation. Communication is the process that allows agents to exchange relevant information about the environment, such as “there’s a fire in that room” or “the door on the left leads nowhere” and so forth.

Panic. Initial percentage of people who will tend to panic when an alarm goes off or when they see a hazard.

Panic Propagation. Percentage of people who although they will not start to exhibit panic behavior when the alarm goes off, may change to panic mode after seeing others panic for some amount of time or by having many individuals around them pushing for a certain period. This is a very interesting feature that allows our system to exhibit emergent panic propagation that will affect bottlenecks and flow rates through portals.

Impatience. Overall percentage of people who when observing a bottleneck in their next portal may reconsider their selection and interactively change their path if they know an alternative short route.

Falling. To represent the fact that some individuals are more likely to fall when they find themselves in a high-density crowd (elderly, disabled, weaker people, etc.), we allow the user to represent this factor by setting a percentage of people that are likely to have equilibrium problems. For example, in the case where the user needs to simulate an evacuation from a building with 80% of elderly individuals, this variable can be used to represent the likelihood of some of those people having difficulties in maintaining their equilibrium when being pushed by others.

Pushing thresholds. Pushing thresholds identify the distance that agents are willing to maintain to other agents of the crowd, i.e., the “contact distance” between individuals. It can be set to very small (0), average (1), or large (2). When an agent falls inside that distance, it will provoke a repulsion force that pushes it away from the other agent. Pushing behaviors can vary in a crowd, where some individuals are more likely to try to open their way through a high-density crowd even if it is at the cost of pushing others.

Right Preference. When people walk in a crowd, they tend to apply social rules that usually match driving rules. In many countries, people drive on the right, and therefore, when they are walking and another person is moving in the opposite direction, social rules will make each human

try to avoid the other by slightly diverting their paths toward their right-hand side. This parameter is used to set the percentage of people that will exhibit right preference.

Avoidance. Avoidance factor is linked to collision avoidance behavior. Collision avoidance deals with applying forces that alter the agent's trajectory to smoothly avoid static and dynamic obstacles. The avoidance factor gives the strength of those forces, which requires agents to do more or less abrupt direction changes. The result affects mainly the width of any line/queue that arises during normal conditions. We establish three values (weak, medium, and strong) and the percentage of each of them among the population.



CHAPTER 5

HiDAC: Local Motion

Local agent motion is based on a combination of geometric information and psychological rules with a forces model to enable a wide variety of behaviors resembling those of real people. HiDAC uses psychological attributes (panic, impatience) and geometric rules (distance, areas of influence, relative angles) to eliminate unrealistic artifacts and allow new behaviors:

- preventing agents from appearing to vibrate
- creating natural bidirectional flow rates
- queuing and other organized behavior
- pushing through a crowd
- agents falling and becoming obstacles
- propagating panic
- exhibiting impatience
- reacting in real time to changes in the environment

These emergent behaviors are driven by the parameters given in Table 3.1.

5.1 INTRODUCTION

In terms of defining the motion of each agent, we classified three main approaches: social forces systems, rule-based models, and cellular automata models. None of these models, however, can realistically animate high-density crowds. HiDAC focuses on the problem of simulating high-density crowds of autonomous agents moving in a natural manner in dynamically changing virtual environments. In this section, we will explain how psychological and geometric rules are layered on top of the basic social forces model to improve high-density crowd movement and add realism. Since applying the same rules with the same parameters to all agents leads to homogeneous behavior, agents are given different psychological (e.g., impatience, panic) and physiological (e.g., speed) traits that trigger heterogeneous behaviors based on crowd density and personality.

Each agent is endowed with perception and reaction to static and dynamic objects and agents within the current room. Common perception approaches in the literature are based on casting a set

58 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

of rays to calculate intersections. We introduce a simpler approach to perceive the environment and make decisions while still achieving highly realistic results.

Realistic movement is achieved both in terms of collision avoidance and collision response. Over longer distances, tangential forces gently steer the agent around obstacles, while over shorter distances, repulsion forces are applied to enable collision response. Pushing behavior is achieved by varying the long/short pushing threshold of each individual. Agents in a hurry (moving fast) and with small pushing thresholds will not respect others' personal space and will appear to push their way through the crowd. In contrast, agents with large pushing thresholds (more "polite") will respect lines and wait for others to move first.

Each agent scans an ellipse-shaped region in front of them. Relaxed agents temporarily stop when another agent moves into their path, while impatient agents do not respond to this feedback and tend to "push." Our model stops impatient agents from appearing to "vibrate" as they try to force their way through dense crowds, as we add temporal braking forces to the social force model. These forces only apply when repulsion forces fall within a specified range of angles opposing forward motion. The angles are set based on agent personality and crowd density.

5.2 AGENTS' SPEEDS AND DENSITIES

This section describes the quantitative factors that can be utilized to estimate the pedestrian movement accurately.

There is a large amount of data in the civil engineering and fire evacuation literature to calculate the movement component of total evacuation time. To simulate real pedestrians' movement, there are several moving parameters to consider such as

- speed: rate of travel along a corridor, ramp, and stairwells;
- flow: number of persons passing a particular segment of the egress system per unit of time;
- specific flow: flow per unit width of the egress component (persons/second per meter of doorway width).

TABLE 5.1: Velocity factors ([SFPE 2003](#))

EGRESS COMPONENT	K (m/s)
Corridor, doorway	1.40
Stair, riser = 190 mm	1.00
Stair, riser = 272 mm	1.08
Stair, riser = 165 mm	1.19

IMPAIRMENT	LEVEL WALKWAY (m/s)	STAIRWELLS: DOWN (m/s)	STAIRWELLS: UP (m/s)
Electric wheelchair	0.89		
Manual wheelchair	0.69		
Crutches	0.94	0.22	0.22
Walking stick	0.81	0.32	0.34
No disability	1.24	0.70	0.70

Most of this information on the movement of people, including disabled individuals, has been collected through fire drills, in stairs and corridors, and through doorways. To accurately simulate human behavior, we employed the data available in the 2002 Society of Fire Protection Engineers document, “The SFPE Engineering Guide to Human Behavior in Fire” ([SFPE 2003](#)).

Speed is a function of the density of the occupant flow, type of egress component, and mobility capabilities of the individual. Let pers/m^2 be the number of people per square meter. For a density greater than $0.55 \text{ pers}/\text{m}^2$,

$$v = k - akD, \quad (5.1)$$

TABLE 5.3: Fruin levels of service				
FRUIN LEVEL OF SERVICE	DENSITY (pers/m ²)	SPACE (m ² /pers)	FLOW RATE (pers/m/s)	AVERAGE SPEED (m/s)
A	<0.31	>3.22	<0.38	1.3
B	0.43–0.31	2.32–3.24	0.38–0.55	1.25
C	0.72–0.43	1.39–2.32	0.55–0.82	1.15
D	1.08–0.72	0.93–1.39	0.82–1.10	1.00
E	2.17–1.08	0.46–0.93	1.10–1.37	0.7
F	>2.17	<0.46	>1.37	

60 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

and for densities less than 0.55 pers/m², there are not enough people around an individual to impede its walking speed, therefore maximum walking velocities are defined by

$$v = 0.85 \cdot k, \quad (5.2)$$

where v is speed (m/s), a is constant (0.266 m²/pers), k is velocity factor, as described in Table 5.1, and D is density of occupant flow (pers/m²).

TABLE 5.4: Summary of walking speeds according to several studies as cited in Thompson and Marchant (1995)

	STUDY	WALKING SPEED (m/s)					
		OLD PEOPLE			WALKING SPEED		
		Slow	Normal	Fast	Slow	Normal	Fast
Men	Blanke and Hageman		1.38			1.32	
	Himann et al.		1.21	1.47			
Women	Finley et al.		0.7			0.84	
	Blanke and Hageman		0.32			1.59	
	Ferrandez et al.	0.82		1.08			
	Himann et al.	0.89		1.14			
	Leiper and Craik	0.96		1.15			
	Obrien et al.	0.74		0.97			
Both	Cunningham et al.	1.05	1.33	1.6	1.08	1.39	1.72
	Elble et al.		0.94	1.39		1.18	1.67
	Waters et al.	0.81	1.22	1.5	0.71	1.32	1.76
Judge et al.			1.06	1.43			

At lower densities, individuals can move freely in the environment, being able to reach their desired maximum speed, and at higher densities, velocity will be reduced. Mean velocities for impaired individuals and people without disabilities given by [Shields et al. \(1996\)](#) are presented in Table 5.2.

It is also important to stress the importance of density when simulating high-density crowds. Real measurements show that crowds can maintain fluid movement even at densities of 7 pers/m² ([Berrow et al. 2005](#)) or in some extreme situations, real densities observed have reached up to 13.5 pers/m² as reported by [Tsuiji \(2003\)](#). Reduction in walking speed is already noticeable for densities above 2 pers/m² and congestion appears for densities of 4 pers/m².

[Fruin \(1971\)](#) introduced a concept of level of service where flow rate is expressed as a function of density. According to Fruin, a pedestrian area occupied ranges from about 0.5 to 2.3 m²/pers on walkways and from 0.4 to 0.9 m²/pers on stairways. The corresponding flows range from 0.38 to 1.37, as indicated in Table 5.3.

In our system, agents are given an initial maximum speed following a normal distribution with mean = 1.24 (standard deviation = 0.2) for rooms and mean = 0.7 (standard deviation = 0.2) for stairwells. The maximum speed can be increased though when an agent is in panic, where a running speed will apply (normal distribution with mean = 1.7 m/s and standard deviation = 0.2) (Table 5.4). As the density increases, individuals will reduce their speed because of interaction and repulsion forces with other agents and obstacles in the environment. Individuals can also fall as a

TABLE 5.5: Maximum and ultimate flow rates as cited in [Thompson and Marchant \(1995\)](#)

SOURCE	MAXIMUM DESIGN FLOW (pers/m/s)	ULTIMATE FLOW CAPACITY (pers/m/s)
Fruin	1.37	
Predtechenskii and Milinskii		1.83
Daly	1.43	
SPFE handbook	1.3	
Handkin & Wright	1.48	1.92
Polus et al.	1.25	1.58
Ando et al.		1.8

62 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

consequence of the pushing behavior in high-density crowds and later stand up again to continue with their movement when the area around them clears. Fruin's levels of service have been used as a reference to calibrate our system in order to achieve realistic flow rates through doors (up to 1.9 pers/m/s; Table 5.5) and realistic densities, although higher densities than the ones described by Fruin have been allowed for panic situations as indicated in the literature (TRB 1994; Berrow et al. 2005).

5.2.1 Walking Speeds and Densities When Walking Downstairs

Figure 5.1 shows the relationship between speed and density, and Figure 5.2 shows the relationship between flow and densities both for downstairs movement. This information has been used to calibrate the agents' movement when walking downstairs.

5.3 PERCEPTION

Autonomous agents need to perceive the environment to avoid static and dynamic obstacles while walking between two attractors. HiDAC provides efficient perception by using the cell and portal graph. As the agents walk around the environment, the lists of dynamic objects within each room are rapidly updated. Therefore, an agent can obtain the necessary data by queries to the cell.

For each obstacle, we need to calculate its distance and, if it is close enough to the agent, the angle between the agents' desired direction and the line joining the center of the agent and the

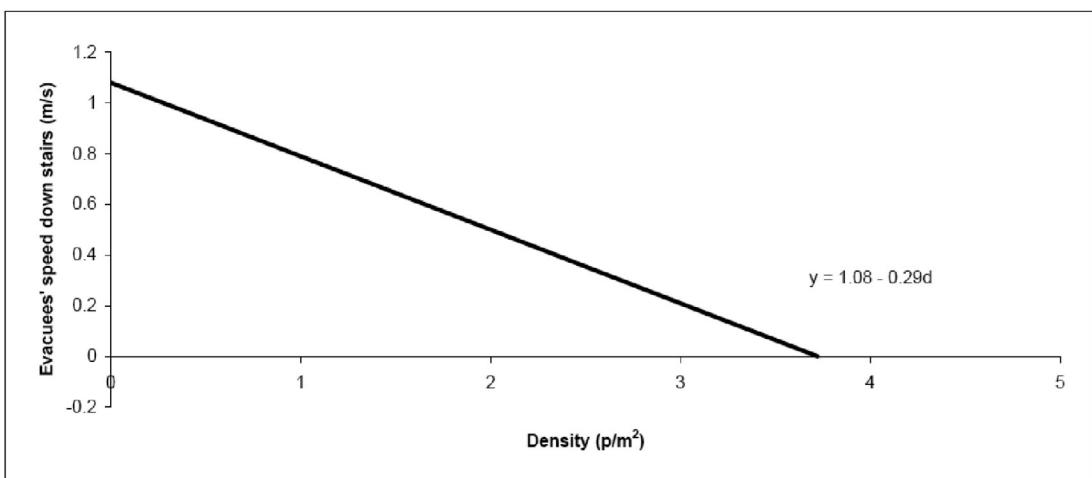


FIGURE 5.1: Relationship between speed and density when walking downstairs during evacuation (SFPE 2003) Reproduced with permission from the SFPE Engineering Guide to Human Behavior in Fire. Copyright 2003, Society of Fire Protection Engineers.

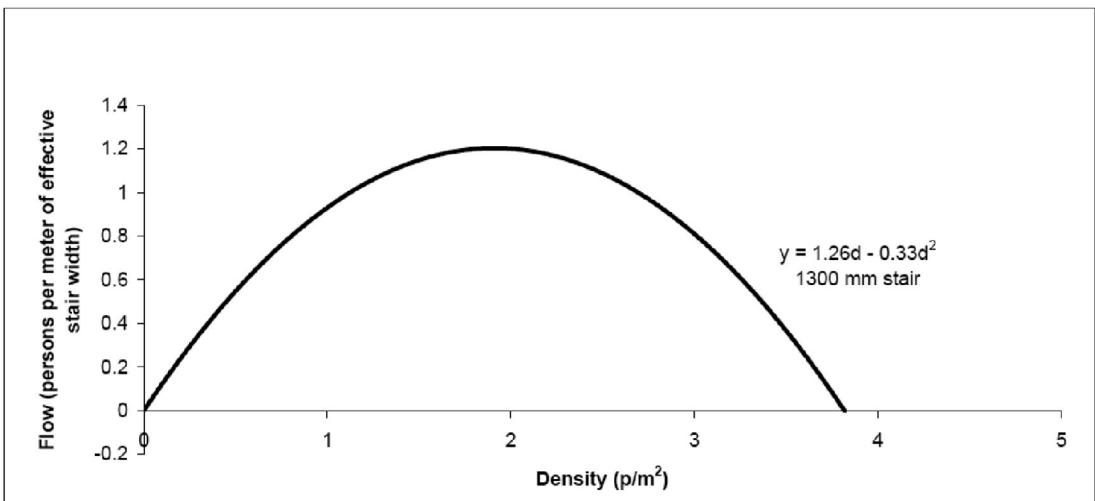


FIGURE 5.2: Relationship between flow and density walking downstairs (SFPE 2003) Reproduced with permission from the SFPE Engineering Guide to Human Behavior in Fire. Copyright 2003, Society of Fire Protection Engineers.

obstacle. The distance and the angle provide enough information to establish how relevant that obstacle is to the trajectory. While the agent looks for possible obstacles, it also updates its perceived density of the crowd ahead, which will be important in the decision-making process.

Humans can perceive a binocular field of view (FOV) from 120° to 180° , the latter being the most common. We can simulate human perception by having the virtual agents only be aware of those objects falling within a specified angle from their direction of movement (assuming the head is oriented in the same direction). Currently, our system is set to detect objects falling in a FOV of 180° (90° right and left of the direction of movement). This is calculated from the dot product between the direction of movement vector and the vector joining the current position with each object in the room. Since the dot product gives us the cosine of the angle between the two vectors, if that value is bigger than 0, it means the object falls within the agent's FOV.

Objects within the FOV are perceived but only objects falling within a rectangle area ahead are relevant in terms of obstacle avoidance. Figure 5.3 shows an agent (*A*) perceiving several obstacles simultaneously. In reality, we do not give obstacles avoidance preferences based on distance, but on how much they affect the desired trajectory. In Figure 5.3, we can observe that although the wall and the column are closer to the agent, our algorithm also factors in angles, which makes the agent (*B*) ahead the most important obstacle at this moment.

Since our algorithm only needs distances and angles, it is faster than casting rays for intersection with every obstacle, since our method has cost $O(N)$, where N is the number of obstacles in the room, while ray casting has cost $O(R \cdot N)$, where R is the number of rays cast and N the number of

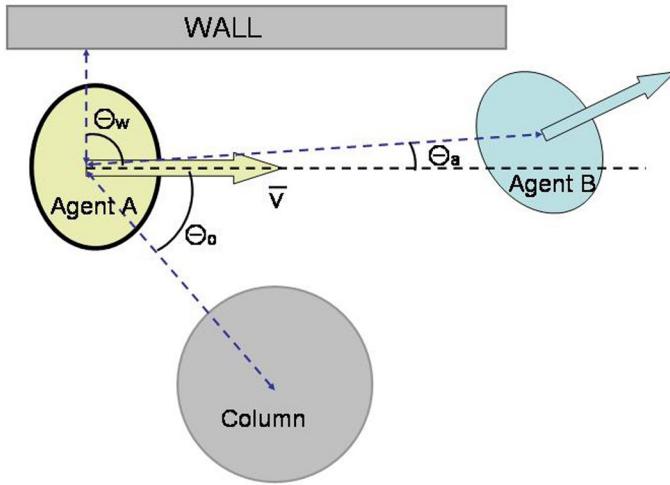


FIGURE 5.3: Perception for the yellow agent (Pelechano et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

obstacles. The visual results achieved for our crowd simulation prove our method to be sufficient for an agent’s environment perception used to make decisions regarding its motion.

5.4 CROSSING PORTALS

In our model, the desired velocity direction within each room is given by an attractor point that is located close to the next portal the agent needs to cross (Figure 5.4). This behavior orients the agent so that its velocity is radially aligned toward the target (attraction point). In the absence of obstacles or other agents, every agent will flow along the evacuation direction field (passing through the portals unobstructed). Floor is treated as a continuum, not as a discrete regular grid.

Collision detection is performed only with the people within the same room, except when people are crossing a portal. In this situation, care must be taken to avoid intersection between agents leaving and agents entering the room. Our approach to this problem consists of keeping track of the people currently crossing a portal. When an agent is near a door, collision detection is performed not only with the other agents in the room but also with those currently crossing the doorway (geometrically located close to the attractors at both sides of the door).

Figure 5.5 shows the different states in which an agent can appear while crossing a door and the transition between states. To walk from cell N to cell $N + 1$, an agent will have A as its first attractor point. When the agent’s position is within half a meter from A , then the high-level algorithm will set B as the next attractor. In this state, the agent will be inserted in the list of current

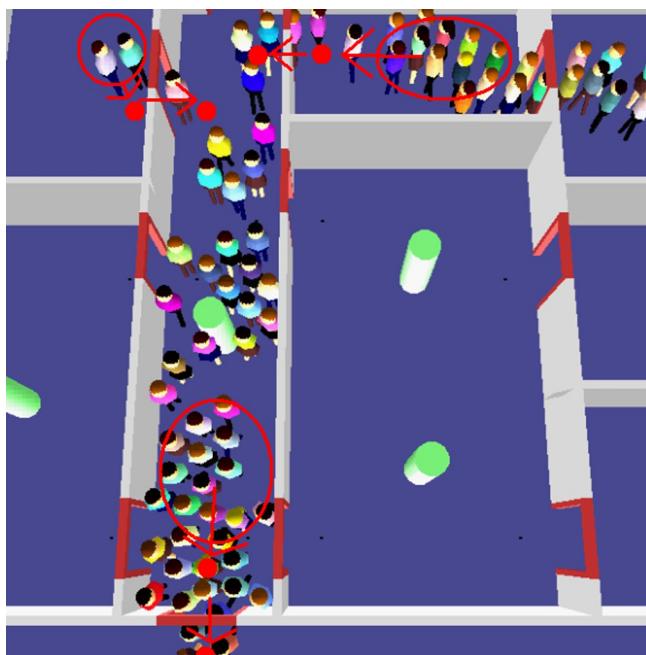


FIGURE 5.4: Agents steered by attraction points.

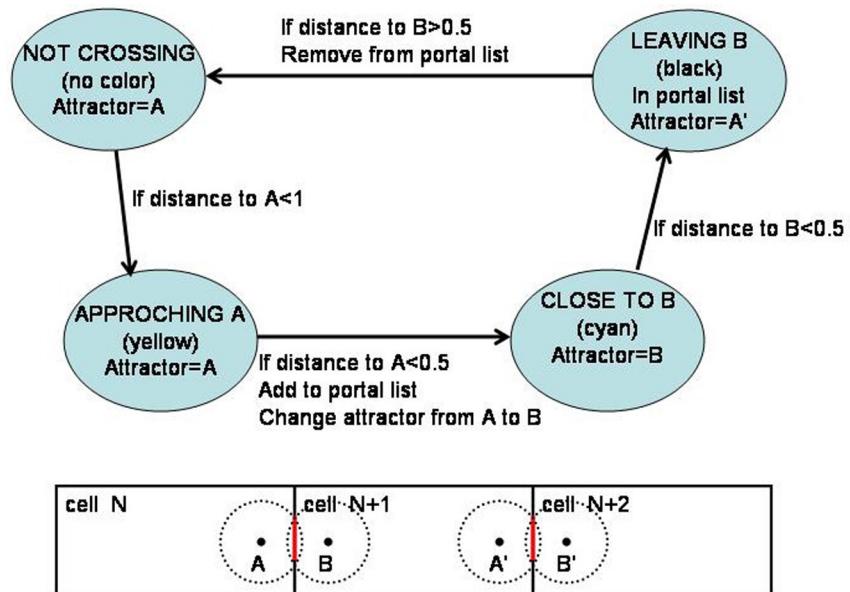


FIGURE 5.5: Crossing portals states.

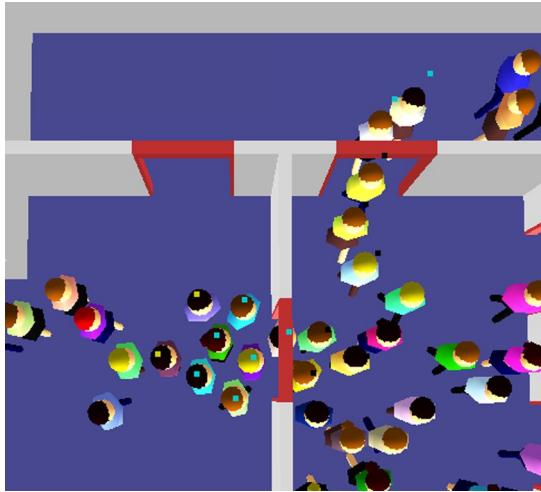


FIGURE 5.6: Agents crossing a portal.

agents crossing the portal. When the agent gets close to attractor B , the high-level algorithm will decide the next door based on the agent's knowledge about the environment and the desired goal. In the figure, the next attractor will be A' . The agent will stay in the list of agents currently crossing until it moves half a meter away from B , and therefore, there will be no risk of intersection with agents crossing the portal.

In Figure 5.5, we can appreciate how the agents' state changes as they walk through the door. The colored dots above each agent represent the state as indicated in Figure 5.6.

5.5 THE HiDAC MODEL

HiDAC is a parameterized social forces model that depends on psychological and geometric rules. Its high-level module determines which attractor point (waypoint or portal) an agent walks to within a room (Pelechano and Badler 2006). Collision avoidance, detection, and response are performed only with the people in the same room and with static elements of that room (walls and obstacles). When people are crossing portals, care must be taken to avoid intersection between agents leaving and agents entering. HiDAC keeps track of the people currently crossing a portal, so that when an agent is near a door, collision detection is performed against agents in the room and agents crossing the doorway.

Collision detection and response must be performed with those agents that are overlapping the agent from any direction. In contrast, collision avoidance is only performed against individuals that appear in the desired direction of movement and therefore are relevant to an agent's future position.

The movement of agent i (\mathbf{F}_i^{To}) depends on the desired attractor (\mathbf{F}_i^{At}), while avoiding walls w (\mathbf{F}_{wi}^{Wa}), obstacles k (\mathbf{F}_{ki}^{Ob}), and other agents j (\mathbf{F}_{ji}^{Ot}) and trying to keep its previous direction of movement to avoid abrupt changes in its trajectory ($\mathbf{F}_i^{To}[n-1]$). All these forces are summed together with different weights w_i , that are the result of psychological and/or geometric rules and determine the importance of each force on the final desired direction of movement:

$$\mathbf{F}_i^{To}[n] = \mathbf{F}_i^{To}[n-1] + \mathbf{F}_i^{At}[n] w_i^{At} + \sum_w \mathbf{F}_{wi}^{Wa}[n] w_i^{Wa} + \sum_k \mathbf{F}_{ki}^{Ob}[n] w_i^{Ob} + \sum_{j(\neq i)} \mathbf{F}_{ji}^{Ot}[n] w_i^{Ot} \quad (5.3)$$

The force vector is therefore:

$$\mathbf{f}_i^{To} = \frac{\mathbf{F}_i^{To}}{|\mathbf{F}_i^{To}|} \quad (5.4)$$

Finally, the new desired position $\mathbf{p}_i[n+1]$ for agent i is calculated as:

$$\mathbf{p}_i[n+1] = \mathbf{p}_i[n] + \alpha_i[n] v_i[n] \left((1 - \beta_i[n]) \mathbf{f}_i^{To}[n] + \beta_i[n] \mathbf{F}_i^{Fa}[n] \right) T + \mathbf{r}_i[n], \quad (5.5)$$

where: $v_i[n]$ is the magnitude of the velocity in the simulation step n . The velocity at each time step is calculated as:

$$v_i[n] = \begin{cases} v_i[n] = v_i[n-1] + aT & \text{if } v_i[n] < v_i^{\max} \\ v_i^{\max} & \text{otherwise} \end{cases},$$

where a is a constant that represents the acceleration of the agent when it starts walking until it reaches v_i^{\max} : the agent's maximum walking velocity. It can be set to depend on agent capability (normal, handicapped) and modified dynamically if the agent enters panic mode or is injured. \mathbf{r}_i is the result of the repulsion forces that affect the agent when it overlaps with a wall, obstacle, or another agent; these will be introduced in Section 5.5.2. α represents whether the agent will move in this step in its desired direction of movement or instead be pushed by a repulsion force.

$$\alpha_i = \begin{cases} 0 & \text{if } |\mathbf{r}_i| > 0 \vee \text{StoppingRule} \vee \text{WaitingRule} \\ 1 & \text{otherwise} \end{cases}$$

The StoppingRule and WaitingRule are used to avoid shaking behavior and allow for line formation, respectively. These rules will be explained in Sections 5.5.3 and 5.5.4. β_i is used to give priority to avoiding fallen agents on the floor:

$$\beta_i = \begin{cases} 0.5 & \text{if distance to fallen agent} < 2 \text{ m} \\ 0 & \text{otherwise} \end{cases}$$

\mathbf{F}_i^{Fa} is the avoidance force to avoid fallen agents and will be explained in detail in Section 5.5.6. T is the increment in time between simulation steps.

5.5.1 Avoidance Forces

Autonomous agents need to perceive the environment to avoid static and dynamic obstacles while walking to a attractor. HiDAC provides efficient perception through a cell and portal graph. Each cell corresponds to a room, and contains information about all the static objects within it. As the agents traverse the environment, the lists of dynamic objects within each room are rapidly updated; thus, an agent can obtain obstacle data by querying the cell.

For each obstacle, wall, and agent, we need to calculate its distance to agent i and if it is close enough, then we calculate the angle between agent i 's desired direction and the line joining the center of agent i and the obstacle. This information is used to determine whether it falls within the rectangle of influence (Figure 5.7). The distance and the angle provide enough information to establish how relevant that obstacle is to the trajectory. As they navigate the environment, agents also update their perceived density of the crowd ahead, which will be necessary to their decision-making process.

Wall and Obstacle Avoidance. Avoidance forces are calculated only for relevant obstacles, walls, and agents: those falling within the rectangle of influence.

The avoidance force for obstacle k is:

$$\mathbf{F}_{ki}^{\text{Ob}} = \frac{(\mathbf{d}_{ki} \times \mathbf{v}_i) \times \mathbf{d}_{ki}}{|(\mathbf{d}_{ki} \times \mathbf{v}_i) \times \mathbf{d}_{ki}|} \quad (5.6)$$

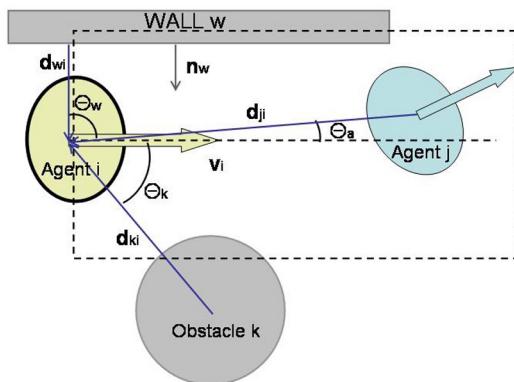


FIGURE 5.7: Perception for the yellow agent i (Pelechano et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

The avoidance force for wall w is:

$$\mathbf{F}_{wi}^{Wa} = \frac{(\mathbf{n}_w \times \mathbf{v}_i) \times \mathbf{n}_w}{|(\mathbf{n}_w \times \mathbf{v}_i) \times \mathbf{n}_w|} \quad (5.7)$$

Other Agent Avoidance: Overtaking and Bidirectional Flow. To exhibit realistic counterflows and overtaking behaviors, we include rules that modify some parameters of the forces model. This approach allows us to simulate human behavior by setting parameters related to real human movement. The parameters that affect the tangential forces for obstacle avoidance are:

- distance to obstacles
- direction of other agents relative to agent i 's desired velocity vector (\mathbf{v}_i).
- density of the crowd

If an agent appears in the rectangle of influence, then tangential forces (described below) will be applied to slightly modify the direction of movement and make a curve in the trajectory to avoid collision.

The angle between two agents' velocity vectors determines whether their movements are confluent or opposed. This angle is also used to simulate human decision making of how to react to an imminent collision. For example, if we are walking on the left side of a corridor and another person walks toward us on our right, none of us would change direction, but if we are both walking in the middle of the corridor, the majority of people have a tendency to move toward their right side. Therefore, when the velocity vectors are almost collinear, the tangential forces will point to the right.

Suppose an agent i detects agent j and agent l as possible obstacles (Figure 5.8). We calculate the distance vector toward agent i for each of them (\mathbf{d}_{ji} and \mathbf{d}_{li}). Agent j is farther away than l , but since it is moving against agent i , the perception algorithm establishes this obstacle as having higher priority. We select an agent to be avoided if it falls within the influence rectangle, unless that agent is walking in the opposite direction and with distance smaller than $D_i - 1.5$, where D_i is the length of the rectangle.

The tangential force (\mathbf{t}_j) that will steer agent i to avoid j is:

$$\mathbf{t}_j = \frac{(\mathbf{d}_{ji} \times \mathbf{v}_i) \times \mathbf{d}_{ji}}{|(\mathbf{d}_{ji} \times \mathbf{v}_i) \times \mathbf{d}_{ji}|} \quad (5.8)$$

Next, the normalized tangential vector is multiplied by two scalar weights to obtain the final avoidance force

$$\mathbf{F}_{ji}^{Ot} = \mathbf{t}_j w_i^d w_i^o \quad (5.9)$$

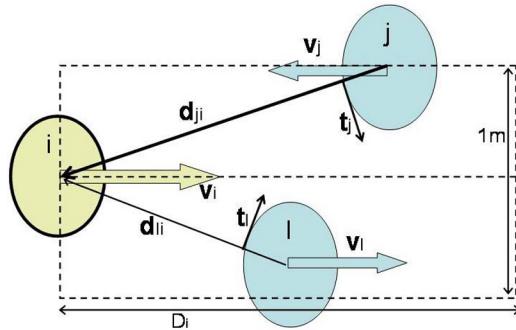


FIGURE 5.8: Collision avoidance rectangle of influence for agent i (Pelechano et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

where w_i^d is the weight due to the distance between agents and increases as the distance between the two agents becomes smaller. Thus the agent i trajectory will change more abruptly as the distance to agent j decreases:

$$w_i^d = (d_{ji} - D_i)^2 \quad (5.10)$$

and w_i^o is the weight due to the difference in orientation of the velocity vectors. It distinguishes whether the perceived agent is moving in the same direction as agent i or against it, and thus the magnitude will be higher to avoid counter-flow.

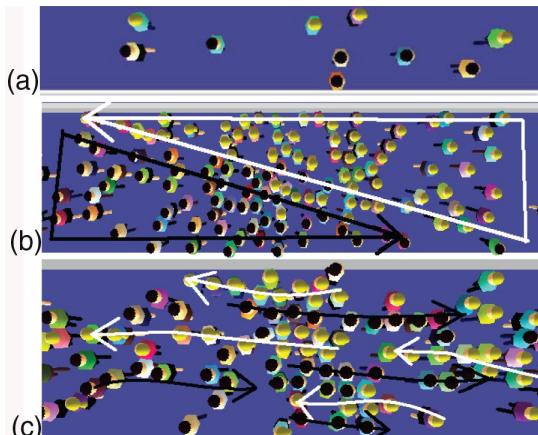


FIGURE 5.9: Bidirectional flows. People with blonde hair walk toward the left, while dark-haired people walk toward the right. (a) Low-density flows, (b) high-density without altering the viewing rectangle and right preference, (c) high-density with HiDAC (Pelechano et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

$$w_i^o = \begin{cases} 1.2 & \text{if } (\mathbf{v}_i \cdot \mathbf{v}_j) > 0 \\ 2.4 & \text{otherwise} \end{cases} \quad (5.11)$$

The last parameter to consider is the crowd density, which each agent perceives at any given time. If the crowd is very dispersed, then people look for avoidance from far away and keep their preference for the right-hand side of the space ($D_r = 3$ m), but when the crowd is very dense, then the right preference is not so obvious and several bidirectional flows can emerge ($D_r = 1.5$ m). Modifying the length of the collision avoidance rectangle and reducing the angle for right preference based on perceived density achieves this behavior.

Figure 5.9 shows different bidirectional flow rate formation for low and high densities. Figure 5.9b shows the result if the length of viewing rectangle and right preference parameters are not affected by density. The emergent behavior shows an unrealistic “triangle” of people moving in opposite directions, and awhile later in the simulation, two perfectly formed groups of people appear to move in opposite directions, which is less common in real high-density crowds.

HiDAC produces an interesting emergent counterflow behavior for high-density crowds (Figure 5.9c): the formation of lanes of people moving in the same direction intermingled among lanes moving in the opposite direction. This is a behavior that is often observed in real crowds, and it emerges here although it is not explicitly implemented.

Overtaking behavior emerges when a faster agent is walking behind a slower agent and there is no immediate oncoming traffic. Figure 5.10 gives an example of this behavior. Agent *A* (marked

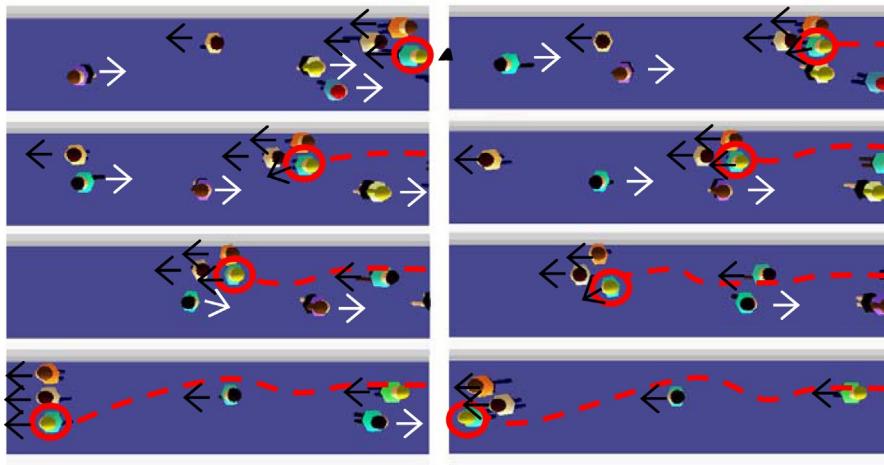


FIGURE 5.10: Example of overtaking animation.

with a red circle) is moving faster than the agent right in front of him. For the first five screenshots of the animation, since there is oncoming traffic within A 's rectangle of influence, those oncoming agents have avoidance preference over the ones walking in the same direction. After all the oncoming traffic has walked by, the slow agent right in front of A will get preference when setting the avoidance forces and consequently A will initiate the overtaking maneuver.

5.5.2 Repulsion Forces

When an agent's position overlaps with any static or dynamic obstacle, wall, or agent, then a collision response force applies. The repulsion force \mathbf{r}_i from Equation (5.5) is calculated as:

$$\mathbf{r}_i[n] = \sum_w \mathbf{F}_{wi}^{R,Wa}[n] + \sum_k \mathbf{F}_{ki}^{R,Ob}[n] + \lambda \sum_{j(\neq i)} \mathbf{F}_{ji}^{R,Ot}[n], \quad (5.12)$$

where $\mathbf{F}_{wi}^{R,Wa}$ is the repulsion force from wall w , $\mathbf{F}_{ki}^{R,Ob}$ is the repulsion force from obstacle k , and $\mathbf{F}_{ji}^{R,Ot}$ is the repulsion force from another agent j :

$$\mathbf{F}_{wi}^{R,Wa}[n] = \frac{\mathbf{n}_w(r_i + \varepsilon_i - d_{wi}[n])}{d_{wi}[n]} \quad (5.13)$$

$$\mathbf{F}_{ki}^{R,Ob}[n] = \frac{(\mathbf{p}_i[n] - \mathbf{p}_k[n])(r_i + \varepsilon_i + r_k - d_{ki}[n])}{d_{ki}[n]} \quad (5.14)$$

$$\mathbf{F}_{ji}^{R,Ot}[n] = \frac{(\mathbf{p}_i[n] - \mathbf{p}_j[n])(r_i + \varepsilon_i + r_j - d_{ji}[n])}{d_{ji}[n]}, \quad (5.15)$$

where p_i is the position of agent i , p_j is the position of agent j , and p_k is the position of obstacle k . Radii r_k , r_i , and r_j belong to obstacle k and agents i and j , respectively. Similarly, d_{ji} and d_{ki} are the distances between the centers of agent i and j and the centers of agent i and obstacle k ; d_{wi} is the shortest distance from the center of agent i to the wall w .

The parameter λ in Equation (5.12) is used to set priorities between agents (that can be pushed) and walls or obstacles (that cannot be pushed). If there is repulsion from walls or obstacles, then λ is set to 0.3 to give preference to avoiding intersection with walls or obstacles over agents that can be pushed away.

Finally, ε_i and ε_j are small personal space thresholds that the agents have and are used for the purpose of assigning different pushing abilities based on personality (discussed in Section 5.3).

5.5.3 Solution to “Shaking” Problem in High Densities

When an agent encounters a bottleneck in a high-density crowd, applying a basic forces model leads to an unnatural behavior where agents appear to vibrate continuously. This behavior must be avoided. (We have verified that this phenomenon is not based on our physics simulation implementation or its step size.) In HiDAC, we incorporate “stopping rules.” These rules are applied based on the personality of the agent, direction of movement of other agents, and current situation (panic or normal).

When repulsion forces from other agents appear against the agent’s desired direction of movement, and the agent is not in panic state, then the stopping rule applies:

$$\text{If } ((\mathbf{v}_j \cdot \mathbf{F}_i^{R,\text{Or}}[n]) < 0) \wedge (\neg \text{panic}), \text{ then } \text{StoppingRule} = \text{TRUE}.$$

To avoid deadlocks, a timer is set to a random value within a small range, and when the timer reaches 0, the agent will set *StoppingRule* = FALSE, so that in the next simulation step, the agent will try to move again.

When *StoppingRule* is TRUE, the parameter α_i in Equation (5.53) is set to 0, which implies that the agent will only change position if it is pushed by other agents; otherwise, it will inhibit the intention to move for several simulation steps. This effect drastically reduces the shaking behavior observed in the social forces model without increasing the computational time of the algorithm.

Only forces directed backward are relevant (Figure 5.11). If the forces appear to be toward our desired movement, we cannot decrease their intensity by not moving forward, and therefore no reaction is necessary.

This method succeeds in reducing shaking behavior, while still allowing body contact and thus pushing behavior. Since stopping rules do not apply when the agent is being pushed forward, this achieves the desired emergent result of people appearing to be pushed through doorways when there is a high-density crowd behind them.

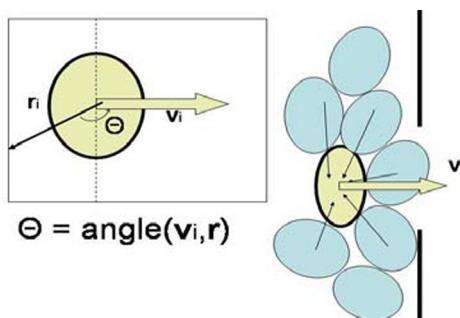


FIGURE 5.11: Example of repulsion forces that are necessary to apply braking forces (Pelechano et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

5.5.4 Organized Behavior: Queuing

In a “normal” (non-panic) situation, people will respect lines and wait for others to walk first. Such organized behavior emerges by adding influence disks ahead of each agent that drive the temporal waiting behavior; they work similar to the *stopping rules*. Figure 5.12 shows the area that triggers waiting behaviors in a non-panicked agent i in a high-density crowd when another agent j , walking in the same direction, falls within the disk: agent i sets $\text{WaitingRule} = \text{TRUE}$ and a timer starts. Agent i moves again when its area of influence does not satisfy the conditions for waiting, or when the timer reaches the value 0 to avoid deadlocks. The radius of the influence disk depends on personality (different people tend to respect different distances) and type of behavior desired, e.g., panicking agents will not respect these distances.

For simulations of “normal” situations (e.g., individuals leaving a cinema after a movie), all the agents exhibit waiting behavior when there is no available space ahead of them. The emergent behavior observed corresponds to queuing. Since agents use tangential forces to move within a crowd while avoiding others, the strength of those tangential forces will lead to narrow or wide queues, as can be observed in Figure 5.13. The user can specify those tangential forces to be minimum, medium, or maximum.

5.5.5 Pushing Behavior

Pushing behavior emerges because HiDAC can handle not only collision avoidance but also collision detection and response. Agents have different behaviors that can be triggered at any time. During an organized situation, individuals wait for space available before moving, but when in panic, they try to move until they collide with other individuals who impede forward progress. By combining both behaviors simultaneously for a heterogeneous crowd, we observe an emergent behavior where some individuals that do not respect personal space will get very close to other agents and push them away to open a path through a dense crowd. The effect of being pushed away is achieved by applying collision response forces and different personal space thresholds [ε_i and ε_j from the repulsion equations (5.13)–(5.15)].

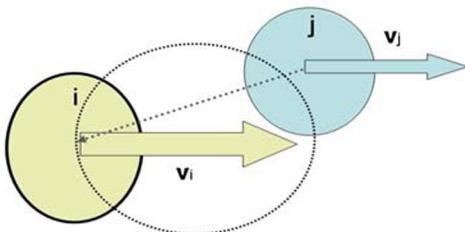


FIGURE 5.12: Area of influence for waiting behaviors (Pelechano et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

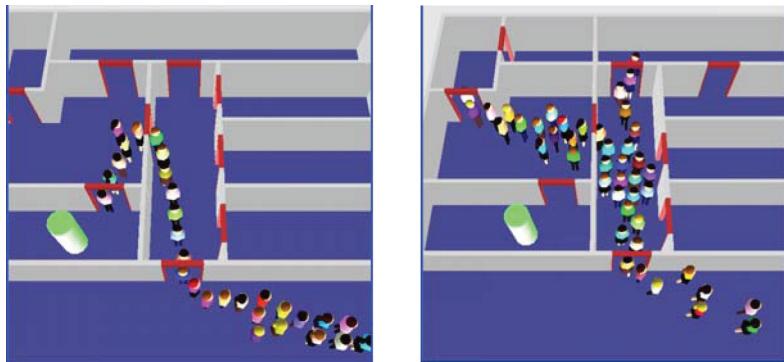


FIGURE 5.13: Examples of thin and wide queues emerging when animating a “normal” scenario.

An agent suffers a repulsion force from another agent when its personal space is overlapped. Figure 5.14 shows a sequence of simulation steps, where a smaller personal space threshold ε_i allows agent i to get closer to agent j who has a larger personal space threshold ε_j . Thus agent i can push away agent j while agent i is not being pushed and can continue with its desired trajectory.

Figure 5.15 shows an example where the top left room has been filled with panicked people (represented by redheads) who will tend to push others away, while the other three rooms contain individuals following more organized behaviors. After a few seconds of simulation, the redhead people have managed to almost empty their room by pushing others away in the corridor to reach the exit faster. Individuals in the other rooms are calmly waiting for their turn to get through the door.

5.5.6 Falling and Becoming Obstacles

A benefit to a physical social force model is that one might use it to gauge potential injury arising from high-density situations. When the majority of pushing forces affecting one individual are

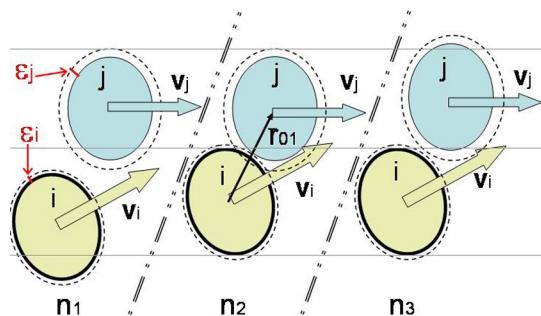


FIGURE 5.14: Pushing forces (Pelechano et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

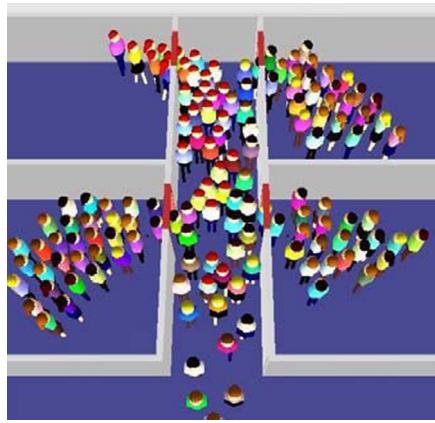


FIGURE 5.15: Redheaded people (starting in upper left room) exhibit panic behavior and push others to open their way through the crowd (Pelechano et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

approximately in the same direction, the agent will receive a sum of forces with magnitude high enough to make it lose equilibrium. At this moment, the person may fall and become an obstacle for the rest of the crowd.

Fallen agents represent a different type of obstacle because, unlike walls and columns, a body on the floor is an obstacle that should be avoided, but if necessary (or unavoidable) can be stepped over. In HiDAC, fallen individuals become a rectangular obstacle (a bounding box covering the torso and head, but not the legs since other individuals can easily step over that part of the agent). When other agents approach this new obstacle, weak tangential forces are applied to walk around the fallen agent [in Equation (5.5)], but repulsive forces are not applied. Therefore, when the crowd

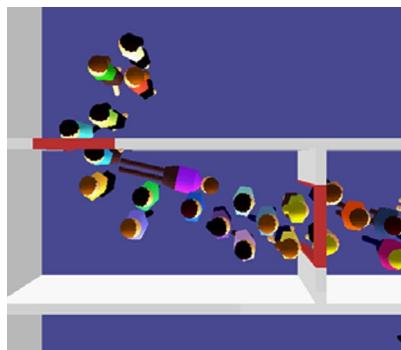


FIGURE 5.16: Agents avoiding a fallen agent (Pelechano et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

is extremely dense and the pushing forces from behind are strong, the result is that agents may walk over the body on the floor, as has been observed in actual extreme situations. Figure 5.16 shows an example of this behavior (where the crowd density is artificially low for visibility).

There are two possible types of avoidance forces. The first one corresponds to the case where the agent is within a neighboring area of the rectangle bounding the fallen agent, and it is calculated as tangential to the closest side of the rectangle. The second type corresponds to the case where the agent is already stepping within the rectangle. In this case, avoidance forces are a combination of tangential forces and the normal of the side that is closer to the agent. These two types of avoidance forces can be observed in Figure 5.17, respectively. In the next two animation examples, we can observe the avoidance behavior of the crowd depending on the density.

In the first example (Figure 5.18), the crowd is relatively dispersed so the avoidance forces that the fallen agent exerts on the other agents are enough for them to modify their trajectory and walk around it. In the figure, we have tracked the trajectory of one of the agents (marked with a red circle) to see the final path followed around the fallen agent.

In the second example (Figure 5.19), we can observe a high-density crowd, where even though the agents try to walk around the fallen individual, we see how some of them get pushed and cannot avoid stepping on or walking over the fallen individual. We tracked the path of one of the agents (marked with a red circle) to show how he gets pushed toward the fallen agent although he is trying to move around it.

5.5.7 Panic Propagation

HiDAC can simulate an emergency evacuation. When an alarm goes off, some agents will start in the panic mode. While in panic, they tend to move faster, push, and exhibit agitated behavior. All these behaviors depend on the agent personality and levels of panic. As the agents start running, they may provoke panic in other agents whose behavior will be modified in turn. To propagate panic, we use either communication between agents (managed by the high-level behavior module)

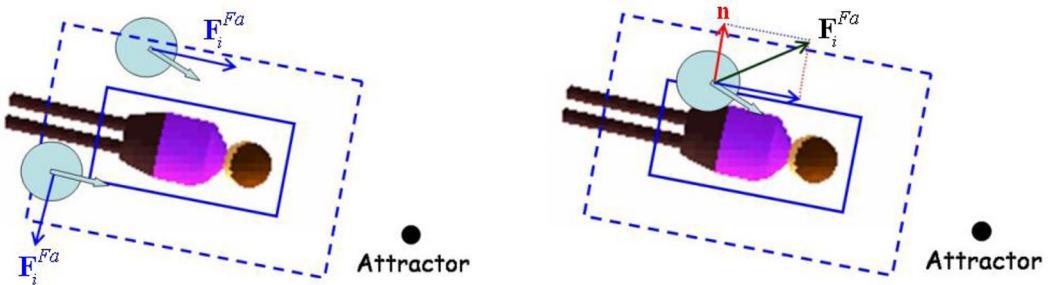


FIGURE 5.17: Avoidance forces around fallen agents.



FIGURE 5.18: Agents avoiding fallen individuals.

or perception to detect relevant changes in low-level behaviors, such as increasing crowd densities and number of people pushing or both.

As observed in the psychology literature, one of the most distinguishing features of people under panic is the fact that they will try to do everything faster and thus speed up. This behavior is easily exhibited by increasing the speed of those agents who tend to panic when an alarm goes off. In our system, panic will not only change an agent's speed, but will also affect some low-level behaviors such as canceling out the waiting behavior and activating instead the pushing behavior while reducing personal space thresholds. Smaller personal distance thresholds on an agent yield stronger repulsion forces on the agents surrounding it. Therefore, the agent exhibiting panic behavior can push others in order to open a path through the crowd.

An interesting effect to simulate is how panic can be spread among a crowd. Some individuals who will normally not tend to panic during an emergency may exhibit more agitated behavior if they find themselves within a panicking crowd with people pushing and creating a claustrophobic effect by leaving little or no space to move in. This effect is also modeled in HiDAC. When an alarm goes off, some agents will start in the panic behavior mode. In this mode, they tend to push

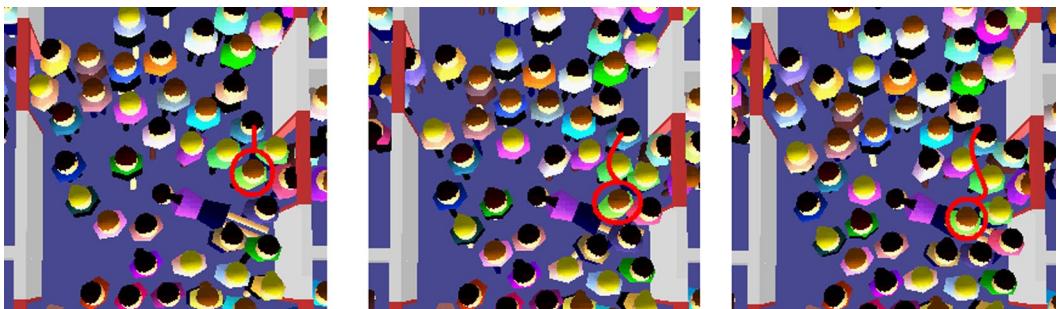


FIGURE 5.19: Agents walking over fallen individuals.

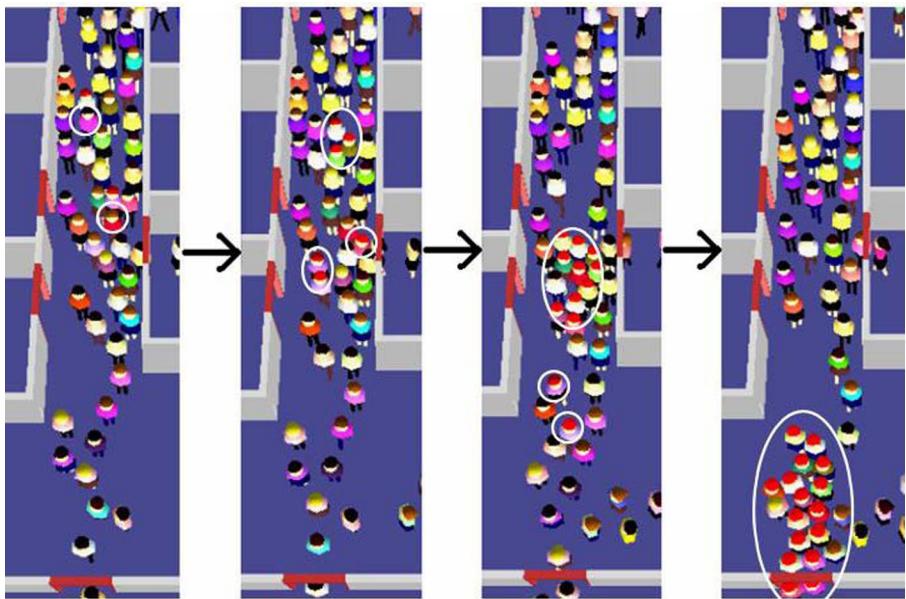


FIGURE 5.20: Panic propagation sequence.

and exhibit agitated behavior, and some have the ability to avoid bottlenecks. All these behaviors depend on the agent personality and levels of panic. As the agents start running, they may provoke panic in other agents who will in turn modify their behavior. To propagate panic, we use either communication between agents (managed by the high-level behavior module), or perception to detect relevant changes in the scenario such as increasing crowd densities and number of people pushing (both low-level behaviors).

In Figure 5.20, we can observe a sequence of images where the panic behavior gets gradually propagated among the crowd. We visually represent people in panic by using redheads.

• • • •

CHAPTER 6

MACES: Wayfinding With Communication and Roles

6.1 INTRODUCTION

To have agents moving within a dynamic complex environment, we need to have a high-level algorithm that will allow the agents to explore the environment and learn its features to perform the right navigation. We call this MACES (Multi-Agent Communication for Evacuation Simulation).

Agents move within complex virtual environments with several rooms, corridors, obstacles, stairwells, and doors that can be opened or closed at any time during the simulation (Figure 6.1). To navigate this virtual environment, route selection is carried out through an interactive high-level

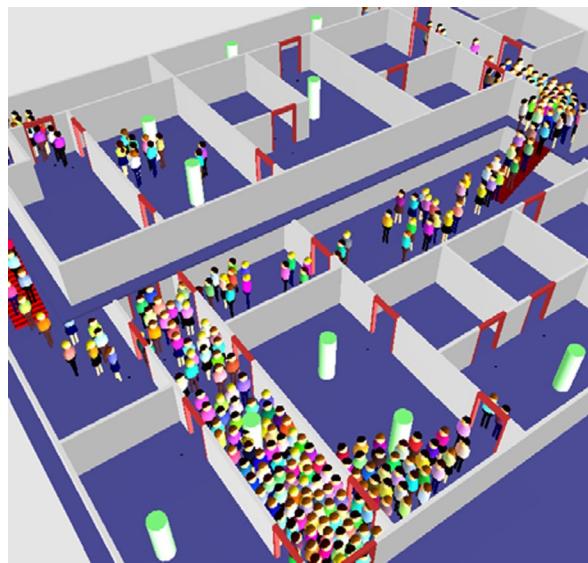


FIGURE 6.1: Example of complex building (Pelechano et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

wayfinding algorithm that dynamically calculates the global path based on the agent's knowledge of the environment (Pelechano and Badler 2006).

6.2 NAVIGATION ALGORITHM

Once the environment has been defined and an algorithm automatically calculates the shortest paths from each room to each exit, the building connectivity is stored in a cell and portal graph (CPG), with each cell corresponding to a room and each edge representing a door that connects two rooms. With the information contained in the CPG, the crowd simulation algorithm proceeds through three main steps (Figure 6.2).

The first step is the communication process. All the agents within a room share their knowledge about the environment (their mental maps containing information about blocked cells and subgraphs that have been fully explored finding no exit). At every time step, we are computing a high-level path over the CPG, which contains the information about the order in which the cells should be visited to get to an exit, and it is thus an ordered sequence of cell identifiers and portals connecting adjacent cells.

In the second step, each agent checks if the known shortest path has no known hazards, and if so it will just follow that path while adding the next cell to its mental map.

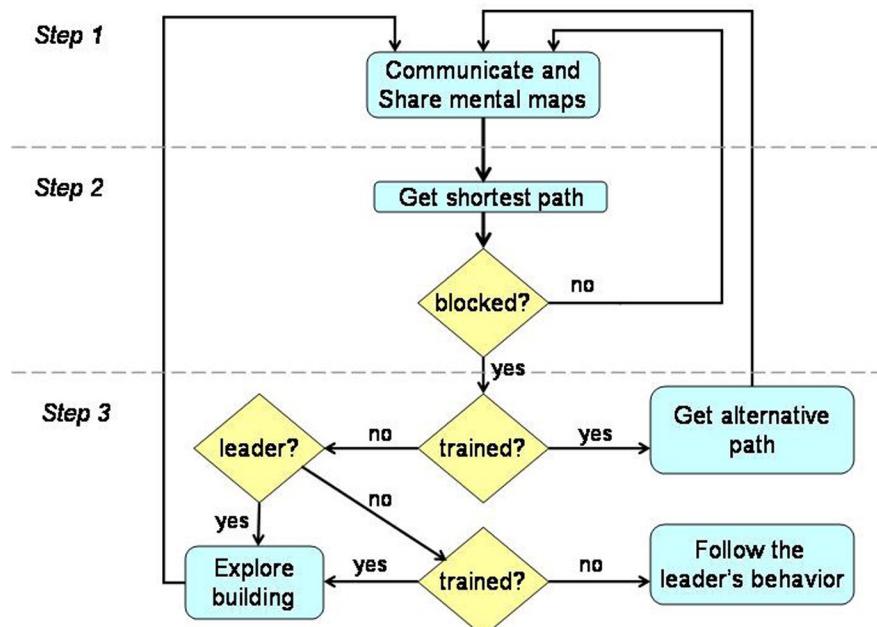


FIGURE 6.2: High-level wayfinding diagram (Pelechano and Badler 2006) C 2006 IEEE.

The third step occurs when there is some hazard in the shortest known path. In this situation, depending on the type of agent, there are different behaviors. If it is a trained agent, then its mental map contains the entire graph with all the portals, and therefore, it can just follow the next shortest path known from its current cell. If it is an untrained agent, then it needs to explore the building to find new routes toward the exit. This exploration will be done through depth first search (DFS). Since initially the agent does not have the entire graph in its mental map, this DFS is implemented in an iterative way, so that it will discover new rooms only when it sees a portal and crosses it. For untrained agents, it could also be a “follower,” which means that if it does not know what to do and as long as it can see another agent in the room, it will tend to follow the decisions taken by the other person instead of doing a DFS of its own.

6.2.1 Exploring the Building

When the known paths appear to be blocked, each agent needs to perform a graph search to find its way toward an exit. At this stage, we assume that agents that are not trained know one path to an exit, which is the one blocked, but have no additional information about any alternative path. In this case, traditional global path planning techniques such as A^* cannot be applied, since the agents will not know what heuristic to use. The exploration method used in our work is based on a DFS algorithm (Kwek 1997). At any given time, the mental map that an agent has in its internal memory will be a connected subgraph of the CPG that represents the environment. As an agent walks toward the exit, each cell visited will be added to its internal mental map (memory) and marked as *VISITED*. Initially, the agent knows all the traversed edges (portals) and visited rooms (cells) but ignores how many rooms the environment has and where each portal that has not been traversed leads to. When the agent finds its desired path blocked, it then needs to search for an alternative path. The agent will start exploring those rooms that appear as neither *VISITED* nor *FINISHED*. *FINISHED* cells are those that have been visited and either there are no more portals leading to other cells or all the adjacent cells have also been marked as *FINISHED*. If from the current room all the adjacent rooms have been either *VISITED* or *FINISHED*, the agent will chose a visited one randomly, since unlike *FINISHED* cells, visited cells can still lead to new paths. Figure 6.3 shows very briefly an iteration of the algorithm.

The main difference between DFS in graphs and what our agents perform is that in graph search, DFS is performed in a sequential manner going through all the cells. Our agents can benefit from the communication process with other agents, which will allow them to prune their graph search. When two or more agents meet within a room, they can exchange information about hazards and parts of the building (subgraphs) that have been fully explored by others, where no exit was found. The mental maps can be updated after visiting a room or after communication with other agents.

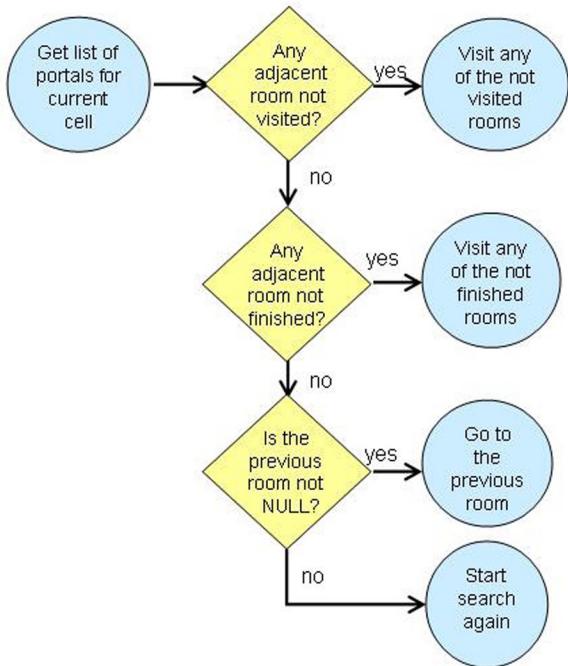


FIGURE 6.3: Exploration diagram.

To make the agents' behavior closer to real humans, we need to keep some considerations in mind. Consider the following examples.

- People during a conversation are unable to give such detailed information in terms of room connectivity about big subgraphs. Therefore, the information is limited to two levels of adjacency from the current cell in the CPG representing the virtual environment. We can think of it as, e.g., “the door on the left leads nowhere,” “the room on the right leads to another office, where there is no exit either”).
- People in panic tend to get disoriented. Therefore, when an agent is in panic, part or all of its internal memory could be “forgotten.”
- People in panic may also change their role from leader to follower. Therefore, an agent that was performing a search, after being affected by the panic behavior, may start following others instead of performing its own search.

When dealing with dynamic environments (e.g., portals that are locked or unlocked at different times), agents may have explored the entire graph, but if no exit has been found yet, then they will keep on searching hoping for a door to become unlocked.

We ran simulation experiments to investigate emergent properties of MACES:

- Comparison of trained leaders vs. untrained leaders
- Importance of leadership
- Psychology affecting roles and navigation
- Interactively modifying navigation based on impatience and changes in the environment.

A random search has been implemented exclusively for benchmarking purposes, since it is not the most realistic behavior for humans. In Section 6.2, we will appreciate the significant impact that communication has in the behavior of the crowd when executing wayfinding. Finally, we will show the impact of having trained agents in the crowd, and we will analyze the percentage of leaders that is actually useful to speed up the evacuation process.

For the experiments, we use three different buildings, all of them mazelike. Two of them have been randomly generated, and the third has been created by hand to produce a maze that better resembles a real building. The three mazes contain 100 rooms, 8 of them blocked by some hazard such as a stationary fire. For each set of parameters, we have run 25 randomly generated starting configurations for the crowds.

The populations (N) used for these tests range from 20 to 200 agents. The levels of leadership range from 0% to 100%. No leaders means they are all followers, and therefore, when several agents meet in a cell, one random agent makes a decision and the others will just follow. This case implies dependent agents: when they find themselves in a panic situation, they will always follow other agents instead of making their own decision. On the other hand, 100% leadership means each of them will perform its own decision-making process with its current knowledge, which in the case of trained people should be complete knowledge of the building's internal structure.

6.2.2 Communication Affecting Evacuation Times

As expected in real life, people will be able to reach their destination faster if they are able to communicate relevant information with other people in the crowd. Thus, we expect our autonomous virtual agents to also find evacuation routes faster when communication is being simulated.

In Figure 6.4, we can appreciate the different performance of the algorithm with and without communication for 200 agents. We are interested in finding the simulation step at which the simulation converges to 100%, meaning that the entire crowd has evacuated the building. As Figure 6.4 shows, the simulation with communication converges to 100% in almost half of the time that it takes the noncommunication case to converge.

Figure 6.5 shows the results obtained for different crowd sizes where all the agents represent independent individuals (they will make their own decision during wayfinding instead of following others). In this simulation, we are not yet considering trained agents, therefore, all the

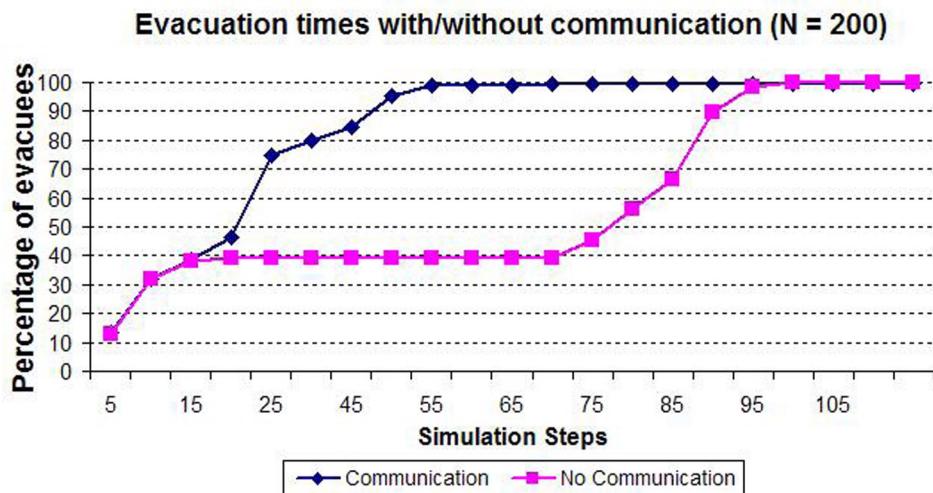


FIGURE 6.4: Communication vs. noncommunication (Pelechano and Badler 2006) C 2006 IEEE.

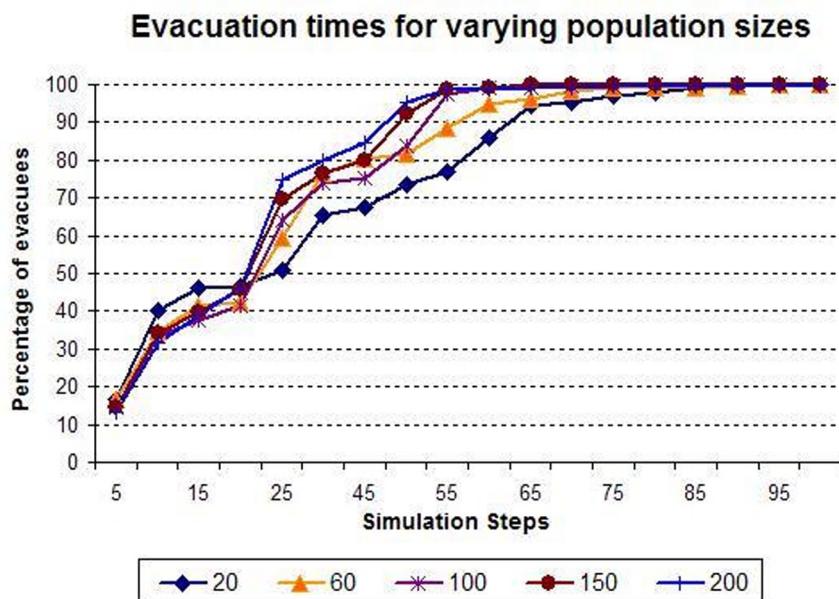


FIGURE 6.5: Evacuation time for different crowd sizes using communication: 100% untrained leaders (Pelechano and Badler 2006) C 2006 IEEE.

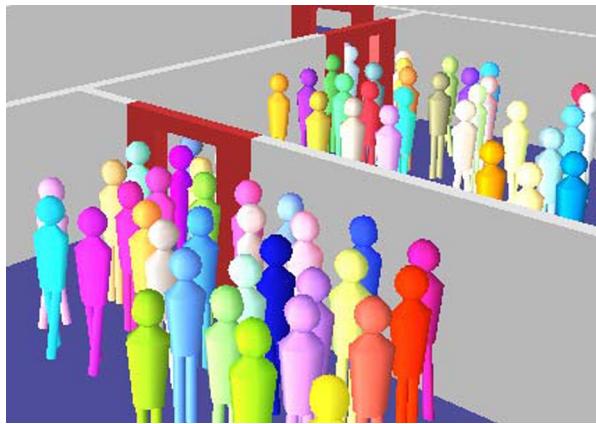


FIGURE 6.6: Congestion at doors (Pelechano and Badler 2006) C 2006 IEEE.

individuals in the crowd are unfamiliar with the internal structure of the building and will find out how to evacuate the building based on their own exploration of the building and their shared communication. The plot shows the evacuation times for crowds of size 20, 60, 100, 150, and 200 agents.

As we can see, the evacuation time decreases as the crowd size increases. This can be explained by the fact that for larger crowds, the probability of meeting another agent increases, and therefore, the important information about hazards in the building and explored areas spreads faster among the individuals. This information helps agents to prune their graph search and therefore find a successful path sooner.

It is important to notice though that this holds as long as the crowd is not so large that congestion blocks the doors, which will obviously decrease the evacuation time. This problem can be observed for crowds of over 500 agents, where the evacuation time is constrained by the number of exits and the flow rate through each of the doors (Figure 6.6).

6.2.3 Relevance of Having Trained Leaders vs. Untrained Leaders

Trained leaders are represented by agents that have complete knowledge of the environment, and therefore, they can provide more reliable information to the rest of the agents in the crowd. We would expect that the more trained leaders we had, the faster the evacuation would be. It is important to notice that although we are using an evacuation scenario as an example, trained leaders are important to consider during any type of crowd simulation. Trained leaders would always represent those individuals that can help others to reach their destination faster and more efficiently.

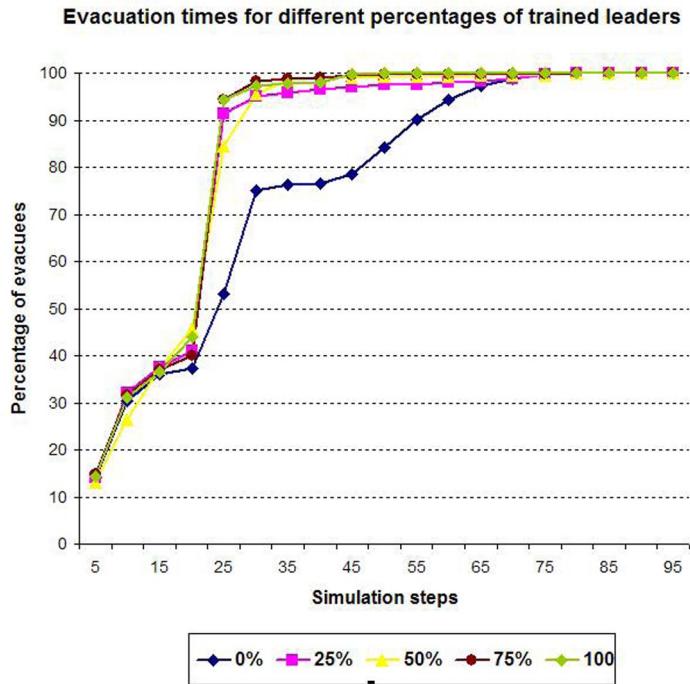


FIGURE 6.7: Evacuation time for 0%, 25%, 50%, 75%, and 100% leadership (Pelechano and Badler 2006) C 2006 IEEE.

As an example, we ran 25 simulations using a crowd size of 100 and 0%, 25%, 50%, 75%, and 100% of trained agents. Figure 6.7 shows the average evacuation times. Note that the percentage of evacuated people converges to 100% faster as the percentage of trained people increases.

Not everyone needs to be trained, however. We can determine the adequate percentage of leaders needed to have a fast evacuation. We have previously observed that there is not a big difference in the convergence values between 50% and 100% leadership, which means that there is no need to have a great proportion of leaders. Figure 6.8 shows that smaller percentages of leaders may be adequate.

Here we can conclude that an optimal percentage of trained people during the evacuation might be only around 10%. For lower values, the evacuation time for the same percentage of evacuees is at least doubled. On the other hand, having more than 10% trained people would only provide an evacuation speedup of at most 16%.

6.2.4 Importance of Leadership

We have shown the behavior of trained leaders against dependent agents with a “follow the leader” behavior, but there is still another case to consider. In real life some people have a higher probability

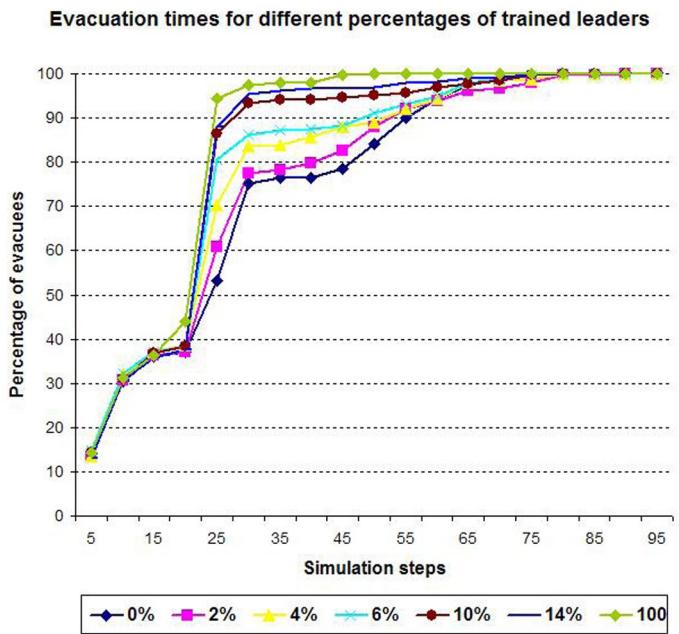


FIGURE 6.8: Evacuation times for small percentages of trained leaders (Pelechano and Badler 2006) C 2006 IEEE.

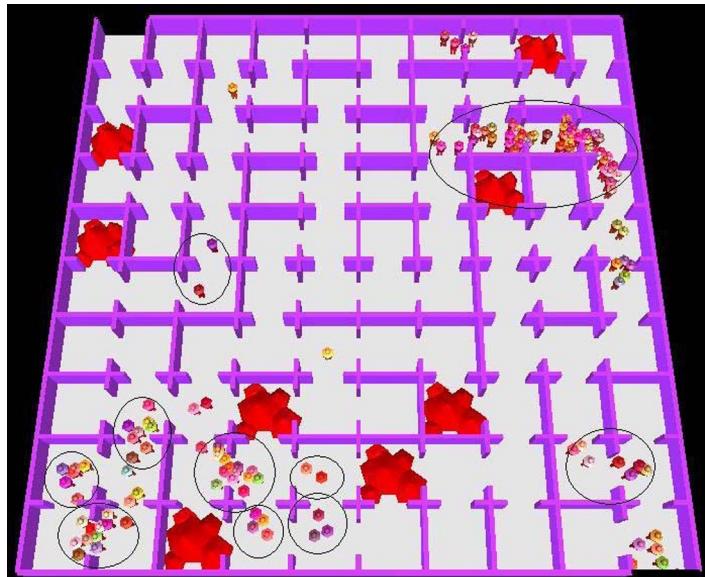


FIGURE 6.9: Sequence of crowd evacuation with high percentage of leadership (Pelechano and Badler 2006) C 2006 IEEE.

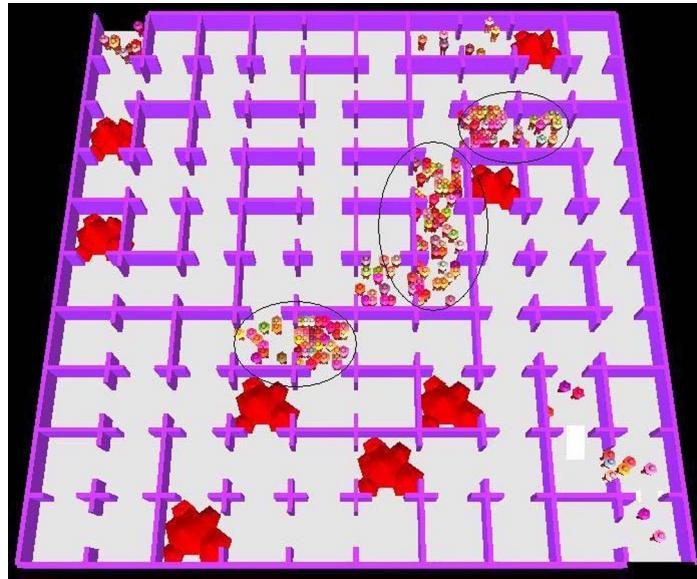


FIGURE 6.10: Sequence of crowd evacuation with low percentage of leadership (Pelechano and Badler 2006) C 2006 IEEE.

of becoming leaders for a group of people when an emergency occurs. They are usually independent individuals who by nature are able to handle emergency situations better and also tend to help others. In MACES, these people correspond to untrained leaders.

We show two images of the evacuation process. The first one (Figure 6.9) corresponds to a population with a high percentage of leaders, where most of the individuals in the crowd tend to make their own decisions when attempting to exit the building. The second sequence (Figure 6.10) corresponds to a population with a high percentage of dependent people, who will tend to follow any leader instead of deciding routes by themselves.

In the first population, we can observe an emergent behavior with many small groups of people. In the second population, the emergent behavior shows fewer but larger groups of individuals. When the number of dependent individuals is higher, the size of the groups formed tends to increase, since dependent people will not leave a group to try to explore new paths on their own. Instead, they tend to stay together and just go where the leader decides to go.

6.2.5 Simulating Psychology Affecting Roles and Navigation

As described by the psychology literature, people under panic may reach the point where they are unable to make their own decisions. Most people react to time pressure through an increase in the

speed of their actions, as well as by subjectively filtering information. In general, the evacuation of a building due to imminent danger is accompanied by considerable physical and psychological stress. Since rising stress levels have the effect of diminishing the full functioning of one's senses, this leads to a general reduction of awareness, especially the ability to orient oneself quickly in rooms and surrounding areas (Walda et al. 2003).

These behaviors need to be modeled in crowd simulation to improve the realism of its output. In MACES + HiDAC, stress due to imminent danger is simulated through panic. People under panic can change their role, going from leader to dependent individual. Therefore, as panic spreads, more people will become dependent and the overall evacuation time will change since dependent individuals do not contribute in terms of exploring and sharing information to speed up the wayfinding process of the group as a whole.

It is also necessary to influence the agents' ability to orient themselves quickly within the virtual environment. When individuals are under high levels of stress due to panic, their memory will be affected. We alter the mental map of the environment by removing subgraphs of their internal memory, and therefore these agents may exhibit disorientation by walking again through an area of the building that they had already visited instead of moving toward unexplored areas.

In Figure 6.11 we can observe the high-level navigation performed by one agent under normal (non-emergency) conditions. Initially, the agent only has knowledge about one short path to each of the two exits in the environment. Initially, he tried to reach the north exit, but after finding that door closed, he starts walking down the corridor toward the south exit. The agent has no knowledge about the alternative route through the southwest room, so when a door on his known south path blocks him, he will start to explore the environment as indicated by the white-dotted path. The agent will not walk again into a room that has already been visited and where no exit or alternative doors were found.

From the psychology literature, we know that a person in panic is very likely to feel disoriented, which can be modeled through memory decay based on the level of panic associated with the agent. We repeat the example above, but this time use an agent suffering from high levels of panic. Figure 6.12 shows the resulting trajectory. At the beginning, the agent follows the same trajectory as under normal conditions and tries to exit the building through the known shortest exits. Once the agent finds those known paths blocked, the exploration stage is initiated. Since this agent suffers from memory decay, he happens to walk several times in rooms that were previously explored, which makes the overall evacuation time longer and the path followed more chaotic.

Figure 6.12 shows six snapshots at different times during a simulation. The path followed in the period between each snapshot is shown with a white-dotted line. The path that goes from point 0 to 1 indicates the initial trajectory when the agent is following the known shortest paths and, after finding them blocked, then starts exploring the building in a way similar to the example without

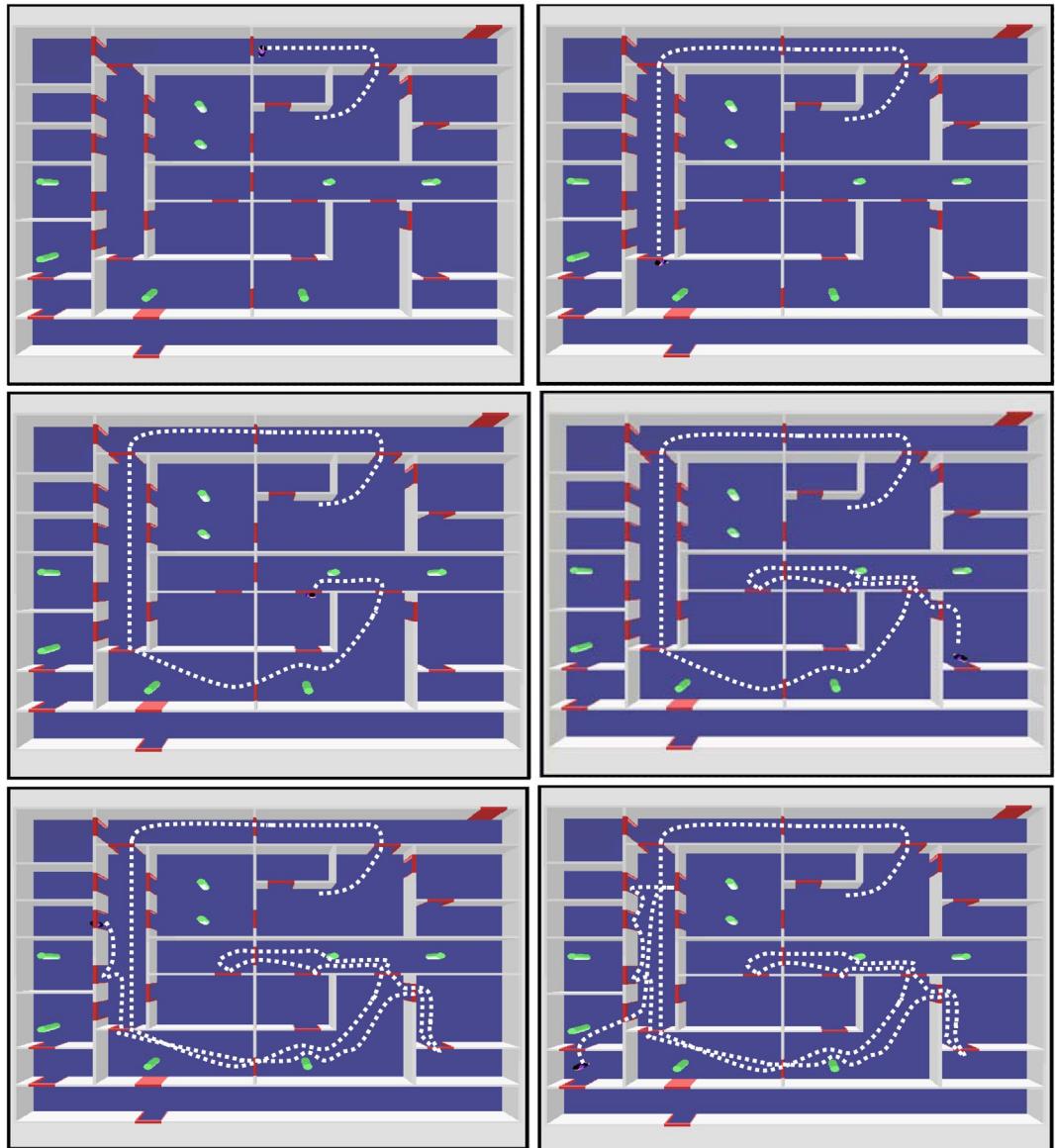


FIGURE 6.11: Autonomous agent exploring a virtual environment under normal conditions.

panic. From point 1 to point 2, the agent explores and ends up in the same doorway. From point 2 to point 3, we observe how the agent ends up back in the initial position after trying to explore a few other rooms on its way. Finally, we can observe how the agent moves from point 3 to point 4 and finds its way out. The last image shows the entire path followed.

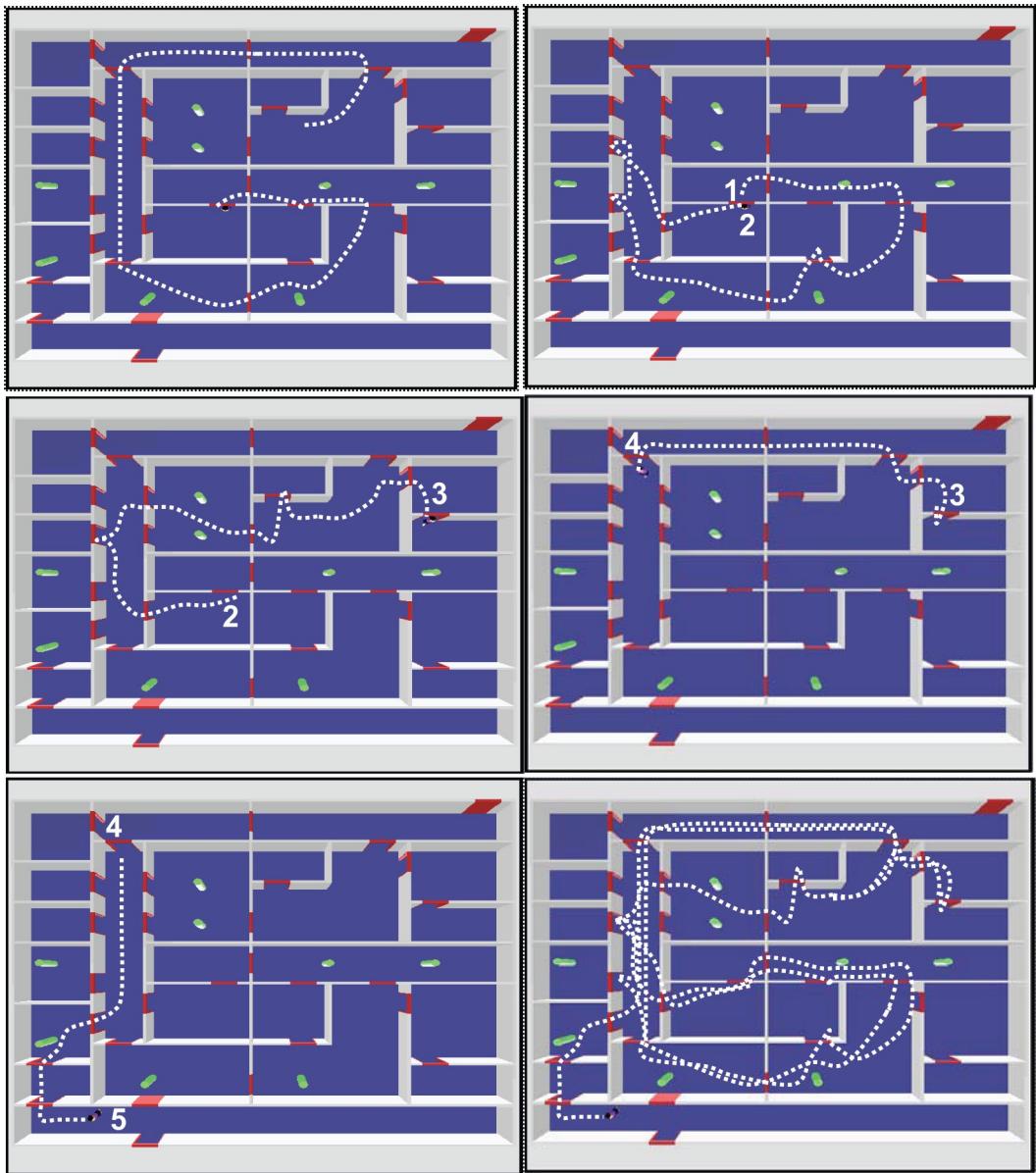


FIGURE 6.12: Autonomous agent exploring a virtual environment in panic and therefore exhibiting disorientation.

6.2.6 Interactive Navigation and Impatient Agents Avoiding Bottlenecks

In any real-time animation of crowds, it is important to demonstrate autonomous agents showing both reactive low-level behavior that affects their local motion based on dynamic obstacles or agents

and reactive high-level behavior based on real-time changes in the environment. These changes can be instigated by environment modifications (such as a door being closed or a hazard such as fire spreading) or congregations of agents generating bottlenecks. The autonomous agents in our system can rapidly react to these changes by detecting them and modifying their high-level navigation to plan an alternative route more applicable to the current situation.

When dealing with high-density crowds in buildings, bottlenecks can appear in the portals. HiDAC incorporates a high-level decision process that will allow impatient agents to react to this situation by finding an alternative path. As the low-level algorithm detects the bottleneck, it sends that information to the high level, which will try to find an alternative route based on what the agent can perceive from its current position (doors, obstacles) and the knowledge that the agent has about the internal connectivity of the building. If an alternative path is available, the high level chooses a new portal as the goal and sets an attractor point to change the direction of movement. Only impatient agents will exhibit this type of behavior. The user can set the percentage of impatient agents in the population through the parameter *impatience* from Table 3.1.

Changes due to congestion are driven by the impatience attribute and environment knowledge. Agents will choose alternative routes depending on their personal impatience value, the current congestion at the destination door, the distance left to reach that particular door, and the known alternatives. Figure 6.13 gives three snapshots of an animation where impatient individuals (represented by redheads) will perform a new high-level planning to choose an alternative route that allows them to detour around the congested area.

When a change occurs in the environment (e.g., a door is blocked), agents perceive and react to it. For a door change, the high-level algorithm needs to make a new wayfinding decision. This decision is sent to the low-level motion by modifying the next attractor point toward which the agent needs to walk. The agent detects this change in real time and, as the new attractor becomes available, the agent needs to steer toward the new destination point.

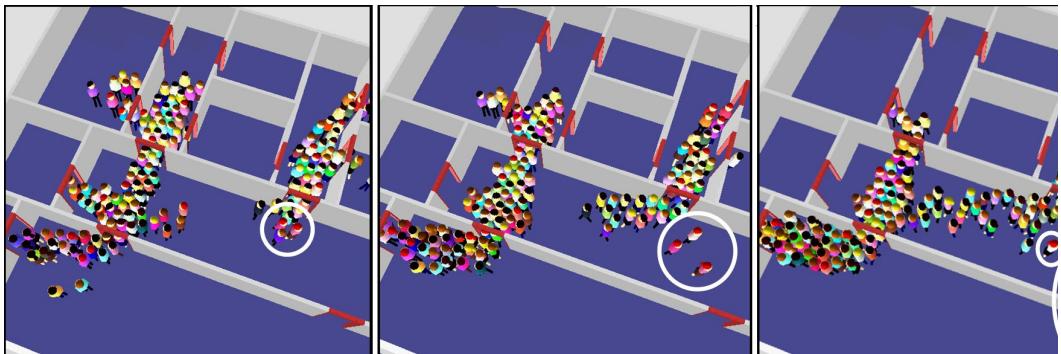


FIGURE 6.13: Results on interactive planning for impatient agents.

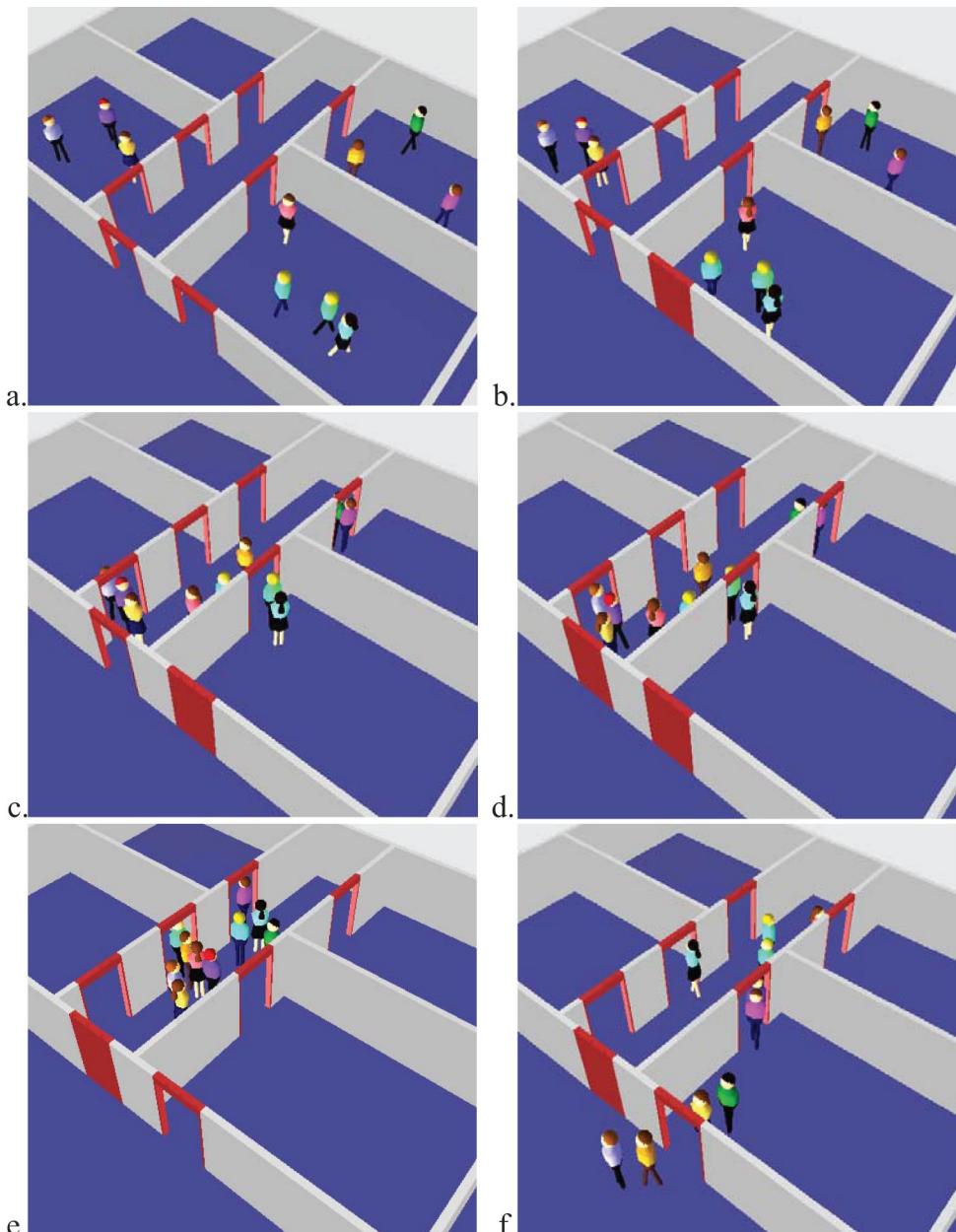


FIGURE 6.14: Crowd reaction to dynamic changes in the environment (Pelechano et al. 2007) C 2007 ACM, Inc. Reprinted by permission.

96 VIRTUAL CROWDS: METHODS, SIMULATION, AND CONTROL

Figure 6.14 shows an animated sequence where dynamic wayfinding is exhibited. Figure 6.14a shows the initial configuration consisting of a simple building with one corridor leading to the exit and four rooms connected to the main corridor. The bottom right room also has an exit door. Initially suppose that the agents in each of the rooms know about the corridor exit. Only those agents in the bottom right room can perceive there is another exit. When the simulation starts, agents walk toward the corridor except those in the bottom right room who walk toward their closest exit. Figure 6.14b shows the moment in which the exit in that room gets locked. Agents in this room immediately start moving toward the corridor to get to the next available exit, Figure 6.14c. As the agents walk down the corridor, the second exit also gets locked (Figure 6.14d). At this moment, the agents do not know how to get out of the building. The high-level module will have to perform navigation to explore and learn about the environment, while setting new attractors to animate the agents. For some time, individuals wander around the building looking through doors trying to find a solution (assuming they did not previously know where the other rooms lead to). Suddenly, the door in the bottom right corner is opened again (Figure 6.14e). Agents do not know about this until they walk through the interior door and perceive it (Figure 6.14). Eventually, all the agents manage to leave the building. This example shows the interaction between high and low levels to achieve realistic simulations with dynamic changes in the environment geometry.

This interactive planning behavior will be exhibited by those agents that have the ability to explore the environment — those with a leadership role or those with a dependent role who find themselves isolated.

• • • •

CHAPTER 7

CAROSA: Functional Crowds

7.1 APPLICATIONS WITH ACTIONS

When creating a simulation with multiple humans such as an urban environment or office building, many graphical, semantic, and functional elements must be created and brought together. Graphical models of the environment, objects, and characters need to be created and annotated with information that enables them to be reasoned about and interacted with. The behaviors of the characters and environment need to be described in concert with the objective of the simulation. Animations, whether motion captured, key-framed, scripted, or procedural, need to be created and linked to the higher-level behaviors and ultimately back to the characteristics of the characters. Naturally, a simulation framework is required to provide control for the characters and environment. This framework focuses on adding actions to such simulations by linking human characteristics and high-level behaviors to graphical depictions.

There are numerous applications for human crowd simulations. Human factors engineering and architectural visualizations could advantage these simulations to more efficiently design and lay out spaces. Military simulations, virtual reality, gaming, and other entertainment enterprises might better control nonplayer characters and more easily create realistic environments. Even the seemingly standard application of simulated crowds, evacuation scenarios, could be improved by starting the evacuees in normal starting positions instead of randomly distributing them in the environment. Finally, as we are able to create simulations that take into account more high-level human characteristics such as roles and individual differences, we may be able to use these simulations as tools for learning and exploring real human behaviors, observing emergent behaviors, and generating predictive models that can be used as design tools.

This chapter outlines a framework for creating and simulating heterogeneous populations that depict actions and behaviors linked to human characteristics such as role (see Figure 7.1). To do so, we need to

- specify the characteristics (e.g. roles, goals, constraints) of individuals or groups including their behaviors and how they might differ from other individuals;



FIGURE 7.1: All characters are instructed to go to class. Lecturers then go to the front of the class and students take a seat.

- establish the temporal (e.g., daily) activities of such individuals or groups according to their occupations or roles;
- access a library of parameterized animated behaviors that can be selected contextually, varied statistically, applied to agents, and executed in real time in a typical simulation environment;
- give the agents enough attention and perception to react to the environment, people, and events around them.

We will begin by describing a parameterized action representation that is used to represent the semantics of both actions and objects and links lower-level navigation and motion controls to higher-level action selectors. We will also describe four different types of actions that we believe will result in more interesting emergent behaviors. We will then give an overview of the CAROSA (Crowds with Aleatoric, Reactive, Opportunistic, and Scheduled Actions) system including an

overview of the processing of Parameterized Action Representation (PAR) actions and how the system interacts with HiDAC.

7.2 PARAMETERIZED ACTION REPRESENTATION

PAR was designed as an intermediary between natural language and animation. A software system for interpreting PARs and animating them has been designed and implemented. PAR has been used for developing virtual environments for training (Bindiganavale et al. 2000), instructing virtual humans (Allbeck et al. 2000), controlling virtual aggregates (Allbeck et al. 2002), and validating maintenance instructions (Badler et al. 2002).

Here we will provide only a quick overview of PAR and its components and will focus on developments needed to successfully use the PAR framework for large-scale crowd simulations. The details of PAR and the PAR software system can be found elsewhere (Allbeck et al. 2000; Badler et al. 2000; Bindiganavale et al. 2000; Allbeck and Badler 2003), and the current parameters can be found in Appendix B.

PARs are stored in two hierarchical databases: an action hierarchy called the *Actionary* and an object hierarchy. These hierarchies are based on data from several sources including WordNet (Fellbaum 1998), the Unified Verb Index, and Cyc (Lenat and Guha 1990). The hierarchical nature of the databases facilitates the addition of new actions and objects. Once the databases are populated with base actions and objects, new entries can be added by finding their proper placement in the hierarchy and simply specifying their distinguishing parameters. Inheritance will fill in all of the other parameters. For example, *desks* may be specified high in the object hierarchy. *Computer desks* would then be a child inheriting properties such as the object's *purpose*: to be a work surface and container. It might then also contain information specific to this type of desk, such as containing a *keyboard drawer*.

7.2.1 Key Fields of the Action Representation

Participants are the agent and object parameters of PARs. The agent is the virtual human executing the action. Selected virtual humans from some population can be assigned as single agents. The object type is defined explicitly for a complete representation of a physical object and is stored hierarchically in a database. Each object in the environment is an instance of this type and is associated with a geometric model in a scene graph.

Some of the fields of PAR are designed to aid in or shortcut the task planning process. The applicability conditions of an action specify what needs to be true in the world to carry out an action. These can refer to agent capabilities, object configurations, and other unchangeable or uncontrollable aspects of the environment. The conditions in this Boolean expression must be true

to perform the action. For *walk*, one of the applicability conditions may be: *Can the agent walk?* If these conditions are not satisfied, the action cannot be executed.

Preparatory specifications are a list of <condition, action> statements. The conditions are evaluated first and must be satisfied before the current action can proceed. If the conditions are false, then the corresponding action is executed; it may be a single action or a complex PAR with sub-actions. In general, preparatory specifications produce a limited form of automated planning, e.g., to indicate explicitly that a handle has to be grasped before it can be turned in order to open a door.

Termination conditions are a list of conditions that, when satisfied, complete the action. Terminations may be caused by success or failure. Particularly in applications dealing with mechanical devices, termination conditions can be crucial. Actions such *loosening and removing* a nut would result in similar performances, but with one terminating before the nut detaches from the bolt and the other after it separates.

Post assertions are a list of statements or assertions that are executed after the termination conditions of the action have been satisfied. These assertions update the database to record the changes in the environment. The changes may be because of direct effects or side effects of the action. For example, increasing the energy level of an agent might be a post assertion of that agent drinking a caffeinated drink.

PARs come in two forms, uPARs and iPARs. uPARs are uninstantiated PARs, lacking characteristics specific to a particular scenario; think of them as *patterns*. iPARs are uPARs that have been instantiated. For example, a *desk* might have its *size* parameters instantiated with 4' or 6'. An action, such as *copying* (using a copy machine), should be instantiated with the virtual human performing the action and the specific object(s) being used.

7.2.2 Key Fields of the Object Representation

The object type is defined explicitly for a complete representation of a physical object and is stored hierarchically in the *Actionary*. Each object in the environment is an instance of this type and is associated with a geometric model. Agents are special types of objects that can select actions to perform.

The state field of an object describes a set of constraints on the object that leave it in a default state. The object continues in this state until a new set of constraints is imposed on the object by an action that causes a change in state. Objects also have a list of actions they are capable of participating in. In a sense, these can be thought of as affordances of the objects (Gibson 1977). These capabilities can aid in a planning process. For example, an agent might be tasked with pounding a nail, but be unable to find a hammer. He might then search through the capabilities of the objects available to him and discover that the stapler and his shoes could be used for pounding. Certainly, this does not

capture all of the subtleties of interactions in the real world like degrees of hardness and fragility, but it does provide a bit of information that can improve the robustness of the system.

Other fields in the representation help to determine where an object is. There is a position field that gives the 3D coordinates, but there is also a location field that provides more semantic-level information. The location field is a link to another PAR object that the object is contained in. For example, a desk object's location would be the room object that it is in. There is also a contents field that lists all of the containing objects. These fields can help to determine the spatial locations of resources and aid in navigation on a global level. Navigation and collision detection are often separate processes with different levels of information required. For an agent to go to an object, the navigation level would like to know what room the object is in and would plot a course to that room. The collision detection and avoidance level would ensure that no collisions take place along the way and help to guide the agent to the object once in the room.

Other important fields include the reference coordinate frame, a list of grasp sites, and directions defined with respect to the object. These fields can be used by motion generators or animation components to interact with the objects. For example, an action for sitting on a chair would need to know where the front of the chair is so that the agent can be maneuvered into place and would need to know where the seat of the chair is so that the agent sits on that surface.

7.2.3 Four Types of Actions

Human activity or behavior has been analyzed in many research areas including philosophy, psychology, anthropology, cognitive science, human factors engineering, artificial intelligence, robotics, and of course computer animation. We are classifying behavior into four types: *scheduled*, *reactive*, *opportunistic*, and *aleatoric*. In artificial intelligence and related fields, behavior is often classified as reactive or planned in accordance to what precipitates the behavior (Russell and Norvig 2002). Agents whose actions arise through planning are often called *deliberative agents* and are normally contrasted with *reactive agents*.

Hybrid control systems have also emerged. The architecture created by Funge et al. (1999) is an example. They outline three types of behaviors: *goal-directed*, *predefined*, and a *middle ground*. Goal-directed behaviors correspond to a deliberative process. Predefined behaviors are reactive rules that are sometimes programmed as finite-state machines. The middle-ground behaviors combine elements of both. These behaviors contain precondition axioms and are often complex. Precondition axioms can be viewed as rules to determine when the behaviors should be performed, like reactive behaviors, but also can aid in the planning process. Complex behaviors combine sub-actions, therefore requiring fewer actions to be chained together in the planning process. For an overview of architectures for embodied agents, see Allbeck and Badler (2002).

Scheduled Actions. In our approach, we view scheduled actions as similar to goal-directed, planned, or deliberative behaviors. CAROSA does not include a sophisticated planner, although nothing precludes this option. Scheduled activities are not restricted to actions traditionally thought of as planned; instead, they arise from the specified role of individuals or groups. These roles are provided by the simulation creator, and therefore, scheduled behaviors are simply actions associated with a group or individual at a specific time and for certain duration. For our example, scheduled actions may include *arriving at work*, *attending classes*, *eating lunch*, *leaving work*, and many others (see Figure 7.2).

Reactive Actions. Reactive actions are triggered by contextual events or environmental constraints. Many of these behaviors arise from the crowd simulator (see Chapter 5). For example, reactive behaviors include a character altering its heading or slowing down to avoid obstacle collisions or other characters. Recalculating a path is another reactive action that can be handled entirely by the crowd simulator. Other reactive actions, such as acknowledging someone as they pass in the hallway, are not handled by the crowd simulator. These reactions will be specified and recognized in a rule-



FIGURE 7.2: Characters go to classes according to their schedules.

based attention mechanism. For our example, reactive actions will add richness to the texture of the simulation by depicting the relationships between characters: *a nod of acknowledgement to coworker, pausing to chat with a friend, or ignoring a stranger.*

Opportunistic Actions. Opportunistic actions are our middle ground. They are not scheduled, but are in a sense planned for and are a reaction to context. They include standing orders, but are not quite “orders” since they may be taken based on goals and priorities rather than as absolutes. Opportunistic actions arise from explicit goals and priorities. These behaviors are akin to the hill-climbing behaviors of characters in the video game *The Sims*. In *The Sims*, characters are created with needs including *hunger*, *hygiene*, *energy*, and *bladder* depending on the character’s activities, and these needs are fulfilled or grow more urgent. As needs grow more urgent, activities that will fulfill these needs gain priority. Without player intervention, actions are chosen based on the current levels of the needs and proximity to objects required to fill the needs. If the character has hunger and is near food, it will eat. If the character is near a restroom and has a high *bladder* need, it will enter and use it.

While we envision our opportunistic actions similarly, the implementation will be a bit different. In *The Sims*, current proximity is heavily weighted when choosing an action, and time is ignored entirely. We plan to take into account time and future proximities. For example, a character may be working in his office and have a non-emergent *energy* need and a meeting to attend in a few minutes. The character could then attempt to address the need by stopping by the lunch room for a cup of coffee on the way to the meeting. This will require him to leave a couple of minutes early and to know the lunch room is close to the path to the meeting room.

As these needs increase, so do the priorities of the fulfilling actions. It would be possible for scheduled actions to be delayed or ignored to fulfill needs, but we intend to balance priorities with proximity to limit this behavior. If desired, the simulation creator could alter this balance.

Aleatoric Actions. Aleatoric actions are random but structured by choices, distributions, or parametric variations. Take for example *working*. For many of the characters in an office building scenario, this would be a default behavior. If there is nothing scheduled and they have no pressing needs, they would be seen *working*. The aleatoric nature of the behavior stems from what the sub-actions might be and their frequency of occurrence. The management science community has provided some statistics on the work patterns of managerial organizations (Kurke and Aldrich 1983). Table 7.1, for example, provides data on the proportion of time managers spend during a day on desk work and telephone calls (at least circa 1983).

Our aleatoric behaviors are somewhat similar to the stochastic behaviors found in *Improv* (Perlin and Goldberg 1996), but are based on real statistical data (to the extent such is available) or explicit specifications through a suitable user interface. The choice of gesture, while also correlated with speech, might be determined aleatorically (see Figure 7.3).

TABLE 7.1: A portion of a table comparing the work of top managers of small, intermediate, and large organizations taken from the work of [Kurke and Aldrich \(1983\)](#)

CATEGORY	SMALL ORGANIZATIONS	INTERMEDIATE ORGANIZATIONS	LARGE ORGANIZATIONS
Number of activities per day	77	34	22
Desk work sessions			
Number per day	22	11	7
Proportion of time (%)			
	35	26	22
Average duration (min)	6	12	15
Telephone calls			
Number per day	29	10	5
Proportion of time (%)			
	17	8	6
Average duration (min)	2	4	6
Schedule meetings			
Number per day	3	4	4
Proportion of time (%)			
	21	50	59
Average duration (min)	27	65	68
Unschedule meetings			
Number per day	19	8	4
Proportion of time (%)			
	15	12	10
Average duration (min)	3	8	12

Small organizations: three presidents, 6 days of observation (Choran study); intermediate organizations: four top managers, 20 days of observation (Kurke and Aldrich); large organizations: five chief executives, 25 days of observation (Mintzberg study).



FIGURE 7.3: Gestures may be modeled as an aleatoric behavior.

7.2.4 Application to Crowds

One of the goals of this work is to be able to more easily create simulations with large numbers of virtual human characters. In addition to using a standard user interface, we propose reusing PAR and its database of actions and objects. PAR provides semantics about actions and objects that enables higher-level systems, such as schedulers and planners, to only consider concepts at that appropriately higher level. PAR can then fill in the information needed to animate the actions and resulting interactions with objects. If done properly, once these semantics are specified for an action or object, they will not need to be specified again. This means less work in setting up the next scenario. For example, if an agent's actions include presenting to a meeting, the *present* action might include standing in front of the audience, attending to the audience, talking about a topic, and gesturing. The specifics of the action, such as the topic, the room, and the manner of the gestures might be changed from one scenario to another, but PAR would be able to automatically fill in the actions to get the agent to the meeting room and in place and run through the base animations, altering them appropriately to these new parameters. This means a scenario creator would only need to

update data for fields specific to this scenario, not for creating the semantics from scratch. As noted earlier, adding new objects and actions also requires less work because of the hierarchical nature of the databases. This is, of course, after there are a sufficient number of entries to jumpstart the database.

PAR fills in details for action execution, subject to context and statistical distribution. For example, consider the start times of actions. All the workers in an office building might finish their day's shift at 5 PM. The agents would begin their next task, that of going home, but they do not start at exactly the same time nor do they teleport there. Each one has to follow a set of waypoints to reach home, but even that cannot simply be executed. Resource competition, dynamic obstacles, secondary goals (e.g., stop at market), and crowding with other agents create low-level path and task decisions at the PAR level that add contextual variation and social realism.

7.3 CAROSA SYSTEM OVERVIEW

What we ultimately desire is a system that depicts human behavior as a realistic 3D animated background texture. Even being able to automatically distribute individuals in an environment to places that they would likely be at a certain time of day based on their roles would enhance crowd simulations. As initial positions, this alone would help increase the accuracy of crowd evacuation simulations, for example. Our aim is to create a framework that a scenario author can use, relatively easily, to create simulations that contain virtual humans with roles and from those roles scheduled behaviors that will enrich simulations with apparently purposeful behaviors.

To create a virtual world, there are many components that need to be constructed. Geometric models of the environment, objects, and agents need to be created and articulated. To interact with the environment intelligently, both the environment and objects need to be tagged with semantic data. Animations also need to be created and linked to higher-level behaviors and ultimately agent roles. These roles need to be specified and assigned to agents. This includes some temporal specification of the behaviors during a typical day, reaction to environmental events and other agents, and a specification of relationships between agents.

Figure 7.4 shows the components and connections of the CAROSA system. The *Scheduler* and *Actionary* exchange data in the form of schedules as well as the definitions of roles and objects. This data can be created in either module and passed to the other. The scheduler that we are currently considering is Microsoft Outlook. Microsoft Outlook is off-the-shelf software that is widely used. It provides an interface for specifying roles and tasks and creating timelines and schedules. We believe it can be a useful tool for creating agent populations. The *Actionary* is a MySQL database containing PARs.

We do not address the actual modeling of the environment, objects, or agents. We believe there are sufficient software packages available for this purpose. For articulating the figures, we

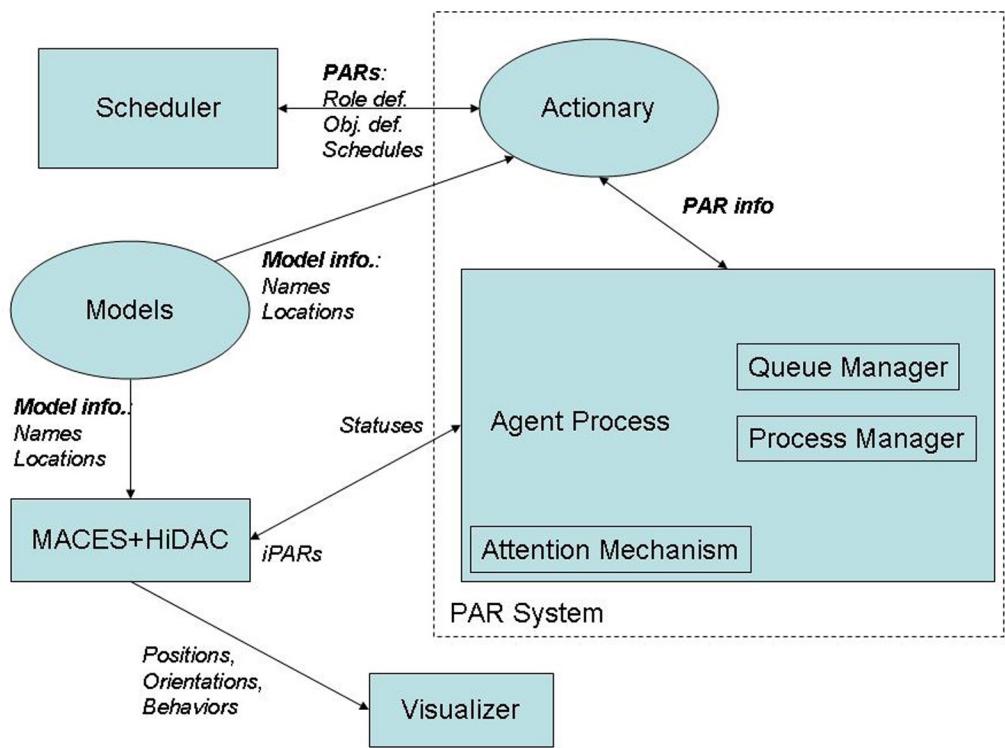


FIGURE 7.4: CAROSA system diagram.

have integrated CAL3D ([CAL3D 2008](#)), an open-source software package that provides elementary functionality for animating human figures, into the HiDAC crowd simulator (see Chapter 6). HiDAC determines the positions and orientations of the human models and what animation clip to play based on information in PARs. It then calls the necessary display methods to display the simulation. CAL3D and the display functionality may be replaced to improve both animation quality and the frame rate. For example, [Ahn et al. \(2006\)](#) has done research on motion level of detail for crowd simulations. Because of the semantic data available through PAR, we may be able to change the level of detail based on the importance of the character in the scene.

MACES + HiDAC provides us with the ability to direct agents through an environment and stop them in certain locations while avoiding collisions and providing other crowd behaviors. MACES + HiDAC, however, provides only low-level navigation in terms of planning for agents.

Currently, agents can stop at predetermined locations in a room to perform actions, but these actions must be scripted by hand and added directly into the code.

The PAR representation enables us to direct the agents at a higher, less-detailed level and provides us with an architecture that includes at least a minimal amount of planning. Additionally, it gives us the opportunity to have agents that react to environmental changes more intelligently. MACES + HiDAC already provides low- and some middle-level locomotive reactions, such as pushing to get through a crowd, finding an alternative path if a doorway is blocked or overcrowded, and overtaking behaviors. PAR can add additional richness at higher levels through its use of preparatory specifications as a partial planner. For example, if an agent needs to use a resource such as a copy machine and finds that it is broken, he may contact the repair service or find another copier.

During a simulation, the *PAR system* can be viewed as the central controller. Information about schedules (provided by the *Scheduler*) and information about the models (populated before the simulation begins) can be found in the *Actionary*. The *PAR system* uses this information and information about the statuses of dynamic environmental elements provided by HiDAC to help determine what the characters' behaviors should be. The *PAR System* also contains *Agent Processes* that control the opportunistic, reactive, and aleatoric actions.

7.3.1 PAR System

For our purposes, the PAR system has two main components, the *Actionary* and the *Agent Processes* (see Figure 7.4).

7.3.2 Actionary

All instances of physical objects, agents, and uPARs are stored in a pair of persistent hierarchical databases. One of the databases contains the objects and agents, and the other contains the uPARs. During the initialization phase of a simulation, a world model is created from the databases. This model is constantly updated during the simulation, recording any changes in the environment or in the properties of the agents and objects.

The agent processes can query the world model for the current state of the environment and for the current properties of agents and objects in the environment. Additionally, through the *Scheduler*, the simulation creator can update the *Actionary* during the course of the simulation.

7.3.3 Agent Process

An agent process controls each instance of an agent, either as an individual character or as a group of characters with similar behavior patterns. Each agent process has a queue manager that manages

a priority-based, multi-layered queue of all iPARs to be executed by the agent. The various tasks of an agent process are to:

- add a given iPAR at the top level of the queue;
- communicate with other agent processes through message passing;
- trigger different actions for the agent based on the agent's role, messages received from another agent process, and the existing environmental state;
- return the process status (on queue, aborted/pre-empted, being executed, completed, etc.) of an iPAR;
- insure nonrecursive addition of iPARs resulting from rules (i.e., reactive actions).

The queue manager in the agent process is implemented using PaT-Nets (i.e., *parallel finite-state machines*) (Badler et al. 1993). Each iPAR is assigned a priority number by the user or by the context. At any time, if the first action on the queue has a higher priority than the iPAR currently being executed, the queue manager pre-empts the current action. In general, either after pre-emption or completion of an action, a new action is selected to be popped from the top level of the iPAR queue of the agent and sent to a process manager. The selected new action has the highest priority in the first subset of monotonically increasing actions (with respect to priorities) at the beginning of the queue.

For each popped iPAR, a process manager first checks the termination conditions. If the termination conditions are already satisfied, then the action is not performed. If they are not satisfied, the applicability conditions are checked. If these conditions are not satisfied, the entire process is aborted after taking care of failure conditions and proper system updates. If the applicability conditions are satisfied, the preparatory conditions are then checked. If any of the corresponding preparatory actions need to be executed, an iPAR is created (using the specified information of the uPAR, agent, and the list of objects) and added to the agent's existing queue of iPARs. It should be noted that the queue of iPARs is a multi-layered structure. Each new iPAR created for a preparatory action is added to a layer below the current one. The current action is continued only after the successful termination of all the preparatory actions. If the current action is very complex, more iPARs are generated, and the depth of the queue structure increases. During the execution phase, a PaT-Net is dynamically created for each complex action specified in the execution steps or in the preparatory specifications. Each sub-action corresponds to a subnet in the PaT-Net. The PaT-Nets are also used to ultimately ground the action in parameterized motor commands to the embodied character. More details of the agent process algorithm can be found in Appendix C.

HiDAC communicates with the PAR system through the agent processor. Each HiDAC character is linked to a corresponding agent process and, as a result, its iPAR data. Thus, HiDAC can update the status of each agent it is controlling including reporting failure states. Each agent process also has a link back to the HiDAC characters it is controlling. When the process manager

determines that an iPAR is ready to be executed, it places the necessary information (e.g., goal position, orientation, motion clip) on each appropriate agent's action list.

7.3.4 Processing the Four Action Types

Scheduled actions are simply placed on the agent's queue as iPARs. These iPARs may have been created directly in the *Actionary* or may have originated in the *Scheduler*. In either case, they contain all of the needed semantic information for the action, including the start time and duration of the action. Actions are not processed by the queue manager before their start time, so all of the scheduled actions can be sent to the queue at the beginning of the simulation. It is also possible to edit these actions during the simulation and even to add additional scheduled actions.

Reactive actions are based on observations from the environment. We will need to build a simple *attention mechanism* in order to implement these actions. This mechanism will consist of a list of rules. When the conditions are true, an action in the form of an iPAR will be created and added to the front of the agent's queue. The conditions need only include the agent or group of agents that are subject to this reaction and the property or state of the environment that they are reacting to. The attention mechanism will then query the state of the simulation through HiDAC to determine if any of these conditions have been met. These queries will be restricted to the room or hallway containing the agent.

Opportunistic actions are based on available time, proximity, and level of need or desire. The PAR representation of each agent will contain values representing their current levels or reservoirs of needs (Silverman et al. 2003). This value will decay over time until the need is met. Each agent will also have corresponding actions on their queue that will fulfill these needs. The priority of these actions will be based on the level of need and on the proximity to objects needed to fulfill the need. Hence, if the need becomes great enough, the action will take priority over other actions and get executed. Also, if the level of need is at the middle but an object to fulfill the need is nearby, the priority will be such that the action will be executed. Time is also a factor in priorities. If the agent has little time before a scheduled appointment (scheduled future action), it may not stop to fulfill a need, because the priority of the appointment would be higher. If their schedules permit, agents may consult their need levels before determining when to leave for a meeting and factor in satisfying needs.

Another variable would naturally be the nature of the character. Individual differences encompass a wide range of properties that may be considered “who a person is.” Our aim is not to implement all of these properties, but we would like to demonstrate where such factors might easily be included. This is one such place. We plan to characterize agents as *hardworking* and *uninspired*.

Hardworking agents will be more likely to prioritize scheduled tasks, and uninspired agents will be more affected by their proximity to need fulfilling objects.

Aleatoric behaviors are statistically driven default or idle behaviors. These behaviors will appear on an agent's queue when there are no other actions there with a priority greater than a threshold. Opportunistic actions will always be found on the queue, but they may have little or no priority. Generally, we view these behaviors as complex PARs with unordered subactions that contain information about their frequency supplied from distributions given or found in literature. When the process manager processes such an action, it chooses sub-actions to perform based on the distribution information. Durations for these actions are also sometimes available as means and standard deviations and could be included.

• • • •

CHAPTER 8

Initializing a Scenario

Creating a new simulation requires numerous elements. Buildings and spaces need to be created and laid out with objects such as furniture or vehicles. Character profiles also need to be defined for the population. This includes roles, relationships, and possible other individual differences such as personality profiles. These character profiles then need to be linked to actions with motion models as well as semantic data. In this chapter, we will discuss these aspects of scenario creation.

8.1 BUILDING MODELING

The navigation of crowds of autonomous agents in complex environments requires having an efficient abstract representation of the virtual environment where the agents can rapidly perform wayfinding. This abstract representation can also be used to store some precomputed information about the environment that will speed up the navigation and also be helpful to achieve fast perception for local motion computation. Our system (MACES + HiDAC) can handle two types of environment generation. The first generates a mazelike building environment from input parameters (dimensions, number of exits, and number of hazards) (Figure 8.1). The second creates a building (Figure 8.2) from a simple floor plan editor (Appendix A).

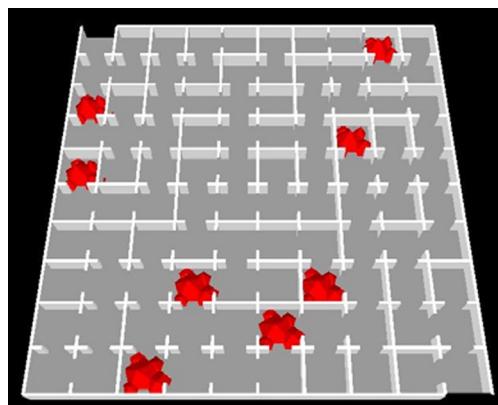


FIGURE 8.1: Example of a maze used for our experiments with two exits and eight hazards (Pelechano and Badler 2006) C 2006 IEEE.

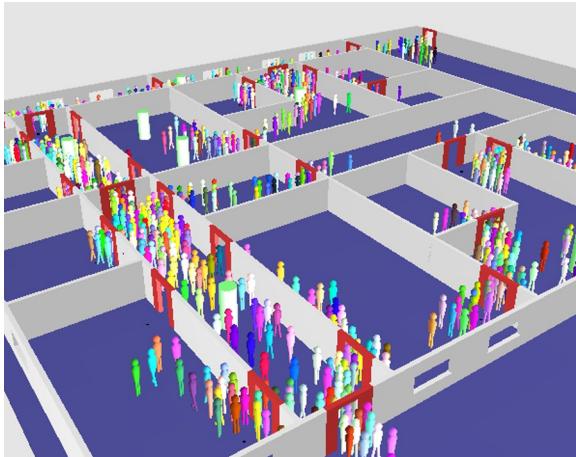


FIGURE 8.2: Example of building plan used for evacuation simulations (Pelechano and Badler 2006) C 2006 IEEE.

For any given environment, our system automatically generates a cell and portal graph (CPG) where each cell represents a room and portals correspond to doors. The stairwells are treated as cells with two portals, one at each end of the stairwell.

Each room contains information about the list of walls in that room and the list of static obstacles, so when the agents need to perform local motion within the room, they will query the room for those lists of static obstacles against which they need to perform collision detection.

To achieve real-time interactive navigation, some other relevant information about the environment is precalculated and stored. Among the information stored are paths toward the exits, distances from each door to a destination point, and the position of the attractors that will be used during the local motion to steer the agents. To save space, this environment information is also considered internal knowledge of the agents; however, since our agents can have different roles, and therefore exhibit diverse behaviors, they will have access to different levels of information that will provide diversity in their level of knowledge about the environment.

8.1.1 Cell and Portal Graph Automatic Generation

Our system receives as an input an arbitrary building model and creates a CPG, identifies all the walls and obstacles that belong to each cell, and stores that information within the cell. The building geometry is represented by a grid decomposition that contains different elements representing walls, doors, obstacles, stairs, windows, etc.

From that representation, an intermediate 2D grid is created for each floor, where the value 0 will be assigned to grid cells representing free space, -1 will be assigned to those grid cells con-

taining a wall, -2 assigned to grid cells occupied by doors, and other negative values are assigned in the same manner for holes and stairs. This 2D grid is not employed in the final movement of the agents as the cellular automata models do, but instead it is used as an intermediate step to obtain the geometric information of the building.

The algorithm proceeds in four steps:

- Generate the CPG for each floor.
- Identify stairs and link floors through new cells.
- Identify and store walls.
- Identify and store obstacles.

The cells in the graph generated correspond to the rooms and determine the continuous space in which the virtual autonomous agents will perform navigation.

8.1.2 Generate Cell and Portal Graph for Each Floor

Once we have the grid decomposition where 0 indicates free space and negative numbers indicate nonempty space (doors, stairs, obstacles, and doors), we start an iterative conquering process starting from the top left corner cell that is empty. We assign a positive number to this cell that will represent the room ID in the CPG, and then this ID is propagated using a breadth-first traversal. The propagation of the cell ID continues until the entire room is bounded by cells having either

TABLE 8.1: Algorithm to assign cell IDs to free space

```

Procedure find_cells (f)
cellID := 1
for f={0...maxFloors}
    for i={0..maxX}
        for j={0...maxZ}
            if grid[f][i][j] = free_space
                create_cell(cellID)
                cellID := cellID + 1
                flood_neighborhood (cellID,f,x,z)
            end_if
        end_for
    end_for
end_for

```

TABLE 8.2: Algorithm to propagate IDs

```

Procedure flood_neighborhood (cellID,f,x,z)
    if (grid[f][i][j] = free_space)
        if ((x=0) and (x<maxX) and (z=0) and (z<maxZ))
            flood_neighborhood(cellID,f,x-1,z)
            flood_neighborhood(cellID,f,x+1,z)
            flood_neighborhood(cellID,f,x,z-1)
            flood_neighborhood(cellID,f,x,z+1)
        end_if
    end_if

```

0 (wall) or -1 (door). The following procedures (Tables 8.1 and 8.2) show the algorithm in detail: where $cellID$ is the positive number representing the room ID, f is the floor number, and x and z are the coordinates of the cell in the 2D grid representation for that floor.

Figure 8.3 shows (a) the initial geometry of the building for one floor and (b) the corresponding grid decomposition where walls, doors, and obstacles have been identified. Figure 8.4 shows how the algorithm to identify cells propagates the IDs.

Once all the cells have been identified, we need to generate the CPG by joining the rooms through the doors. This is carried out by traversing the grid representation from left to right, top to bottom, looking for doors. When a door is found, a portal is created that will join the two cells appearing at both sides of the door. Two attractor points will be associated with each portal. These

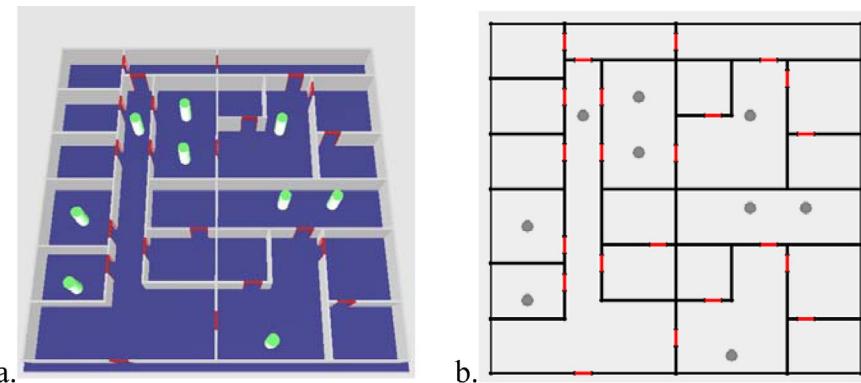


FIGURE 8.3: (a) Building geometry and (b) grid partition with walls, obstacles, and door.

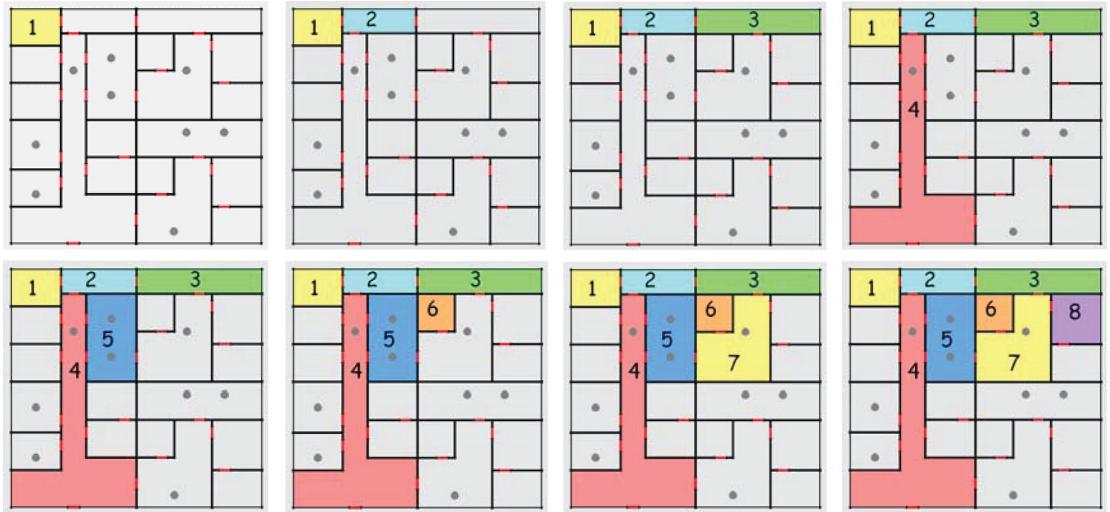


FIGURE 8.4: Eight first steps of the cell identification and ID propagation algorithm.

attractors are located at both sides of the door (Figure 8.7) and will be used for local motion of the autonomous agents as steering points.

Once all the doors have been detected and the portals created, we obtain the final CPG for the floor (Figure 8.5)

8.1.3 Identify Stairs and Link Floors Through New Cells

If we have a multistory building, then the algorithm will create the CPG for each floor, and then the stairs will be included as new cells that have portals with the lower and upper floors.

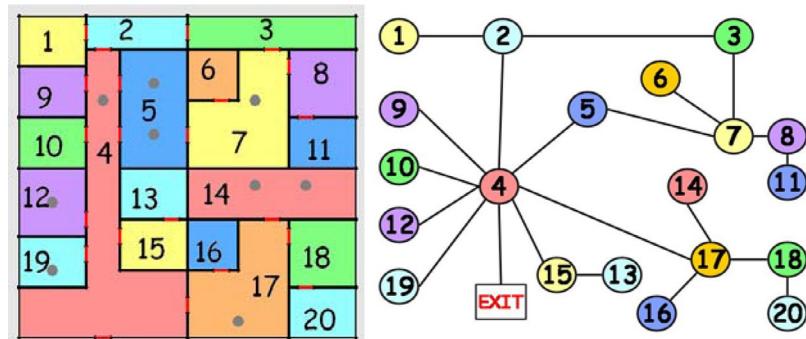


FIGURE 8.5: Cell and portal graph.

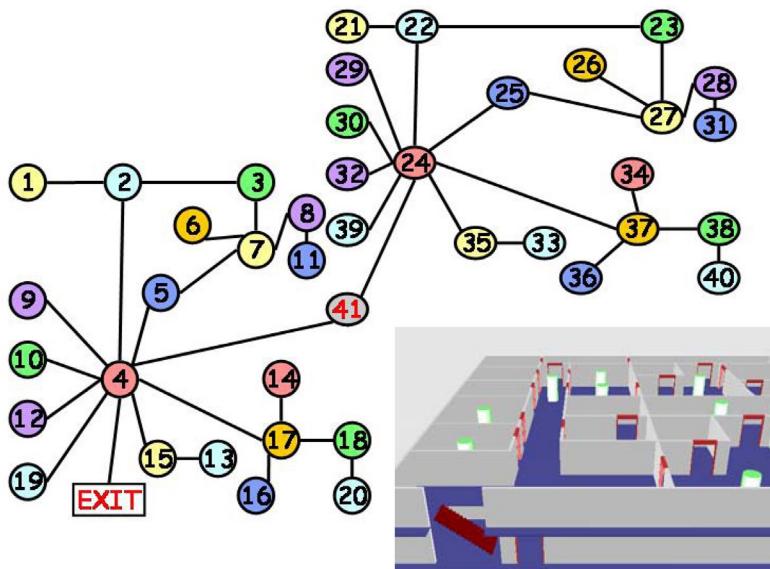


FIGURE 8.6: Cell and portal graph with stairwells.

After creating all the subgraphs that correspond to each floor of the building, the algorithm traverses the grid searching for stairwells. When a stairwell is found, a new cell and two new portals are created. One portal will link the stair cell with the lower floor and the other portal will link it to the upper floor.

Figure 8.6 shows the CPG of a two story building with one stairwell at the left bottom corner of the building. Cell 41 in the graph corresponds to the stair.

8.1.4 Identify and Store Walls

Once the CPG has been generated, we need to create the list of walls corresponding to each room. From the grid cell decomposition generated after creating the CPG, we will have information about walls (cells with ID = 0) and room IDs (since each free-space cell now contains the ID of the corresponding room in the CPG).

The grid is traversed sequentially from left to right, top to bottom, searching for walls along the X axis, and then from top to bottom, left to right, searching for walls along the Z axis. Walls are delimited by corner walls and doors. For each sequence of cells marked with 0's, between those delimiters, a wall in the environment will be created and assigned to the rooms at both sides of the wall. The result of the wall-finding algorithm can be observed in Figure 8.7. For clarity of the results, each of the walls has been colored to match the color of the cell. In the image, we can also

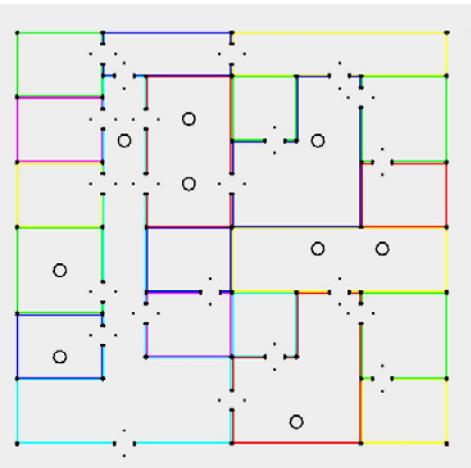


FIGURE 8.7: Walls assigned to each room and door attractors.

see the attractor points located centered in each doorway and displaced slightly (0.7 m) into each room.

8.1.5 Identify and Store Obstacles

Finally, the algorithm searches for obstacles in the environment and assigns them to the room based on the ID of the adjacent neighboring cells in the 2D grid decomposition. The obstacles can be observed in Figure 8.7.

8.1.6 Precalculating Data for Real-Time Simulation

The autonomous agents in the crowd need to interact with the environment to avoid obstacles and implement global planning. To achieve fast perception of the static obstacles in the environment, every cell has a list of walls and objects. For each wall, we store the equation of the plane that defines the wall, with the normal of that plane pointing toward the interior of the room and also the ending points of that wall segment. For every obstacle, we store its position and dimensions (obstacles are bounded by cylinder or oriented bounding boxes). The last elements to be stored are the two door attractors. These attractors are used to steer the agent in a natural manner when crossing portals.

When an agent needs to move from one location to another in a virtual room, it will query the room for the lists of walls and obstacles and then calculate collision detection with those objects, while moving from the previous attractor to the next attractor.

To perform global navigation, we also need to store information about paths within the building from each cell to each of the destination rooms or exits in the building. Each cell will contain one or more alternative paths to each destination or exit. During the simulation, agents will be able to query the room for different types of information, depending on their roles.

At each room, the information stored will contain for each destination room the sequence of rooms and doors and the overall distance in meters that correspond to the shortest path. Shortest path information is calculated by performing a breadth-first traversal starting at the destination cells and propagating to adjacent rooms. Figure 8.8 shows an example of the breadth first-traversal algorithm, indicating the propagation order through the rooms with decreasing intensity of color. The example shows distances to the exit door.

The breadth-first traversal starts from the room that can be a destination room during the simulation and pushes it in a queue. When we simulate evacuation scenarios, we only consider as destination rooms the ones with an exit, but for more general simulations where agents could go to any room in the environment to achieve a given goal, then we need to run this algorithm for all the rooms in our environment.

The algorithm proceeds iteratively, popping an unvisited node from the queue, visits it, marks it as visited, adds its neighbors to the queue, and repeats it until all the nodes in the graph have been visited. As the traversal advances, it updates the distances from each portal to the destination, and if a cell is reached through several doors, then the algorithm compares the previous shortest distance stored against the new one, and only if the new distance is shorter, the node will be pushed in the

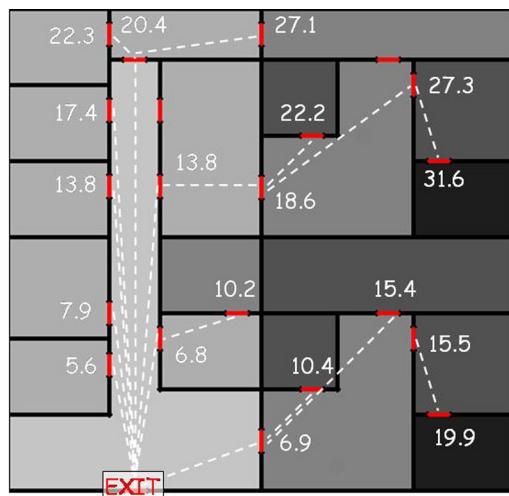


FIGURE 8.8: Shortest paths propagation and distances.

list. When two cells are connected through several doors, the one with the minimum global distance to the destination is chosen as part of the shortest path. The details of the algorithm are shown in Table 8.3.

If only shortest paths were stored, then when an agent would find the desired path toward a destination blocked, it would need to perform some search in the environment to find a different route. Agents with limited knowledge about the environment may not be able to find in their mental maps an alternative route and therefore need to explore the environment. But for those agents with complete knowledge about the environment, a search in the complete CPG representing the building should be feasible. To speed up the simulation time, we calculate this information offline and store it in the environment. This represents the knowledge of the agents and can be shared among those with trained roles. The alternative paths are calculated by modifying slightly the previous algorithm. In the search for alternative paths, when a node is reached through a different route, first of all, the algorithm checks that the current node is not already contained in the path toward the destination to

TABLE 8.3: Algorithm for shortest paths propagation

```

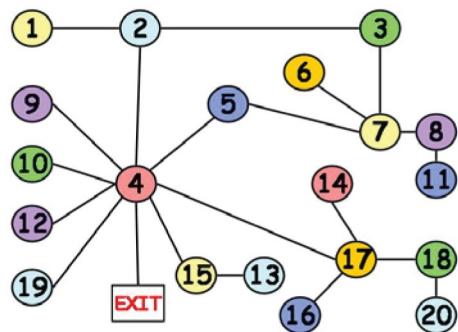
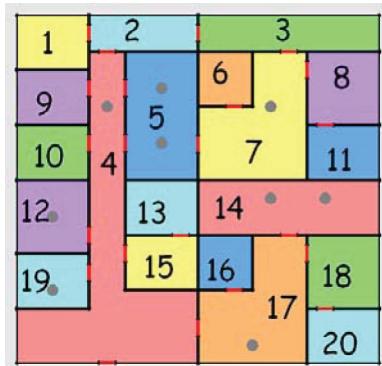
Procedure propagate_shortest_paths
for each destination node E
    push_list(E)
    while list non empty V=pop_list
        mark V as visited
        for each neighbor room N
            if not discovered N
                if N already had a path stored then
                    if newGlobalDist < prevGlobalDist
                        modify stored shortest path
                        push_list(E)
                    end_if
                else
                    push_list(E)
                end_if
            end_if
        end_for
    end_while
end_for

```

avoid cycles. Then, if it is not already in the path, this new path is stored in the node as an alternative route and propagated to all the neighboring nodes except for the one from which it arrived.

The algorithm finishes when all the possible alternative routes have been propagated and stored. Then, the list of alternative paths is ordered by distances, with the aim of speeding up the query process from the agents during simulation time. Table 8.4 shows the alternative paths stored for the current building.

TABLE 8.4: Algorithm for shortest paths propagation



NODE ORIGIN	SHORTEST PATH	ALTERNATIVE PATHS	NODE ORIGIN	SHORTEST PATH	ALTERNATIVE PATHS
1	{2,4}	{2,3,7,5,4}	11	{8,7,5,4}	{8,7,3,2,4}
2	{4}	{3,7,5,4}	12	{4}	
3	{2,4}	{7,5,4}	13	{15,4}	
4	{}		14	{17,4}	
5	{4}	{7,3,2,4}	15	{4}	
6	{7,5,4}	{7,3,2,4}	16	{17,4}	
7	{5,4}	{3,2,4}	17	{4}	
8	{7,5,4}	{7,3,2,4}	18	{17,4}	
9	{4}		19	{4}	
10	{4}		20	{18,17,4}	

8.2 LAYOUT OF ENVIRONMENT

The environment is not complete with just the layout of rooms and hallways or buildings and streets. Environments also need furniture, vehicles, street signs, or fire hydrants. These objects need to be placed appropriately in the environment, and they also need to be labeled with semantic data that will allow the characters of the simulation to interact with them.

At the basic level, the bounding volume of objects is required for collision detection (see Section 5.5.1). For characters to interact with objects, other data is required. For example, knowing the front, back, and sides of objects helps determine where characters should be placed to interact with them. Other more specialized regions, such as the seat of a chair, provide information needed to automate other actions (e.g., sitting) (Coyne and Sproat 2001). Some of these features, such as holes and handles might be able to be detected using computer graphics techniques. Others may have to be hand tagged, but this may be feasible since objects tend to be reused often. If the scenario requires more sophisticated planning, information about the purpose and capabilities of objects should be provided (Bindiganavale et al. 2000).

The placement of objects in the environment can, of course, be done by hand either interactively or by writing coordinates in a file. It might also be possible to tag rooms with information similar to that provided for objects, including likely regions for the placement of furniture objects. Figure 8.9 depicts a line of offices with similar furniture placements. If one were to lay out the basic regions of an office, including the furniture placement relative to other regions such as the door, even with different room dimensions, furniture layouts for new rooms might automatically be constructed merely from basic geometric features (e.g., door placement) and its type (e.g., office). Certainly statistical deviations could be used to create natural variations.

8.3 CHARACTER PROFILES

Another aspect of setting up a simulation scenario is creating the characters. In many current simulations, the characters are homogeneous or have a few random variations. We would like to have a much more diverse crowd with meaningful variations. Certainly geometric models of the characters need to be created or acquired. There is little that can be done to shortcut this necessity aside from minor statistical variations in form and various attribute changes. Here, however, we are more concerned with character profiles that link to activities.

One option is to associate behaviors and activities with roles. Roles are expected behavior patterns associated with social status or occupations. By linking actions and their contexts to characters, these roles can be used to create more naturally heterogeneous simulations. Potentially, roles could even be statistically linked with types of buildings or rooms to create populations more automatically. If, for example, a building has several classrooms, a large percentage of the population could be created as students and a small percentage as instructors. These might be a good starting



FIGURE 8.9: Office environment with sparse furniture.

point for a simulation creator. From here, the designer could change the percentages or age ranges and sprinkle in a few other roles such as janitors and cooks.

Some roles have inherent status hierarchies: instructor/student, parent/child, and supervisor/worker. As discussed in Section 3.4, status relationships affect that way that people interact both verbally and non-verbally. Depicting these variations in interactions can add to the realism of the simulation. Status relationships not implicit in the roles can be assigned randomly or by the scenario creator.

8.4 CREATING GROUPS

People have relationships outside of status relationships. These relationships also affect the way that people interact with each other. Even actions as simple as passing in a hallway are affected by relationship. People who know each other tend to at least acknowledge each other. Friends may stop to chat. Enemies may purposefully ignore each other.

In a crowded environment, people in some way associated with each other (e.g., co-workers, family, friends) tend to clump together and navigate the environment together. Rarely are crowds composed solely of individuals. They more often contain groups of varying sizes. Groups may enter the environment together or form organically as they traverse. Generally, these groups continuously evolve, gaining and losing members, and separating and reforming as the group navigates obstacles.

How can we create simulations depicting these groups? Certainly we could randomly choose agents to pull together into groups. For many simulations where the mass of the agents are just background characters, this might work fine. For others, observers may note odd associations or the lack of expected associations. We could make an effort to create and represent different relationships by hand, but for large simulations, this task would be overwhelming. Instead, it may work to use location demographics. As noted above, groups tend to form from people who know one another from work, home, school or some other mutual activity. Demographic data for the area or areas being simulated might be used to create and associate a reasonable population to locations. For example, an apartment is assigned a woman and two school-aged children. These three characters then become a family group and may be drawn together when they are in the same area. The same woman might also be assigned to an office building and the children to a school. These location associations would add them to groups of co-workers and classmates that would stochastically also tend to form groups when in proximity.

Certainly human relationships and groups are more complicated than described here and much more work could be done. This may, however, provide a basis from which to work. People who cross paths could also randomly form friendships and therefore display more friendly behavior and tend to group. These friendships and relationships may also fade from time to time. Specific groups may also be hard-coded into the simulation by its creator to serve an important role in the simulation.

8.5 CONSTRUCTING ACTIONS

In Section 7.2.3, we introduced four action types, all of which can be represented as Parameterized Action Representations (PARs). Ultimately, we envision an Actionary full of these actions that can be pulled from the database either automatically in accordance with the character roles or, if

required, by the simulation creator through a simple interface. There may, however, always be more actions and associations required for new simulations, and therefore, there needs to be a way to create new actions and edit existing ones.

Creating parameterized actions that are general enough to be used in many scenarios but detailed enough to fit into the contexts appropriately can be a difficult knowledge engineering problem. Using the inheritance provided by the Actionary hierarchy eases this burden. By properly placing new actions in the tree, many of their parameters will be inherited from their parent and need not be set manually. The action author can then focus on just those parameters that differentiate it from existing actions. The Actionary is stored as a MySQL database, and as such, any interface to the database software can be used to enter the necessary data. Custom graphical user interfaces can also be created to help an action creator access and enter data. These actions will then need to be linked to their associated motion generator or animation clip. This is currently done by providing a pointer to the function that initiates the motion.

Creating actions for scenarios (i.e., iPARs) entails choosing the actions from the Actionary, filling in the necessary parameters that link them to the characters and environment, and where necessary indicating the start times or orderings of the actions. Much of this can be done automatically through character roles and the automated code based on the parameters of the four action types.

Scheduled actions need to have their start time, duration, location, and participants specified. The most natural interface for specifying such information may be a calendar program. Such programs are used by people everyday and are designed to be easy to use. We have explored using Microsoft Outlook as an interface for specifying scheduled actions. Microsoft Outlook allows for numerous calendars, which for our purposes represent scheduled actions for the different roles of characters. Certainly individual characters could also be scheduled through this interface. A commercial software package called [GeniusConnect \(2007\)](#) provides the necessary connection to the Actionary, a MySQL database.

Reactive actions are PAR actions that are associated with states of the world. When the specified state of the world is detected, the action is performed by every character or characters who have been assigned the reaction. States may include properties or states of objects and agents in the world. They may also be spatial configurations such as two agents being near one another. These spatial configurations require recognizers. An interface for constructing such actions may simply include lists of agents, objects, actions, properties, states, and prepositions. The rules for reaction actions could be created just by choosing the appropriate components.

An interface for creating opportunistic actions could work similarly. This interface, however, would include a list of needs, fields for specifying growth and decay rates, and then the actions or objects needed to fulfill the needs. It would also be possible to include data that allows the

growth rates and decay rates to vary depending on the actions or the objects or even the agents themselves.

Aleatoric actions can be specified as a special type of complex PAR. The choices of action are specified as sub-actions and their distribution as means and standard deviations. Certainly more complex functions for choosing between the actions could be provided if desired. Because the sub-actions of a complex PAR are also just PAR actions, these actions would already have been specified and stored in the Actionary.

8.6 REFINING THE SIMULATION

Once a simulation has started, there will inevitably be modifications desired by the simulation author. A tool that does not provide this capability is not a very useful tool. The aim of the CAROSA framework is not only to provide a mechanism for quickly creating realistic scenarios with many characters, but also provide the author with enough control to refine the scenario to fit future needs. This might include changes to the environment or modifying the characters or perhaps even scripting the behaviors of some of the characters.

8.6.1 Effects of Changes to the Environment

Changing the structure of the building or city that is the backdrop for the simulation could affect the characters' ability to successfully navigate it. Adding or removing a room or moving a doorway could potentially require reformulating the data required by the navigation processes. However, the structure that we use, a CPG (Section 2.6.1), is created automatically from the environment file. As a result, changes to the structure of the environment require no additional effort from the simulation creator.

The environment may also be modified by changing the objects found there. One object, such as a desk or a fire hydrant, may be swapped for another. An object may also be added, removed, or simply moved. This has two potential impacts on the successful execution of the simulation. The first is the ability of the characters to navigate through the environment without colliding with objects. In our framework, collision detection with objects is based on their bounding volume, which is computed as the objects are loaded, and their current position (Section 8.1.5). Thus, changes to the objects in the environment do not require any additional specification in the collision detection.

The second potential impact arises from the objects' roles in character actions. If a character is assigned a task of photocopying a report, a photocopier needs to be available. Simply moving objects to new locations will not require any additional data entry from the scenario creator. The positions of the objects are acquired automatically during the simulation and planned for appropriately. Problems may, however, arise if objects necessary to actions are not present in the environment. If

there is no photocopier present in the environment, the photocopying action will fail. These failures are captured by the CAROSA framework. Presently, the failures are ignored during the simulation and the next actions are attempted. The system does, however, provide support for failure reporting, including the cause of the failures. This would help the scenario creator to fix the failure if desired. It is important that the simulation framework does not fail entirely even as a result of user error. We cannot guarantee that the resulting simulation will meet the desires of the author, but it should not fail completely.

We also see a potential for automated processes that alter the environment during the simulation. Certainly, hazards such as smoke, fire, or blocked hallways are one set of possible changes (Section 5.5.6). These changes would impact the navigation of the characters through the environment and depending on their nature could also impact their decision-making (Section 5.5.7). Hazards such as fire could be programmed to start in user-specified or random locations and spread according to known patterns (McGrattan et al. 2008). There are, however, other potentially interesting processes that could be included to enrich the environment. In a building environment, systems and equipment could be programmed to fail either randomly or as scheduled by the simulation creator. If an elevator is suddenly broken, the characters would be forced to use the stairs. If a photocopier is broken, they would need to locate another or notify service personnel.

In a city environment, a process simulator might be used to simulate traffic and traffic signals. The entire operation of a city can be impacted by the ability or inability of its inhabitants to travel through it. Vehicles and signals would be PAR objects and as such could be interact with all other objects. We can envision characters waiting for buses or taxis and perhaps even forming relationships as they do so (Section 3.3).

8.6.2 Modifying Roles

After seeing an initial run of a simulation, the author may also wish to change the characters. Changing the number of characters in each role is as simple as changing the number in the quantity field. An author may also wish to redefine the roles and other individual differences of the characters. Creating new roles would require the same effort as creating the original simulation roles (Section 8.3). Roles may also be modified by changing the associated actions or their placement in the status hierarchy.

Character roles are linked to different types of actions. In particular, scheduled and aleatoric actions have a direct impact on the characters' association with roles. A simulation creator can change the scheduled actions for a role simply by changing their placement on the calendar for that role. They may change an entry on the calendar to another action or change its time, duration, or location. In the CAROSA framework, default actions are often represented as aleatoric actions

(Section 7.2.3). Here the action chosen and its duration are based on a distribution for all of the associated actions (sub-actions). If the scenario creator would like to see the characters of a certain role performing an action more or less than was seen previously, only the distribution needs to be altered accordingly.

Other actions are less directly linked to the roles of the characters. These actions are primarily associated with character needs and reactions. In the CAROSA framework, needs are easy to create and modify. If one wanted to create characters that appear a bit neurotic, they might create an *energy* need that is fulfilled by drinking coffee. They would then specify a relatively large decay rate. This would result in the characters constantly running to get coffee as their need for energy would be felt often. Decreasing the decay rate would make them feel the need less often and therefore drink coffee less often. Needs do not need to correlate with an actual human need. For example, an author could create a *fidget* need that is fulfilled by clicking on and off marker caps. Altering and creating needs can be done through a simple interface as described earlier.

Reactions could also be modified using an interface similar to the one used to create them. A user could modify either the impetus for the reaction or the reaction itself. This might include defining an aleatoric action as the reaction. In this way, characters would not always react to an event in the same way. Their reactions would vary from one instance to the next. No action could also be included as one of the aleatoric subactions, resulting in no reaction taking place from time to time, which is natural behavior.

8.6.3 Scripting Characters

While the CAROSA framework is designed to create realistic, but general behaviors for many characters, there will likely be a few characters that the simulation author would like to control more precisely and in more detail. Toward this end, an author could construct calendars for the characters needing scripted behaviors. Our current calendar interface, Microsoft Outlook, provides temporal resolution of 1 minute, so through this interface a simulation author could give characters different actions for every minute of the day. We believe that for many scenarios, this will be more than adequate, particularly for background characters.

There may, however, be circumstances where even finer temporal resolution is required. In these cases, character behavior could be created through connections to a complex PAR action. For example, an author may want to focus on the actions of a particular character that will have an impact on the purpose of the simulation, by choosing actions from the Actionary and combine them into a PAR constraint graph. A PAR constraint graph indicates the ordering of the actions in a complex PAR (Bindiganavale et al. 2000).



CHAPTER 9

Evaluating Crowds

One of the most difficult aspects of doing research in crowd simulation is evaluation. The real world and its real characters are very complex, making it difficult to compare a simulation to events that might happen in reality. Another option for evaluation is performing user studies to try to determine if the desired qualities for the simulation have been met. Finally, technology may aid in this endeavor. In this chapter, we will outline a few methods of evaluating crowd simulations.

9.1 FEATURE COMPARISON

An issue with doing a feature comparison of crowd simulation models is determining what features are priorities for the application. To aid in this determination, features might be segregated into low, middle, and high levels. Low-level features would include the quality of the graphics and animations. Middle-level features would include more behavioral aspects of the simulation, such as navigation ability and physical-level agent interactions. High-level features would include more deliberative processes, functional crowds, and behavioral heterogeneity.

9.1.1 Low-Level Features

There are a number of low-level features that may vary in importance and therefore quality from one simulation to another. First, there are the graphical models of the characters, objects, and environment. These may vary from simple line figures and primitive shapes, to cartoon characters, to photorealistic figures. The quality of the characters will have an impact on the frame rate that can be achieved for any given number of characters and the perceived “human-ness” of other aspects of the simulation (Mori 1970). For example, the level of detail and visual quality of the character models set an expectation about the realism of their behaviors. For even marginally realistic characters, foot skate (sliding) becomes a distraction to most who observe the simulation. In crowd simulation, there are rarely any actual interactions with objects. Characters may push a button on an object or perhaps carry an object through the environment, but they very rarely operate a device at any level of detail. Crowd simulations have been more concerned with performing collision detection and recovery than object interactions. This may change as technology progresses.

In addition to the quality of animation, animation transition quality can have a large impact on the effectiveness of the simulation. Popping from one animation clip to another or other inadequate transitions often detract from simulations. Finally, rendering quality may be considered a low-level feature of a crowd simulator. Can the crowd simulator render shadows or lighting effects or special effects like fire and smoke? These elements can have an impact on the effectiveness of the simulation.

9.1.2 Middle-Level Features

Middle-level features build on the lower-level graphics and animation features to develop more sophisticated behaviors. These features might include how well the characters can navigate through the environment and how well they react to dynamic environments. If a passageway is blocked, can the characters discover an alternate route? If there are vehicles on the roads, can they avoid being hit by them? A part of this process is the characters' need to have fast perception of their environment. Perception can be done in several ways with different levels of accuracy and realism. Methods include using techniques from computer vision to more realistically sample what the characters might see or else simply assume that characters have complete knowledge of their entire environment or a limited area near them.

There are numerous other features that have emerged from both the crowd simulation community and the crowd psychology community. Some of these include natural bidirectional flow and overtaking, emergent queuing behavior, realistic pushing behavior, falling agents and other obstacles, and panic propagation. Researchers also strive to eliminate side effects of their implementations, such as eliminating shaking behavior. Ideally, whatever the method chosen, the simulator should be robust. Appropriate and consistent behavior should result whatever the environment, overall context, or number of agents.

9.1.3 High-Level Features

In some ways, high-level features are used to create crowds more specific to the purpose of the simulation. An author should be able to create behaviors typical to the scenario desired. This includes both deliberative and reactive actions. For many scenarios, characters in the crowd need to appear purposeful or functional. A viewer should be able to discern character goals. At the same time, the characters should be able to react to the world around them and adhere to the constraints and needs of real humans.

Furthermore, these various types of actions and behaviors should be allocated to the characters in a manner that results in a heterogeneous yet plausible crowd. Assigning actions of the characters randomly would result in a chaotic simulation. Allocating actions according to character traits should create a more realistic simulation. There are numerous human traits as studied in psychology,

sociology, physiology, and many other disciplines. The challenge in finding a reasonable set of traits to create rich characters but not overwhelm scenario creation continues.

9.1.4 Summary

As discussed, there are many possible crowd simulation features. Table 2.2 compares some of the major systems based on just a few features. Constructing a simulator that includes them all can be challenging, but the biggest challenge is also making sure that the simulator is robust and modifiable. It should be able to handle many scenarios and crowd sizes and it should be able to be tweaked for specialized scenarios. Comparison of speed vs. quantity is also a consideration. How many characters can the system simulate while maintaining a viable frame rate? Not all simulations need to be real-time or even interactive, but even offline systems should be as efficient as possible to facilitate the necessary testing and scenario exploration.

9.2 COMPARISON TO REAL-WORLD DATA

Naturally, one method of evaluation for crowd simulations is to compare it with real-world data. Although such data may not encompass all of the features we wish to analyze, there may be enough data to do a baseline comparison.

9.2.1 Sensor Data

The Mitsubishi Electric Research Laboratories recently recorded a year's worth of motion sensor data from their offices (Wren et al. 2007). Motion detectors were placed in the public spaces of their offices and recorded data 24 hours a day. This data is limited to activations of motion sensors at 1-s resolution but does provide a cursory sense of the flows of people. These flows could be compared with simulation flows and temporal densities (Sunshine-Hill et al. 2007).

The computer vision community has also done research in tracking the flows of crowds and even tracking individuals (Kang-Hoon et al. 2007a, 2007b). This type of data can help to illuminate how people navigate an environment and cluster together. Other behavior, such as the type of actions performed and their frequency, could also be obtained from video clips. Missing from such footage are the character traits and goals. We may be able to determine what the characters are doing and where, but we still do not know why. In any case, it may provide a basis for comparison for simulations.

9.2.2 Action Statistics

Evaluating whether the characters in a simulation are performing appropriate actions in appropriate proportions can be done by comparing the frequencies with the distributions acquired by various

communities. For example, studies have been done in management science to determine what percentage of time office workers spend on various tasks (Kurke and Aldrich 1983). The U.S. Department of Labor also has a Bureau of Labor and Statistics, which provides some information about the actions performed by various types of workers (last visited February 2008). Such resources can provide information on both the demographics of the area to be simulated as well as the activities of the population. This data could either be used to set up the simulation or for verification.

9.2.3 Validation Through the Society of Fire Protection Engineers Guide

Guaranteeing safe egress is an important design requirement for any building size. There has been considerable work trying to simulate human movement to be able to study egress, but validation of those models is still a challenge. Some researchers have undertaken evacuation drills with humans to compare the results against the simulation (Still 2000). Others are studying how to use tracking data from real crowd movement to create data-driven methods by recording videos of real crowds, extracting the 2D moving trajectories of each pedestrian and then learn an agent model from observed trajectories (Kang-Hoon et al. 2007; Lerner et al. 2007). The final goal of fully validating a crowd model in terms of human decision-making, psychological elements affecting human behavior, individuals' knowledge of the building affecting the simulation, or signaling and communication having an impact on evacuation times has not been successfully carried out yet. So although there is still much work that needs to be done before being able to move in that direction, currently we can use quantitative data such as that provided by the Society of Fire Protection Engineers handbook (SFPE 2003) to validate the accuracy of human movement for an evacuation application. Current quantitative data can be used to validate egress (overall evacuation time, flow rates, and densities). The SFPE guide is generally used by fire protection engineers to guarantee the evacuation of a building within a required time. It is important to consider though that it is based on a hydraulic model and therefore does not include the large variety of behaviors that humans can exhibit.

The Society of Fire Protection Engineers Model. The SFPE Engineering Guide to Human Behavior in Fire presents a hydraulic model to calculate human movement. The required evacuation time (T_{revac}) is defined as the overall time from the moment the fire starts until the entire building is evacuated. T_{revac} is defined as:

$$T_{\text{revac}} = t_d + t_a + t_p + t_i + t_e,$$

where t_d is the detection time (time passed since the beginning of the fire until it is detected), t_a is the alarm time (time passed since the detection of the fire until the occupants of the building are alerted), t_p is the perception time (time passed until the building occupants perceive the fire), t_i is

the interpretation time (time passed since the fire perception and decision-making, until the actual evacuation starts), and t_e is the evacuation time (time passed since the evacuation starts until everybody has left the building).

Accurately calculating T_{revac} is a difficult problem, since it depends on different elements that can greatly differ between scenarios, based on the fire alarm systems, building knowledge of the occupants, individuals' decision making, etc. For this reason, the time that is usually calculated and for which we will carry out the validation of our model is t_e (time passed since occupants start moving until the building has been completely evacuated).

According to the SFPE handbook, speed depends on the density and a constant k , which varies depending on whether we are calculating speeds for a corridor, ramp, or staircases (k values can be found in the SFPE handbook). The speed vs. density relationship is defined as:

$$v \text{ (m/s)} = \begin{cases} 0.85 \cdot k & \text{if } \delta \leq 0.54 \text{ pers/m}^2 \\ k - 0.266 \cdot k \cdot \delta & \text{if } \delta > 0.54 \text{ pers/m}^2 \end{cases}$$

For density values below 0.54 pers/m², people move freely and independently of other occupants' speed of movement.

The specific flow rate is the flow of occupants through a point of the evacuation route per width unit and time. The flow rate vs. density curve stated by the SFPE follows the equation:

$$\Phi = (1 - 0.266 \cdot \delta) \cdot k \cdot \delta \quad (\text{pers/m/s})$$

This curve indicates that as the density increases, the flow rate (persons per meter per second) increases, but after a certain density, the flow rate starts decreasing because of the bottleneck that appears in the doorway when many people are pushing to cross at the same time. We can therefore ensure that these equations are satisfied by our simulation so that it closely matches real crowds' densities, speeds, and flow rates.

9.3 USER EVALUATIONS

One of the more common techniques for evaluating simulators is performing user studies. Studies are set up to try to determine if the objectives of the simulations are being met or if the viewers are being distracted by other elements of the simulation. These evaluations may include side-by-side comparisons of different models or techniques. This may be used to determine the effectiveness of different features or implementations of the simulation. Ablation studies can also be used to determine what features add to the simulation and which are simply not noticed. To obtain more hard data, eye trackers might be considered. This would help to underline what is capturing viewers' attention and what is not being seen.

Another important aspect of crowd simulation evaluation is determining the amount of effort needed to create and modify the simulations. User studies can also help to make this determination. By asking subjects to create a new simulation or modify an existing one and then answer a few questions about the process, we may gain insight about authoring effort.

9.4 PRESENCE IN VIRTUAL WORLDS

Large animated groups of autonomous agents are being widely used for computer graphics applications, video games, training, and education. An important practical problem in this research lies in how to validate the models. There has been considerable work done in validating egress for evacuation simulations based on the literature on human movement behavior, but there is no quantitative data on how to validate human behavior when it comes to decision-making in this context.

Controlled experiments are therefore needed where human behaviors in response to different crowd models can be tested. For example, during a fire, which exit routes would people select? If there are leaders giving instructions, how many people would follow them? If there are strangers communicating information, how much would others trust them? What motion paths are taken and what movements are made by an individual in a crowd?

These experiments are usually either difficult to replicate in real life or simply impossible to run in the first place (i.e., actual fire evacuation). Experiments in virtual environments (VEs) could be invaluable for gathering the behavioral information necessary to improve current crowd simulation models and consequently experimentally validate them.

To gather accurate information, it is essential to achieve *presence* so that a subject immersed in the virtual experiment will behave as close as possible to real life (Sanchez-Vives and Slater 2005). *Presence* is described as the extent to which people respond realistically to virtual events and situations. Responding realistically implies realism at many levels, ranging from physiological through behavioral, emotional, and cognitive behaviors (Sanchez-Vives and Slater 2005).

An accepted method of measuring *presence* has yet to be agreed upon. Classic *presence* work relied on questionnaires, but since questionnaires depend entirely on users' subjective views of their experience (Usoh et al. 2000), researchers found it necessary to develop other supplementary methods (Freeman et al. 1999). Those methods include behavioral measurements (social and postural responses, etc.) (Freeman et al. 2000; Bailenson et al. 2003), physiological measurements (galvanic skin response, heart rate, etc.) (Slater et al. 2006), task performance measurements (completion times, error rates, etc.) (Basdogan et al. 2000), and counting breaks in *presence* (Slater and Steed 2000).

Using one or more measuring methods, a number of findings have been published about *presence*:

- Being able to physically manipulate objects (Schubert et al. 2001) and communicate with virtual humans in a VE increases a sense of *presence* (Slater et al. 2006).

- Unnatural interactions with the VE, such as using a joystick to maneuver, can reduce the sense of *presence* when compared with techniques that resemble real-life navigation such as “walking in place” (Slater et al. 1995).
- Breaks in *presence* (Slater et al. 2006) have been used to count the transitions from the virtual to the real world. These transitions can be triggered by occurrences such as bumping into a wall in an immersive environment, tripping over cables, and whiteouts (Slater and Steed 2000).

These findings are important to consider when designing a realistic crowd simulation model. Although crowd simulation validation currently exists for safe egress during evacuation by using engineering guidelines, there has yet to be any validation based on human behavior during decision making in more dangerous situations. With the knowledge that people act in a VE as if they are in a real-world situation when they experience a high sense of *presence*, we believe that a good crowd simulation model should promote this sense of *presence*. Once we have crowds that provide a high sense of *presence*, we can confidently run simulations to study human behavior and use the resulting data both to validate and improve current models.

Our interest lies in differentiating external crowd motion features from internal or egocentric features. The computer animation community has been primarily concerned with the former, as a good simulation will produce crowd movements that appear realistic to an outside observer. Egocentric features, on the other hand, are about what an active participant in the crowd simulation would perceive visually or kinesthetically and thus provide computable measures of *presence* for the subject.

This section first surveys the different crowd simulation models in the literature. We discuss egocentric features that may affect *presence* and then qualitatively analyze which of these features may break or increase *presence*. Finally, we present a pilot experiment and the results obtained.

9.4.1 Important Egocentric Features

For the purpose of this study, we focus on four models that have been widely used for crowd simulation [social forces (Helbing et al. 2000), rule-based (Reynolds 1987; Reynolds 1999), and cellular automata (CA; Kirchner et al. 2003)] and the hybrid approach [HiDAC, Pelechano et al. 2007)] explained in this book.

The main egocentric features that we can extract from these crowd models, which we believe are significant factors influencing *presence* in VEs, are shaking, discrete/continuous movement, overlapping, communication, and pushing. We will now describe how each of these features is present or absent in each of the four models used for our study (a summary appears in Table 9.1).

- Shaking: how much the agents appear to vibrate while trying to move. Force-based models are unstable and thus the position of each agent is slightly modified for each time step, which yields the illusion of agents shaking continuously. In contrast, CA or rule-based models do not suffer from this artifact, and HiDAC, although built on top of a forces model, corrects this behavior through rules.
- Discrete/continuous movement: how the agent moves from one position to another and whether it is discretized or continuous in space. In CA models, agents move between discrete adjacent cells in one time step, limiting turn direction options. The other models do not discretize the space and therefore allow the agent to move within continuous space.
- Overlapping: whether overlapping with other agents can occur. This effect can be observed in some rule-based models where only collision avoidance is performed but not collision response. Later versions of these models apply stopping rules to prevent overlapping ([Shao and Terzopoulos 2005](#)). Although CA models avoid collisions by not allowing agents to move to occupied cells, they allow agents to seemingly cross through each other. This occurs when two agents simultaneously wish to move into each other's occupied cells. Because the cells are occupied, they choose instead to move diagonally to the empty cells next to the occupied ones, resulting in the trajectories of the agents crossing each other within one simulation step. Social forces models and HiDAC do perform collision detection and response to minimize overlapping.
- Communication: represents the ability of the agents to exchange information about the VE ([Pelechano and Badler 2006](#)). The original social forces, rule-based, and CA models do not include this feature. HiDAC as well as some later versions of rule-based models

TABLE 9.1: Simulation methodology impact on *presence*

	SOCIAL FORCES	RULE-BASED	CA	HIDAC
Shaking avoidance	-	+	+	+
Continuous movement	+	+	-	+
Overlapping avoidance	+	*	-	+
Communication	-	*	-	+
Pushing	+	-	-	+

+ indicates that the model readily admits this feature; -, it does not

*Later versions of this model have built these features on top of the original model.

incorporate communication as a way of sharing information about the environment and giving instructions to other members of the crowd.

- Pushing: having physical contact between the agents' bodies. If this interaction occurs, then one agent should be able to push others through the crowd. This feature is exhibited by social forces models and HiDAC, but it is not performed in rule-based or CA models.

9.4.2 Experimental Evidence From the Literature

There have been many experiments to date studying which elements of a VE could enhance or reduce *presence*.

[Slater et al. \(2006\)](#) discovered that when a whiteout occurs while a participant is immersed in a VE, there is a break in *presence*. This effect occurs, for example, if while navigating a VE the participant walks through a virtual object or agent. The observed result would be as if the VE had suddenly disappeared. Based on these results, we conclude that it is essential there be no overlapping.

According to [Schubert et al. \(2001\)](#), “*Presence* is observable when people interact in and with a virtual world as if they were there, when they grasp for virtual objects or develop fear of virtual cliffs.” Interaction means “the manipulation of objects and the influence on agents.” Accordingly we conclude that to enhance the sense of *presence*, a participant must be able to manipulate virtual objects. One way a participant could feel as if they were affecting the virtual world would be by pushing other agents they came into contact with.

Another way of interacting that increases the sense of *presence* is through communication with the virtual agents. Some studies show that the heart rate of a participant increases when spoken to by a virtual agent ([Slater et al. 2006](#)).

Studies show that discontinuous movement or jerkiness reduces *presence*. Jerkiness can be observed when, for example, the VE suffers from low frame rate. As [Barfield and Hendrix \(1995\)](#) concluded, “The subjective report of *presence* within the VE was significantly less using an update rate of 5 and 10 Hz when compared to update rates of 20 and 25 Hz.” Therefore we can expect that crowd models suffering from agents shaking, jerking back-and-forth or appearing to move between separated discrete positions will likewise diminish the participant’s sense of *presence*.

9.4.3 Pilot Experiment

For this work, we carried out a pilot experiment to closely study the behavior of people interacting with a virtual crowd ([Pelechano, et al. 2008](#)).

For the experiment, we created a virtual scenario simulating a cocktail party. At the party were virtual partygoers who walked around “mingling” with others through nonverbal communication and gestures. After a specified time, a bell rang and the virtual agents calmly exited the party.

The virtual agents were rendered using [Cal3D \(2008\)](#) and they had several animations assigned, including different walking styles that could be blended smoothly, and a set of idle and gesturing animation clips that could be used when agents stop walking or gather around a table.

Figure 9.1 shows a crowd of virtual agents interacting during a cocktail party. People gather around the tables to eat and engage in (nonverbal) conversation with others. On the right, we can observe a close-up of one of the tables.

The scenario used for all four crowd models was a large room with round tables distributed so that virtual agents could move around and stop around any of them to engage in nonverbal conversation with other members of the crowd. When the bell rings, they all start walking calmly toward the door with the exit sign above it. As the participant will walk within the crowd as another agent, individuals will react depending on the crowd model being used [i.e., perform collision avoidance (in rule-based and HiDAC), respond to interactions such as being pushed (in HiDAC and social forces), not occupy the same cell (in CA), etc.]

Setup. Participants were members of a university community. They were recruited throughout the campus by posting signs. Each volunteer subject was randomly assigned to a group when they arrived.

The stimulus was a 3D model of a building, populated with virtual characters and furniture, and presented using an eMagin Z800 3DVisor head-mounted display (with a resolution of 800×600 , field of view of 40° , and refresh rate of 60 Hz). In addition, participants wore four head sensors that are part of the ReActor2 suit, an optical motion capture system from Ascension Technology. The head sensors were used to determine where participants were looking and located in the VEs.



FIGURE 9.1: Virtual crowd in a cocktail party ([Pelechano, Stocker et al. 2008](#)).

Task. Each subject was placed in the same VE with the same virtual characters, varying only in the crowd model implemented (social forces, rule-based, CA, HiDAC) according to their group. They were told that the purpose of the research was to assess the validity of the VE that we had created. The potential risks of the experiment — eyestrain and nausea — were explained to them, and they were told that they could withdraw at any time. The experimental protocol was formally approved by our institution's IRB.

The subject's first experience in the virtual world was to locate three objects in the environment while the virtual characters in the environment were stationary. This was used as a training phase to get them comfortable with moving through the environment, but not influenced by a particular crowd model.

The subject was then assigned the task of walking around the cocktail party, counting the number of red-haired partygoers and leaving when an alarm sounded. They were told to feel free to explore the environment after finishing their task, but not to leave the room until they heard the bell sound. When the alarm sounds, all of the partygoers also exit. We included this part of the experiment so that each subject was guaranteed to experience a high-density crowd.

After completing the task, subjects were administered a questionnaire to help us determine the level of *presence* that they experienced during their time in the VE. They were questioned about

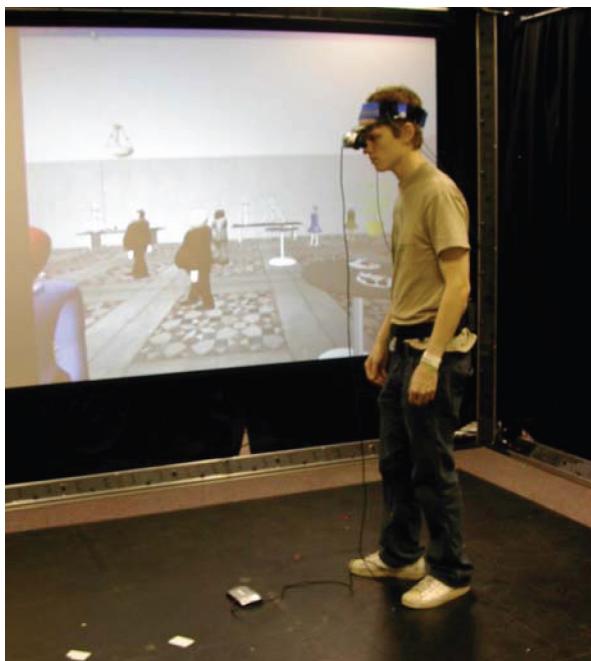


FIGURE 9.2: Participant during the experiment (Pelechano, Stocker et al. 2008).

their experience with video games and VEs to ensure that the independent variable (the different crowd models) was the only contributing factor to the differences in achieved *presence*.

After the first questionnaire was completed, they returned to the virtual cocktail party and were asked to count the number of red-haired partygoers again. As in the first part of the experiment, they were asked to exit the room when a bell sounded. This time, the partygoers were driven by a different crowd model. After the second experience, they filled out another copy of the questionnaire.

All the participants were videotaped during their participation for collection of data that could be used to study their involvement with the virtual people. After the experiment, participants would answer several questions regarding their experience.

Figure 9.2 shows a participant during the experiment wearing the head-mounted display and a large screen showing what the participant is observing. By videotaping the subject's behavioral response together with the scene we can simultaneously study the response of the person to the behavior of the virtual crowd.

9.4.4 Initial Results and Future Work

The goal of this pilot experiment was to examine whether participants interacting with a virtual crowd experience would react to the virtual crowd as they would do in a similar real situation.

From our current experiments, we have been able to observe that some participants did exhibit some behaviors consistent with the notion that they were responding to the crowd realistically. As we indicated above, each participant did two experiments, the scenarios were exactly the same, but in each case, we used a different crowd simulation model. Our goal for this pilot experiment is to study *presence* in a virtual crowd regardless of the crowd model being implemented.

The results obtained for this study came from standard questionnaires that contained a part with general questions and a part where participants could give any comments they had about their experience. The other source of results came from the authors' observing the subject's behavioral response from the videos. The part on questions was done initially to study the differences when running different crowds models and the part on gathering their comments and observing the videos were done to evaluate their *presence* in (by reactions to) a virtual crowd. In this section, we will focus on the comments and the behavioral response, since the questionnaires did not provide significant differences. As indicated in the literature on *presence*, questionnaires are not good enough by themselves, and therefore, in future work, we should include other methods such as galvanic skin response, ECG, respiration, administering personality tests, etc.

From the comments that our participants provided after doing the experiments, it is worth mentioning a few:

"The sense of crowd movement was most compelling during the evacuation."

"I felt bad whenever I bumped into someone."

“The second time, everyone immediately started leaving, and it made me really want to leave as well.”

These examples show that some people do think about the interaction with virtual agents in a similar way as when they interact with real people.

In addition to administering a questionnaire, we also gained insight by examining videotapes of participants' behavioral responses. In those videos, we observed people moving backward after bumping into a virtual agent, stepping sideways to avoid a virtual agent walking into them, and turning their head to watch an agent walk around them. One of the participants even waved back in response to a virtual agent's wave.

The pilot experiment had background crowd noise as well as the noise of the bell. A participant reported after the experiment, “I don't remember if the tables or people made sounds when I bumped into them. If they didn't, that might have helped knowing when I hit something.” This comment is very interesting from two perspectives; on the one hand, it shows such a high level of *presence* that the person is not even aware of what he has or has not heard during the experiment; on the other hand, it provides us with a valuable way of improving the next experiments. Given that it is not feasible to provide force feedback for such a scenario, it would be interesting to have some “natural” feedback that could allow the participant to realize that there is something wrong about the interaction or help in feeling more immersed in the VE. There were more comments from several participants regarding this topic, and although in general they were all pleased by the background noise enhancing their experience in a virtual crowd, several improvements should be made in the future such as:

- including stereo sound through headphones to enhance *presence* by being able to realize when, as a participant, you are bumping into an object or a person in the virtual crowd (i.e., when you bump into virtual agent you hear a noise or complaint);
- making the sound localized and clearer as the participant approaches a small group of people engaged in conversation, so that the participant can hear what they are talking about instead of just the noise of background voices.

As introduced in Section 4.2, during our pilot experiment, participants were first given a training session where they learned to navigate the environment, followed by two identical scenarios where different crowd simulation models were used. During training, participants were allowed to walk around and observe the environment until they located all three objects. This time varied from subject to subject. After the objects were located, subjects returned to the center of the room, and the crowd of agents began to move according to the crowd simulation model being used. The vast majority of the participants reported feeling more comfortable with the interaction during the second experiment, probably because the training time was not long enough or should have included agent movement.

“Much easier to navigate the second time. I had a feel for how fast I would be moving in the virtual world and felt like I could pay more attention to the task and less on walking/looking.”

An additional finding from the comments that were made about the insufficient training is that people appear to gauge their virtual movement based on the relative movement of others. Since subjects claim to not have understood their movement relationship with the world until they saw the virtual humans move, this is evidence that they are very sensitive not only to the general movement of the members of the virtual crowd but specifically to the inconsistencies between their own real movement and the artificial crowd movements. If this is the case, it is essential for the crowd members to move in a realistic way that the subject expects and can mimic.

Another important element that is mentioned in Section 3.1 is the communication factor, which would significantly increase the feeling of being part of a virtual crowd and the level of interaction with the agents:

“It would be more realistic to be able to make out conversations while close to groups of people.”

Finally, it is worth mentioning the current limitations of the equipment, mainly the low resolution of the head-mounted display and the narrow field of view:

“Restricted field of view made it harder, but I’m used to that from (other) games.”

“Low resolution made identifying the shrimp hard. . .”

In the future, we are considering using equipment that can provide higher levels of immersion and increase the feeling of *presence*, such as an immersive display room, which offers higher resolution and wider field of view.

9.4.5 Conclusions on *Presence* as a Validation Method

Crowd simulation models are currently lacking a commonly accepted validation method. We suggest the sense of *presence* in immersive VE as a possible method of validation. With the experimental evidence found in the *presence* literature, we can make a decision on which features a crowd simulation model should have to achieve high levels of *presence*.

Using egocentric features based on established *presence*-enhancing experiences, we hypothesize that interacting with the other agents in a crowd (by our virtual representation being pushed physically and by communicating with them) and being able to materially affect the movements of other members of the crowd (by pushing on them and having them avoid collisions with the user) will likely enhance a subject’s sense of *presence*. Arranging for the virtual crowd to push back (physically) on the subject is clearly more difficult, and we may be able to explore a haptic solution using vibrotactile elements (Bloomfield and Badler 2007). Experiments are in progress to test these hypotheses.

Virtual reality experiments with virtual crowds are necessary to study human behavior under panic or stressful situations that cannot be evaluated in the real world (i.e., building evacuation because of fire). To carry out those experiments, it is necessary to use a crowd simulation model in which a real person is seamlessly immersed and experiences a high sense of *presence* when interacting with such a crowd.

With a participant immersed in a VE crowd, we expect to observe the same type of behavior as in real life. Therefore, we could run experimental scenarios to study human behavior and decision making in stressful situations. Immersive VEs have successfully been applied to cure some phobias, such as fear of public speaking, heights, flying, etc. Likewise, we could use a VE for two new purposes: studying human behavior to improve current crowd simulation models and employing this VE for building design simulations.

• • • •

CHAPTER 10

Summary

We have presented a framework to realistically simulate crowds affected by psychological and physiological elements within complex virtual environments. The framework also includes contextual actions performed by agents with roles to create functional, heterogeneous crowds. To deal with the simulation of wayfinding and communication for each agent, we described the MACES system, and for low-level navigational movement of the agents, we presented the HiDAC system. To create functional crowd animations, the CAROSA system incorporates higher-level control and authoring of human textures.

CAROSA provides a framework for authoring simulations of everyday life. It is not limited to simulating evacuation scenarios, but it could be used to provide starting positions for such simulations. It aims to simulate the rich tapestry of human behaviors found in real-world scenarios. Attributes of the CAROSA framework include:

- the simulations of functional crowds depicting agents performing actions appropriate to the context;
- a parameterized action representation, and a persistent reusable database for storing them, that holds general semantics for both actions and objects, providing a baseline of information from which new scenarios can be constructed;
- four action types grounded in parameterized action representations that enrich the simulation with a variety of behaviors and can be fine-tuned to particular scenarios;
- behaviors that are emergent from the interaction of the four action types in a rich, dynamic environment;
- heterogeneous crowds that stem from both parameters available in MACES and HiDAC and from parameters used to link character traits to specific actions and contexts;
- interfaces designed to ease the burden of scenario creation and modification.

MACES implements wayfinding and inter-agent communication during an evacuation scenario for crowds unfamiliar with the internal structure of the environment. It considers individualism by assigning different roles to each agent (trained leaders, untrained leaders, and followers). The

flexibility of the model allows for variations in the number of people, building structure, number of hazards, and combinations of roles for the agents.

The main contributions of MACES are:

- A wayfinding algorithm to allow individuals in a crowd to explore an unfamiliar building in order to find exits during an emergency;
- using inter-agent communication to share knowledge of the building during building exploration and wayfinding;
- inclusion of roles to provide individualism into the crowd. Agents have a given personality that will drive high-level behavior, while they are also endowed with psychological elements such as impatience and panic that can affect internal state at any time and consequently modify overall behavior;
- inclusion of psychological elements (panic and impatience) that affect agents' wayfinding by introducing orientation difficulties and interactive path planning based on changes in the environment and bottlenecks.

The low-level system, HiDAC, can be tuned to simulate different types of crowds, ranging from extreme panic situations (fire evacuation) to high-density crowds under calm conditions (leaving a cinema after a movie). The system has been calibrated using data from real human behavior to exhibit reasonable velocities, flow rates, and densities.

At the local motion level, we introduced a list of techniques to achieve realistic high density crowd simulation including:

- eliminating shaking behavior implicit in the basic social forces model. The method consists of applying braking forces to the social forces model when repulsion forces coming from other agents appear opposite to the desired direction of movement.
- fast perception method based on having influence rectangles and prioritizing obstacles based on distances, angles, and directions of movement;
- natural bidirectional flows and overtaking based on a combination of variable length rectangles of influence, differential right preferences, and relative direction between autonomous agents;
- emergent queuing behavior by using influence discs that trigger waiting behavior based on agent direction. This, combined with different tangential weights for the avoidance forces, yields a variety of line and queue formations.
- realistic pushing behavior achieved by applying collision response based on pushing thresholds and personal distances;

- falling agents becoming new obstacles. These obstacles are addressed by applying weak tangential forces (but not repulsion forces), which do not guarantee that the agents will always walk around the fallen victim.
- panic behavior and panic propagation. Panic does not only serve to increase the velocity of the agents. In HiDAC, panic affects velocities and overall behavior by driving agents to not respect lines and by modifying pushing thresholds. Panic behavior can be perceived by other agents in the crowd and, given their personality parameters, who may also start exhibiting that type of emergent behavior.

By combining high-level action selection for functional crowds (CAROSA), mid-level decision making including navigation driven by communication and roles (MACES) and a low-level local motion system (HiDAC), groups of agents exhibit a large variety of emergent behaviors. The three systems interact in real time while being driven by a set of psychological and physiological parameters that allow the user to have control over the initial setup and final behavior exhibited by the crowd.

• • • • •

APPENDIX A

Simple Building Plan Editor

To facilitate rapid generation of a wide variety of large complex environments in which we could run our crowd simulation model, we implemented a building editor (Figure A.1).

This editor allows us to create buildings of any number of floors while we can simultaneously visualize them in both 2D and 3D (Figure A.2). The 2D view allows visualizing one floor at a time and specifying the geometry of the layout by locating walls, columns, stairs, windows, etc. just by clicking with the mouse in the position where we want to place the element selected from the right panel.

Once the building is created, we can save it in ASCII format that can be loaded afterward in our crowd simulation system to generate the cell and portal graph in which the autonomous agents can navigate. The ASCII format used allows easily picturing the final layout and also straightforwardly adding changes to the geometry without even needing to employ the editor.

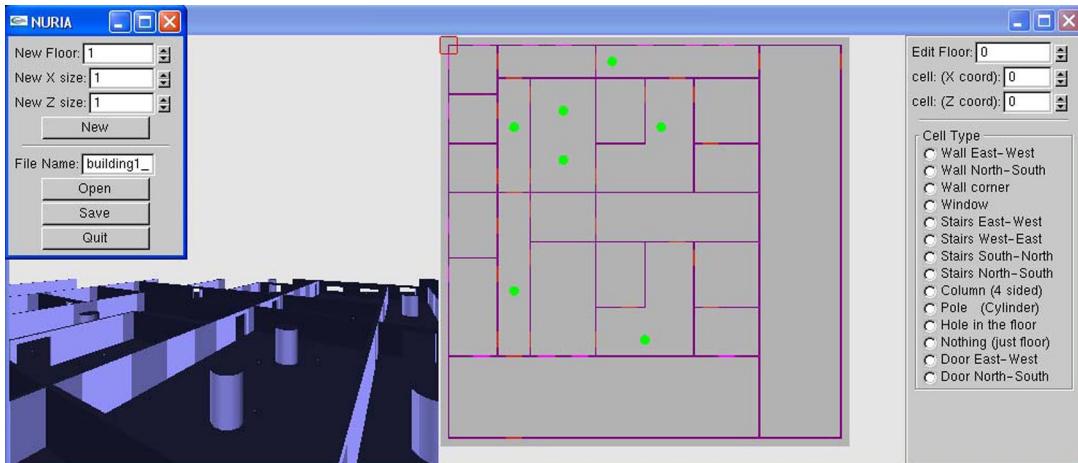


FIGURE A.1: Building editor.

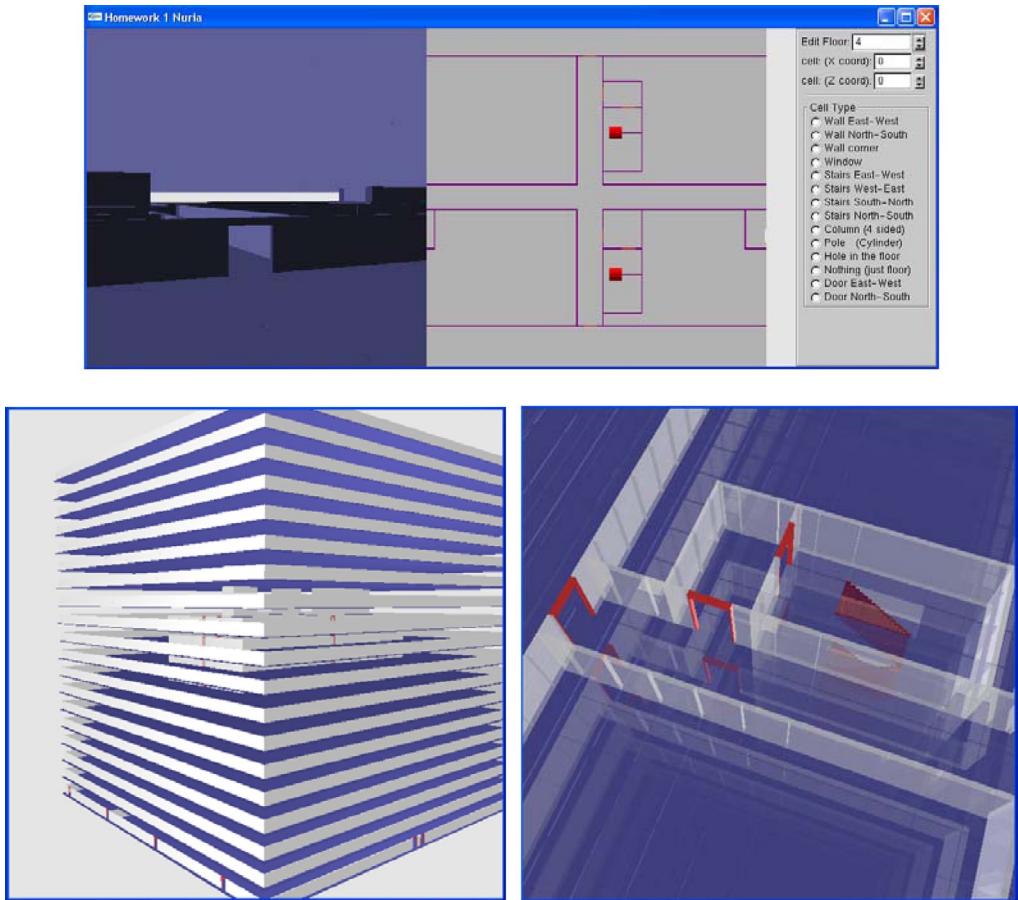


FIGURE A.2: Example of high-rise building editing and some 3D views.

Although this building editor is very limited in the type of buildings that we can create, it was out of the scope of this work to develop as complete a building editor as the commercial software tools available, but instead, our goal was to have our own fast and simple way of creating new buildings complex enough to test all the abilities of the crowd simulation system presented in this book. In the future, we would like to load appropriate CAD files as well and create the cell and portal graphs from those files that are necessary for our crowd simulation system.

APPENDIX B

Parameterized Action Representation

type parameterized action =

(name:	STRING;
participants:	agent-and-objects;
applicability conditions:	BOOLEAN-expression;
preparatory specification	<i>sequence</i> conditions-and-actions;
termination conditions	BOOLEAN-expression;
post assertion:	STATEMENT;
during conditions:	STATEMENT;
purpose:	purpose-specification;
subactions:	par-constraint-graph;
parent action:	parameterized action;
previous action:	parameterized action;
concurrent action:	parameterized action;
next action:	parameterized action;
start:	time-specification;
duration:	time-specification;
priority:	INTEGER;
data:	ANY-TYPE;
kinematics:	kinematics-specification;
dynamics:	dynamics-specification;
manner:	manner-specification;
adverbs:	<i>sequence</i> adverb-specification).

```

type agent-and-objects =
  (agent:                                agent representation;
   objects:                               sequence object representation).

type conditions-and-actions =
  (condition:                            BOOLEAN-expression;
   actions:                               parameterized action).

type purpose-specification =
  (achieve:                             BOOLEAN-expression;
   generate:                            sequence parameterized action;
   enable:                              sequence parameterized action).

type par-constraint-graph =
  (SEQUENTIAL,
   PARALLEL,
   PARALLEL-JOIN,
   PARALLEL-INDEPENDENT,
   WHILE).

type time-specification =
  (type:                                (ABSOLUTE,
   units:                               PAR-RELATIVE),
   value:                               (FRAMES, SECONDS);
                                         REAL).

type kinematics-specification =
  (time:                                time-specification;
   velocity:                            vector;
   acceleration:                       vector;
   position:                            site;
   path:                                path-specification).

type dynamics-specification =
  (force:                               vector;
   torque:                             vector).

type vector =
  (x:                                  REAL;

```

```

y:           REAL;
z:           REAL).

type site =
  (position:      vector;
   orientation:   vector).

type path-specification =
  (direction:    sequence direction-specification;
   start:        location-specification;
   end:          location-specification;
   distance:     REAL;
   modifiers:    (single-path-modifiers, aggregate-path-
                  modifiers)).

type direction-specification =
  (direction:   (ACROSS,
                 CLOCKWISE,
                 TO,
                 AROUND,
                 DOWN,
                 . . .),
   object:       object representation).

type location-specification =
  (site, sequence position-specification).

type position-specification =
  (position:    (ON,
                 AT,
                 IN,
                 . . .),
   object:       object representation).

type single-path-modifiers =
  (FOLLOWING,
   GUIDING,
   SHADOWING,
   . . .

```

type aggregate-path-modifiers = SWARMING,
 CONGREGATING,
 DISPERSING,
 ...).

type manner-specification = (effort: effort-specification;
 shape: shape-specification).

type effort-specification = (space: REAL;
 weight: REAL;
 time: REAL;
 flow: REAL).

type shape-specification = (vertical: REAL;
 lateral: REAL;
 sagittal: REAL;
 shapeflow: REAL).

type adverb-specification = (name: (SLOWLY,
 HAPPILY,
 EXCITEDLY,
 DIRECTLY,
 STRONGLY,
 HAPHAZARDLY,
 ...),
 modifiers: (EVEN,
 MORE,
 ...)).

type object representation = (name: STRING);

is agent:	BOOLEAN;
properties:	sequence property-specification;
status:	status-specification;
posture:	posture-specification;
location:	object representation;
contents:	<i>sequence</i> object representation;
capabilities:	<i>sequence</i> parameterized action;
relative directions:	<i>sequence</i> relative-direction-specification;
special directions:	<i>sequence</i> special-direction-specification;
sites:	<i>sequence</i> site-type-specification;
bounding volume:	bounding-volume-specification;
coordinate system:	site;
position:	vector;
velocity:	vector;
acceleration:	vector;
orientation:	vector;
data:	ANY-TYPE.

type property-specification =

(name:	STRING;
value:	ANY-TYPE).

type status-specification =

(NONE/DEAD,
IDLE/OPERATIVE,
ACTIVE(parameterized action).

type posture-specification =

(NONE,
NEUTRAL,
SIT,
STAND,
CROUCH,
PRONE,

SUPINE,
 KNEEL,
 OPEN,
 CLOSE,
 AJAR,
 ...).

type relative-direction-specification =
 (name: relative-orientation;
 value: site).

type relative-orientation = (FRONT,
 BACK,
 LEFT,
 RIGHT,
 TOP,
 BOTTOM).

type special-direction-specification =
 (name: STRING;
 value: site).

type site-type-specification =
 (name: (GRASP,
 APPROACH,
 BASE,
 ...),
 sites: sequence site).

type bounding-volume-specification =
 (type: (SPHERE,
 BOX,
 CONVEXHULL);
 value: sequence site).

APPENDIX C

Agent Process Algorithm

```

AgentProc::update() //executed for each agent in the simulation
{
    // Reactive actions
    attendToEnv(getCurrentRoom()) // reactive actions added to action queue

    // Opportunistic actions
    updateNeeds()
    checkNeeds() // updates the priority of the associated ipars on the queue
    scheduleOpportunistic() // determines if any of the opportunistic actions need
                            to be scheduled and updates the queue

    //Scheduled actions
    sort(actionQueue, priority) // queue manager functionality
    foreach ipar in the actionQueue { // starting with highest priority and going to lowest
        if (ipar.actionType == reactive) {
            suspendCurrentAction(); // stop current action and return it to the queue with the highest
                                    priority
            perform(ipar); // execute the reactive ipar, perform is the
                            process manager
        }
        else if(ipar.startTime <= currentTime)
            perform(ipar); //if there are conflicting actions, the perform function will keep processing
                            the highest priority action
    }
    if (status == idle) { //if there are currently no actions being performed. Note:
                            opportunistic actions don't have startTimes below a certain needLevel

```

```

        perform(defaultIpar); // the specific sub-action will be chosen from
                           // the distribution in the perform procedure
    }
}

AgentProc::perform(ipar)
{
    if (!ipar.terminationConditions() && // check termination conditions
        ipar.applicabilityConditions()) { // check applicability conditions
        handlePrepSpecs(ipar) // check preparatory specifications and add any necessary actions
        if (ipar.actionType == aleatoric)
            ipar.setSubActions(ipar.duration) // choose the sub-actions and their durations
                                              according to the distributions
        execute(ipar) // send the ipar to the motion generators (HiDAC) to be performed
        while (ipar.getStatus == executing) {
            if(ipar.terminationConditions()) {
                ipar.terminate() // end action
                ipar.postAssert() // assert post assertions including reaction statuses, locations, etc.
            }
            else updateStatus() // query or process updates from the motion generators, including
                               // during conditions and failures
        }
    }
}

//if action was a reaction, the post-assert updates the reaction.status to completed for that room
//and marks the time. When the agent leaves the room, completed is reset to pending.
void AgentProc::attendToEnv(int roomID) {
    foreach reaction on reactionlist {
        if (reaction.status == completed && currentTime - reaction.lastTime > delta) // reset the
            reactions after a predetermined time
        reaction.status = pending

        // if the condition holds and the reaction isn't happening
        if (HiDAC->checkStatus(reaction.condition, roomID) && reaction.status == pending)
            reaction.ipar.priority = highest on queue // could be based on individual
                                              differences (personal priorities)
    }
}

```

```

reaction.agents->addToQueue(reaction.ipar)
reaction.status = reacting
}
}
}

void AgentProc::updateNeeds() {
foreach need on needlist
need.level = need.level - need.rate // each need may have a different deterioration rate, which
should be much less than the additive rate
}

// There is currently one action that will fulfill each need and the action is always on the queue (if
not being executed)
void AgentProc::checkNeeds() {
foreach need on needlist
    need.ipar.priority = f(need.level) // priority is a function of need level
}

void AgentProc::scheduleOpportunistic() {
sort(needlist, -need.level) // prioritize needs with high levels
foreach need on needlist {
    availableTime = getSlackTime(timeWindow); // timeWindow would be how far in the
                                                future to look for available time
    proximity = f(need.level) // if the need is great the agents are willing to travel further
    resource = getClosestResource(proximity, getCurrentLocation(),
getNextAction(), need); // returns resource with proximity to the path to the next action
    if (canSchedule(resource, getNextAction()->location, availableTime, need.ipar.
duration))
    {
        // time to resource + action time + time to end goal
        ipar = createComplexIpar(need.ipar, getNextAction) // combine
                                                        the two ipars
        addToQueue(ipar)
    }
}
}

```


References

- AEA-Technology (2002). A Technical Summary of the AEA EGRESS Code.
- Ahn, J., Oh, S. and Wohn, K. (2006). Optimized Motion Simplification for Crowd Animation. *Computer Animation and Virtual Worlds*. **17**: 155–165. doi:[10.1002/cav.119](https://doi.org/10.1002/cav.119)
- Allbeck, J. and Badler, N. (2002). Embodied Autonomous Agents. *Handbook of Virtual Environments: Design, Implementation and Applications*. K. Stanney (ed.). Lawrence Erlbaum Associates, Philadelphia, PA: 313–332.
- Allbeck, J. and Badler, N. (2003). Representing and Parameterizing Agent Behaviors. *Life-like Characters: Tools, Affective Functions and Applications*. H. Prendinger and M. Ishizuka (eds.). Springer, Germany: 19–38.
- Allbeck, J. and Badler, N. (2004). Creating Embodied Agents with Cultural Context. *Agent culture: Designing virtual characters for a multi-cultural world*. R. Trappl and S. Payr (eds.). Lawrence Erlbaum Associates, New York: 107–126.
- Allbeck, J., Bindiganavale, R., Kipper, K., Moore, M., Schuler, W., Badler, N., Joshi, A. K. and Palmer, M. (2000). Authoring Embodied Agents' Behaviors through Natural Language and Planning. In. *Proc. Workshop on Key Problems for Creating Real-time Embodied Autonomous Agents at Autonomous Agents Conference*, Barcelona, Spain.
- Allbeck, J., Kipper, K., Adams, C., Schuler, W., Zoubanova, E., Badler, N., Palmer, M. and Joshi, A. (2002). ACUMEN: Amplifying Control and Understanding of Multiple Entities. In. *Proc. Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, ACM Press. New York, USA: 191–198.
- Allbeck, J. M. and Badler, N. I. (2001). Consistent Communication with Control. In. *Proc. Workshop on Non-Verbal and Verbal Communicative Acts to Achieve Contextual Embodied Agents at the 5th International conferences on Autonomous Agents*, Montreal (Canada): 21–26.
- Andersen, R., Berrou, J. L. and Gerodimos, A. (2005). On Some Limitations of Grid-Based (CA) Pedestrian Simulation Models. In. *Proc. First International Conference on Crowds Simulation. V-CROWD'05*, Lausanne (Switzerland).
- Ashida, K., Lee, S.-J., Allbeck, J.M., Sun, H. and Badler, N. I. (2001). Pedestrians: Creating Agent Behaviors through Statistical Analysis of Observation Data. In. *Proc. Computer Animation*,

- Seoul, Korea, IEEE Computer Society. Washington, DC, USA: 84–92. doi:[10.1109/CA.2001.982380](https://doi.org/10.1109/CA.2001.982380)
- Badler, N., Allbeck, J., Zhao, L. and Byun, M. (2002). Representing and Parameterizing Agent Behaviors. In. *Proc. Computer Animation*, Geneva, Switzerland, IEEE Computer Society Washington, DC, USA: 133–143. doi:[10.1109/CA.2002.1017521](https://doi.org/10.1109/CA.2002.1017521)
- Badler, N., Erignac, C. and Liu, Y. (2002). Virtual Humans for Validating Maintenance Procedures. *Communications of the ACM*. **45**(7): 56–63. doi:[10.1145/514236.514264](https://doi.org/10.1145/514236.514264)
- Badler, N., Phillips, C. and Webber, B. (1993). *Simulating Humans: Computer Graphics, Animation, and Control*. Oxford University Press, New York.
- Badler, N. I., Bindiganavale, R., Allbeck, J., Schuler, W., Zhao, L. and Palmer, M. (2000). Parameterized Action Representation for Virtual Human Agents. *Embodied Conversational Agents*. J. Cassell (ed.). MIT Press, Cambridge, MA, USA: 256–284.
- Bailenson, J. N., Blascovich, J., Beall, A. C. and Loomis, J. M. (2003). Interpersonal Distance in Immersive Virtual Environments. *Personality and Social Psychology Bulletin*. **29**: 1–15.
- Ball, G. and Breese, J. (2000). Emotion and Personality in a Conversational Character. *Embodied Conversational Agents*. J. Cassell, J. Sullivan, S. Prevost and E. Churchill (eds.). MIT Press, Cambridge, MA: 189–219.
- Barfield, W. and Hendrix, C. (1995). The Effect of Update Rate on the Sense of Presence within Virtual Environments. *Journal of the Virtual Reality Society*. **1**(1): 3–16. doi:[10.1007/BF02009709](https://doi.org/10.1007/BF02009709)
- Basdogan, C., Ho, C., Srinivasan, M. A. and Slater, M. (2000). An Experimental Study on the Role of Touch in Shared Virtual Environments. *ACM Transactions on Computer Human Interaction*. **7**(4): 443–460. doi:[10.1145/365058.365082](https://doi.org/10.1145/365058.365082)
- Bayazit, O. B., Lien, J.-M. and Amato, N. M. (2002). Roadmap-Based Flocking for Complex Environments. In. *Proc. Pacific Conference on Computer Graphics and Applications*, Beijing, China, IEEE Computer Society Washington, DC, USA: 104–113.
- Berrow, J. L., Beechan, J., Quaglia, P., Kagarlis, M. A. and Gerodimos, A. (2005). Calibration and Validation of the Legion Simulation Model using Empirical Data. In. *Proc. Pedestrian and Evacuation Dynamics (PED)*, Vienna, Springer Berlin: 167–181.
- Bindiganavale, R., Schuler, W., Allbeck, J., Badler, N., Joshi, A. and Palmer, M. (2000). Dynamically Altering Agent Behaviors Using Natural Language Instructions. In. *Proc. Autonomous Agents*, ACM New York, NY, USA: 293–300. doi:[10.1145/336595.337503](https://doi.org/10.1145/336595.337503)
- Bloomfield, A. and Badler, N. (2007). Collision Awareness using Vibrotactile Arrays. In. *Proc. IEEE Virtual Reality Conference*, Charlotte, NC, USA: 163–170.
- Bouvier, E. and Cohen, E. (1995). Simulation Of Human Flow With Particles Systems. In. *Proc. Simulators International XII*, Phoenix.

- Bouvier, E. and Guilloteau, P. (1996). Crowd Simulation in Immersive Space Management. In. *Proc. Eurographics Workshop on Virtual Environments and Scientific Visualization*, Springer-Verlag, Berlin: 104–110.
- Braun, A., Musse, S. R., de Oliveira, L. P. L. and Bodmann, B. E. J. (2003). Modeling Individual Behaviors in Crowd Simulation. In. *Proc. Computer Animation and Social Agents (CASA)*, IEEE Computer Society, Washington, DC, USA: 143–148. doi:[10.1109/CASA.2003.1199317](https://doi.org/10.1109/CASA.2003.1199317)
- Brogan, D. and Hodgins, J. (1997). Group Behaviors for Systems with Significant Dynamics. *Autonomous Robots*. **4**: 137–153.
- Brogan, D. and Hodgins, J. (2002). Simulation Level of Detail for Multiagent Control. In. *Proc. Autonomous Agents and Multiagent Systems*, Italy, ACM Press, New York, USA: 199–206 doi:[10.1145/544741.544789](https://doi.org/10.1145/544741.544789)
- Brown, R. W. (1954). Mass Phenomena. *Handbook of social psychology*. G. Lindzey (ed.). Addison-Wesley, Cambridge, Mass. **2**: 833–876.
- Buckmann, L. T. and Leather, J. (1994). Modelling Station Congestion the PEDROUTE Way. *Traffic Engineering and Control*. **35**(6): 373–377.
- Burgoon, J. K., Buller, D. B. and Woodall, W. G. (1989). *Nonverbal Communication, The UnSpoken Dialogue*. Harper and Row, New York.
- CAL3D(2008). 3D Character Animation Library: <http://home.gna.org/cal3d/>.
- Cassell, J., Bickmore, T., Billinghurst, M., Campbell, L., Chang, K., Vilhjalmsson, H. and Yan, H. (1999). Embodiment in Conversational Interfaces. In. *Proc. Special Interest Group on Computer-Human Interaction SIGCHI*, Pittsburgh, USA: 520–527. doi:[10.1145/302979.303150](https://doi.org/10.1145/302979.303150)
- Cassell, J., Sullivan, J., Prevost, S. and Churchill, E. (2000). *Embodied Conversational Agents*. The MIT Press, Cambridge, MA, USA.
- Chenney, S. (2004). Flow Tiles. In. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Grenoble, France: 233–242. doi:[10.1145/1028523.1028553](https://doi.org/10.1145/1028523.1028553)
- Chi, D., Costa, M., Zhao, L. and Badler, N. (2000). The EMOTE Model for Effort and Shape. In. *Proc. ACM SIGGRAPH*, New Orleans, LA, ACM Press, New York, USA: 173–183. doi:[10.1145/344779.352172](https://doi.org/10.1145/344779.352172)
- Collier, G. (1985). *Emotional Expression*. Lawrence Erlbaum, Hillsdale, NJ.
- Coyne, B. and Sproat, R. (2001). WordsEye: An Automatic Text-to-Scene Conversion System. In. *Proc. ACM SIGGRAPH*, ACM Press, New York, USA: 487–496. doi:[10.1145/383259.383316](https://doi.org/10.1145/383259.383316)
- Dijkstra, J., Timmermans, H. J. P. and Jessurun, A. J. (2000). A Multi-Agent Cellular Automata System for Visualizing Simulated Pedestrian Activity. In. *Proc. Theoretical and Practical Issues on Cellular Automata. Cellular Automata for research and Industry*, Springer-Verlag, Berlin: 29–36.

- Durupinar, F., Allbeck, J., Pelechano, N. and Badler, N. (2008). Creating Crowd Variation with the OCEAN Personality Model. In. *Proc. Autonomous Agents and Multi-Agents Systems*, Estoril, Portugal, ACM Press. New York, NY, USA: 1217–1220.
- Ekman, P. and Friesen, W. V. (1977). *Manual for the Facial Action Coding System*. Consulting Psychologists Press, Palo Alto, CA.
- El-Nasr, M. S., Ioerger, T. R. and Yen, J. (1999). Petteei: A Pet with Evolving Emotional Intelligence. In. *Proc. 3rd International Conference on Autonomous Agents*, Seattle, WA, ACM Press. New York, USA: 9–15. doi:10.1145/301136.301150
- Farenc, N., Boulic, R. and Thalmann, D. (1999). An Informed Environment Dedicated to the Simulation of Virtual Humans in Urban Context. In. *Proc. Eurographics*: 309–318. doi:10.1111/1467-8659.00351
- Farenc, N., Musse, S. and Schweiss, E. (2000). A Paradigm for Controlling Virtual Humans in Urban Environment Simulations. *Applied Artificial Intelligence* **14**: 69–91. doi:10.1080/088395100117160
- Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA.
- Freeman, J., Avons, S. E., Meddis, R., Pearson, D. E. and IJsselstijn, W. A. (2000). Using Behavioral Realism to Estimate Presence: A Study of the Utility of the Postural Responses to Motion Stimuli. *Presence: Teleoperators and Virtual Environments*. **9**: 149–164.
- Freeman, J., Avons, S. E., Pearson, D. E. and IJsselstijn, W. A. (1999). Effects of Sensory Information and Prior Experience on Direct Subjective Ratings of Presence. *Presence: Teleoperators and Virtual Environments*. **8**(1): 1–13. doi:10.1162/105474699566017
- Fruin, J. J. (1971). *Pedestrian Planning and Design*. 2nd. Ed., Elevator World, Mobile AL. 1987.
- Funge, J., Tu, X. and Terzopoulos, D. (1999). Cognitive Modeling: Knowledge, Reasoning, and Planning for Intelligent Characters. In. *Proc. SIGGRAPH*, Los Angeles, USA: 29–38.
- Galea, E. R., Perez Galparsoro, J. M. and Pearce, J. (1993). A Brief Description of the EXODUS Evacuation Model. In. *Proc. International Conference on Fire Safety*, San Francisco, USA: 149–162.
- GeniusConnect. (2007). GeniusConnect. <http://www.geniusconnect.com/articles/Products/2/3/>.
- Gibson, J. J. (1977). The theory of affordances. *Perceiving, Acting and Knowing*. R. Shaw and J. Bransford (eds.). Erlbaum, Hillsdale, NJ.
- Golledge, R. G. (1999). *Wayfinding behavior: cognitive mapping and other spatial processes*. Johns Hopkins University Press, Baltimore, MD, USA.
- Gratch, J. and Marsella, S. C. (2004). Evaluating the modeling and use of emotion in virtual humans. In. *Proc. Autonomous Agents and Multiagent Systems*, New York, ACM Press. New York, USA: 320–327.

- Hait, A., Siméon, T. and Taïx, M. (2002). Algorithms for Rough Terrain Trajectory Planning. *Advanced Robotics*. **16**(8). doi:[10.1163/15685530260425693](https://doi.org/10.1163/15685530260425693)
- Haumont, D., Debeir, O. and Sillion, F. (2003). Volumetric cell-and-portal generation. *Computer Graphics Forum*. **22**(3): 303–312.
- Hayes-Roth, B., Van Gent, R. and Huber, D. (1996). Acting in Character, Knowledge Systems Laboratory, Stanford University, Stanford, CA, USA.
- Helbing, D., Buzna, L., Johansson, A. and Werner, T. (2005). Self-Organized Pedestrian Crowd Dynamics. *Transportation Science*. **39**(1): 1–24. doi:[10.1287/trsc.1040.0108](https://doi.org/10.1287/trsc.1040.0108)
- Helbing, D., Farkas, I., Molnar, P. and Vicsek, T. (2002). Simulation of Pedestrians Crowds in Normal and Evacuation Situations. In. *Proc. Pedestrian and Evacuation Dynamics*, Springer-Verlag, Berlin: 21–58.
- Helbing, D., Farkas, I. and Vicsek, T. (2000). Simulating Dynamical Features of Escape Panic. *Nature*. **407**: 487–490.
- Henderson, L. F. (1971). The Statistics of Crowd Fluids. *Nature*. **229**: 381–383. doi:[10.1038/229381a0](https://doi.org/10.1038/229381a0)
- Hoogendoorn, S. P. (2003). Pedestrian Travel Behavior Modeling. In. *Proc. Travel Behavior Research* Lucerne, Elsevier: 10–15.
- Johnstone, K. (1979). *Status: Improv: Improvisation and the Theatre*. Theatre Arts Books, New York, NY, USA.
- Kavraki, L., Svestka, P., Latombe, J. and Overmars, M. (1996). Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transaction on Robotics and Automation*. **12**(4): 566–580. doi:[10.1109/70.508439](https://doi.org/10.1109/70.508439)
- Kirchner, A., Namazi, A., Nishinari, K. and Schadschneider, A. (2003). Role of Conflicts in the Floor Field Cellular Automaton Model for Pedestrian Dynamics. In. *Proc. 2nd International Conference on Pedestrians and Evacuation Dynamics. (PED)*, London, UK: 51–62.
- Klupfel, H. (2003). A Cellular Automaton Model for Crowd Movement and Egress Simulation. PhD thesis, University of Duisburg-Essen.
- Knapp, M. L. and Hall, J. A. (1992). *Nonverbal Communication in Human Interaction*. Harcourt Brace Jovanovich College Publisher, Fort Worth, TX.
- Kuligowski, E. D. and Peacock, R. D. (2005). A Review of Building Evacuation Models, Fire Research Division. Building and Fire Research Laboratory. National Institute of Standards and Technology.
- Kurke, L. B. and Aldrich, H. E. (1983). Mintzberg was Right!: A Replication and Extension of the Nature of Managerial Work. *Management Science*. **29**(8): 975–984.

- Kwek, S. (1997). On a Simple Depth-First Search Strategy for Exploring Unknown Graphs. In. *Proc. Workshop on Algorithms and Data Structures (WADS)*, Springer Lecture Notes in Computer Science: 345–353.
- Lamarche, F. and Donikian, S. (2004). Crowd of Virtual Humans: a New Approach for Real Time Navigation in Complex and Structured Environments. *Computer Graphics Forum*. **23**(3): 509–518. doi:[10.1111/j.1467-8659.2004.00782.x](https://doi.org/10.1111/j.1467-8659.2004.00782.x)
- Lau, M. and Kuffner, J. (2005). Behavior Planning for Character Animation. In. *Proc. ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation*, ACM Press. New York, USA: 271–280. doi:[10.1145/1073368.1073408](https://doi.org/10.1145/1073368.1073408)
- Lee, K. H., Choi, M. G., Hong, Q. and Lee, J. (2007). Group Behavior from Video: A Data-Driven Approach to Crowd Simulation. In. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, Eurographics Association Aire-la-Ville, Switzerland: 109–118.
- Legion. (2003). Legion International Ltd. from <http://www.legion.biz>.
- Lenat, D. and Guha, R. V. (1990). *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, Cambridge. MA, USA.
- Lerner, A., Chrysanthou, Y. and Cohen-Or, D. (2006). Efficient Cells-and-Portals Partitioning. *Computer Animation & Virtual Worlds*. Wiley, NY, USA. **17**(1): 21–40. doi:[10.1002/cav.70](https://doi.org/10.1002/cav.70)
- Lerner, A., Y., C. and Lischinski, D. (2007). Crowds by Example. *Computer Graphics Forum*. Blackwell Publishing Ltd. **26**: 655–664. doi:[10.1111/j.1467-8659.2007.01089.x](https://doi.org/10.1111/j.1467-8659.2007.01089.x)
- Lester, J. C., Towns, S. G., Callaway, C., Voerman, J. L. and FitzGerald, P. J. (2000). Deictic and Emotive Communication in Animated Pedagogical Agents. *Embodied Conversational Agents*. J. Cassell, J. Sullivan, S. Prevost and E. Churchill (eds.). MIT Press, Cambridge, MA, USA: 123–154.
- Lewis, H. (1998). *Body Language, a guide for professionals*. Response Books, New Dehli, India.
- Lien, J. M., Rodriguez, S., Malric, J. P. and Amato, N. M. (2005). Shepherding Behaviors with Multiple Shepherds. In. *Proc. IEEE Robotics and Automatation (ICRA)*, Barcelona, Spain: 3402–3407.
- Loscos, C., Marchal, D. and Meyer, A. (2003). Intuitive Crowd Behaviour in Dense Urban Environments using Local Laws. *IEEE Theory and Practice of Computer Graphics*: 122. doi:[10.1109/TPCG.2003.1206939](https://doi.org/10.1109/TPCG.2003.1206939)
- Lovas, G. C. (1994). Modeling and Simulation of Pedestrian Traffic Flow. *Transportation Research* **28**(6): 429–443. doi:[10.1016/0191-2615\(94\)90013-2](https://doi.org/10.1016/0191-2615(94)90013-2)
- Maletic, V. (1987). *Body, Space, Expression: The development of Rudolf Laban's Movement and Dance Concepts*. Mouton de Gruyter, New York.
- Marsella, S. C. and Gratch, J. (2002). A Step Toward Irrationality: Using Emotion to Change Belief. In. *Proc. 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, Bologna, Italy, ACM Press. New York, USA: 334–342. doi:[10.1145/544818.544821](https://doi.org/10.1145/544818.544821)

- Maslow, A. (1943). A theory of human motivation. *Psychological Review*. **50**: 370–396. doi:10.1037/h0054346
- Massive Software, Inc. “3D animation system for crowd-related visual effects.” Massive Software, Inc. <http://www.massivesoftware.com> (accessed 2005).
- Massive Software, Inc. “Artificial Life Solutions.” Massive Software, Inc. <http://www.massivesoftware.com/architecture-engineering-construction> (accessed September 19, 2008).
- McDonnell, R., Dobbyn, S. and O’Sullivan, C. (2005). LOD Human Representations: A Comparative Study. In. *Proc. International Workshop on Crowd Simulation (V-CROWDS)*, Lausanne, Switzerland: 101–115.
- McDonnell, R., Larkin, M., Dobbyn, S., Collins, S. and O’Sullivan, C. (2008). Clone Attack! Perception of Crowd Variety. *ACM Transactions on Graphics (SIGGRAPH 2008)*. **27**(3). doi:10.1145/1360612.1360625
- McDonnell, R., Newell, F. and O’Sullivan, C. (2007). Smooth Movers: Perceptually Guided Human Motion Simulation. *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*. San Diego, CA, ACM Press. New York, USA.
- McGrath, J. E. (1970). A Conceptual Formulation for Research on Stress. *Social and Psychological Factors in Stress*. J. E. McGrath (ed.). Holt Rinehart and Winston, New York: 10–21.
- McGrattan, K., Hostikka, S., Floyd, J., Baum, H. and Rehm, R. (2008). Fire Dynamics Simulator (Version 5) Technical Reference Guide. *NIST Special Publication 1018-5*, National Institute of Standards and Technology (NIST).
- Milazzo, J. S., Roushail, N. M., Hummer, J. E. and Allen, D. P. (1998). The Effect of Pedestrians on the Capacity of Signalized Intersections. *Transportation Research Record*. **1646**: 37–46 doi:10.3141/1646-05
- Moffat, D. (1997). Personality Parameters and Programs. *Creating Personalities for Synthetic Actors*. R. Trappi and P. Petta (eds.). Springer, New York: 120–165. doi:10.1007/BFb0030575
- Mori, M. (1970). The Uncanny Valley. *Energy*. **7**(4): 33–35.
- Mott-MacDonald (2003). STEPS Simulation of Transient Evacuation and Pedestrian Movements User Manual, Mott MacDonald.
- Musse, S. R., Babski, C., Capin, T. and Thalmann, D. (1998). Crowd Modelling in Collaborative Virtual Environments. In. *Proc. ACM Virtual Reality Software and Technology (VRST)*, Taipei, Taiwan: 115–123. doi:10.1145/293701.293716
- Musse, S. R. and Thalmann, D. (1997). A Model of Human Crowd Behavior: Group Inter-Relationship and Collision Detection Analysis. In. *Proc. Workshop of Computer Animation and Simulation of Eurographics*, Budapest, Hungary: 39–51.
- Musse, S. R. and Thalmann, D. (2000). From One Virtual Actor to Virtual Crowds: Requirements and Constraints. In. *Proc. Autonomous Agents*, ACM Press. New York, USA: 52–53. doi:10.1145/336595.336975

- Musse, S. R. and Thalmann, D. (2001). Hierarchical Model for Real Time Simulation of Virtual Human Crowds. *IEEE Transaction on Visualization and Computer Graphics* 7(2): 152–164. doi:[10.1109/2945.928167](https://doi.org/10.1109/2945.928167)
- O'Sullivan, C., Cassell, J., Vilhjalmsson, H., Dobbyn, S., Peters, C., Leeson, W., Giang, T. and Dingliana, J. (2002). Crowd and Group Simulation with Levels of Detail for Geometry, Motion and Behavior. In. *Proc. Third Irish Workshop on Computer Graphics*: 15–20.
- Ortony, A., Clore, G. L. and Collins, A. (1988). *The Cognitive Structure of Emotions*. Cambridge University Press, Cambridge.
- Owen, M., Galea, E. R., Lawrence and Filippidis, L. (1998). The Numerical Simulation of Aircraft Evacuation and its Applications to Aircraft Design and Certification. *The Aeronautical Journal*. **102**(1016): 301–312.
- Pan, X., Han, C. S. and Law, K. H. (2005). A Multi-agent Based Simulation Framework for the Study of Human and Social Behavior in Egress Analysis. In. *Proc. The International Conference on Computing in Civil Engineering*, Cancun, Mexico.
- Pelechano, N. (2006). Modeling Realistic Autonomous Agent Crowd Movement: Social Forces, Communication, Roles and Psychological Influences. PhD thesis, Computer and Information Science, University of Pennsylvania.
- Pelechano, N., Allbeck, J. and Badler, N. (2007). Controlling Individual Agents in High-Density Crowd Simulation. In. *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)*, San Diego, CA, ACM Press. New York, USA: 99–108.
- Pelechano, N. and Badler, N. (2006). Modeling Crowd and Trained Leader Behavior during Building Evacuation. *IEEE Computer Graphics and Applications*. **26**(6): 80–86.
- Pelechano, N. and Malkawi, A. (2008). Evacuation Simulation Models: Challenges in Modeling High Rise Building Evacuation with Cellular Automata Approaches. *Automation in Construction (Elsevier)*. **17**(4): 377–385. doi:[10.1016/j.autcon.2007.06.005](https://doi.org/10.1016/j.autcon.2007.06.005)
- Pelechano, N., O'Brien, K., Silverman, B. and Badler, N. (2005). Crowd Simulation Incorporating Agent Psychological Models, Roles and Communication. In. *Proc. First International Workshop on Crowd Simulation. (V-CROWDS '05)*, Lausanne, Switzerland: 21–30.
- Pelechano, N., Stocker, C., Allbeck, J. and Badler, N. (2008). Being a Part of the Crowd: Towards Validating VR Crowds Using Presence. In. *Proc. Autonomous Agents and Multiagent Systems (AAMAS)*, Estoril, Portugal, ACM Press. New York, NY, USA: 136–142.
- Perlin, K. and Goldberg, A. (1996). Improv: a System for Scripting Interactive Actors in Virtual Worlds. In. *Proc. ACM SIGGRAPH*, ACM Press. New York, USA: 205–216. doi:[10.1145/237170.237258](https://doi.org/10.1145/237170.237258)
- Pettre, J., Laumond, J.-P. and Thalmann, D. (2005). A Navigation Graph for Real-Time Crowd Animation on Multilayered and Uneven Terrain. In. *Proc. First International Workshop on Crowd Simulation (V-CROWD'05)*, Lausanne, Switzerland: 81–90.

- Poggi, I. and Pelachaud, C. (2000). Performative Facial Expressions in Animated Faces. *Embodied Conversational Agents*. J. Cassell, J. Sullivan, S. Prevost and E. Churchill (eds.). MIT Press, Cambridge, MA, USA: 155–188.
- Rastegary, H. and Landy, F. J. (1993). The Interaction Among Time Urgency, Uncertainty, and Time Pressure. *Time Pressure and Stress in Human Judgement and Decision Making*. O. Svenson and A. J. Maule (eds.). Plenum Publishing Corporation, New York: 217–240.
- Reynolds, C. (1987). Flocks, Herds, and Schools: a Distributed Behavior Model. In. *Proc. ACM SIGGRAPH*: 25–34. doi:[10.1145/37402.37406](https://doi.org/10.1145/37402.37406)
- Reynolds, C. (1999). Steering Behaviors for Autonomous Characters. In. *Proc. Game Developers Conference*: 763–782.
- Reynolds, C. (2006). Big Fast Crowds on PS3. In. *Proc. Sandbox (ACM SIGGRAPH symposium on Videogames)*, Boston, USA: 113–121.
- Rousseau, D. and Hayes-Roth, B. (1996). Personality in Synthetic Agents, Knowledge Systems Laboratory, Stanford University. CA, USA.
- Russell, S. J. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, USA. doi:[10.1038/nrn1651](https://doi.org/10.1038/nrn1651)
- Sanchez-Vives, M. V. and Slater, M. (2005). From Presence to Consciousness Through Virtual Reality. *Nature Reviews Neuroscience*. **6**(4): 332–339.
- Schroder, M. (2001). Emotional Speech Synthesis: A Review. In. *Proc. Eurospeech*, Aalborg: 561–564.
- Schubert, T., Friedmann, F. and Regenbrecht, H. (2001). The Experience of Presence: Factor Analytic Insights. *Presence: Teleoperators and Virtual Environments*. **10**(3): 266–281. doi:[10.1116/2/105474601300343603](https://doi.org/10.1116/2/105474601300343603)
- SFPE (2003). *The SFPE Engineering Guide to Human Behavior in Fire*. Society of Fire Protection Engineers, Bethesda, MA. doi:[10.1145/1073368.1073371](https://doi.org/10.1145/1073368.1073371)
- Shao, W. and Terzopoulos, D. (2005). Autonomous Pedestrians. In. *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, Los Angeles, California ACM Press. New York, USA: 19–28. doi:[10.1016/j.gmod.2007.09.001](https://doi.org/10.1016/j.gmod.2007.09.001)
- Shao, W. and Terzopoulos, D. (2007). Autonomous Pedestrians. *Graphical Models*. **69**: 246–274.
- Shields, T. J., Dunlop, K. and Silcock, G. (1996). Escape of Disabled People from Fire. A Measurement and Classification of Capability for Assessing Escape Risk, British Research Establishment, Borehamwood, London, UK.
- Silverman, B., Bharathy, G., Cornwell, J. and O'Brien, K. (2006). Human Behavior Models for Agents in Simulators and Games: Part II -Gamebots for a Foreign Culture. *Presence*. **15**(2): 163–185. doi:[10.1162/pres.2006.15.2.163](https://doi.org/10.1162/pres.2006.15.2.163)
- Silverman, B. G., Cornwell, J. and O'Brien, K. (2003). Human Performance Simulation. *Metrics and methods in human performance research toward individual and small unit simulation*.

- J. W. Ness, D. R. Ritzer and V. Tepe (eds.). Human Systems Information Analysis Center, Washington, DC.
- Sime, J. (1984). Behavior In Fire: 'Panic' Or Affiliation? *Department of Psychology*, University of Surrey, UK.
- Slater, M., Guger, C., Edlinger, G., Leeb, R., Pfurtscheller, G., Antley, A., Garau, M., Brogni, A. and Friedman, D. (2006). Analysis of Physiological Responses to a Social Situation in an Immersive Virtual Environment. *Presence: Teleoperators and Virtual Environments*. MIT Press, Cambridge, MA, USA. **15**(5): 553–569. [doi:10.1162/pres.15.5.553](https://doi.org/10.1162/pres.15.5.553)
- Slater, M. and Steed, A. (2000). A Virtual Presence Counter. *Teleoperators and Virtual Environments*. **9**: 413–434. [doi:10.1162/105474600566925](https://doi.org/10.1162/105474600566925)
- Slater, M., Usoh, M. and Steed, A. (1995). Taking Steps: the Influence of a Walking Technique on Presence in Virtual Reality. In. *Proc. ACM Transactions on Computer-Human Interaction (TOCHI)*: 201–219. [doi:10.1145/210079.210084](https://doi.org/10.1145/210079.210084)
- Still, G. K. (2000). Crowd Dynamics, Warwick University, Coventry, UK.
- Sud, A., Andersen, E., Curtis, S., Lin, M. and Manocha, D. (2007). Real-time Path Planning for Virtual Agents in Dynamic Environments. In. *Proc. IEEE Virtual Reality*, IEEE Computer Society, Washington, DC, USA: 91–98.
- Sud, A., Gayle, R., Andersen, E., Guy, S., Lin, M. and Manocha, D. (2007). Real-time Navigation of Independent Agents Using Adaptive Roadmaps. In. *Proc. ACM Symposium on Virtual Reality Software and Technology*, Newport Beach, CA, USA: 99–106. [doi:10.1145/1315184.1315201](https://doi.org/10.1145/1315184.1315201)
- Sung, M., Kovar, L. and Gleicher, M. (2005). Fast and Accurate Goal-Directed Motion Synthesis for Crowds. In. *Proc. Symposium on Computer Animation*: 291–300. [doi:10.1145/1073368.1073410](https://doi.org/10.1145/1073368.1073410)
- Sunshine-Hill, B., Allbeck, J. M., Pelechano, N. and Badler, N. I. (2007). Generating Plausible Individual Agent Movement from Spatio-Temporal Occupancy Data. In. *Proc. Workshop of Massive Datasets at the 9th International Conference on Multimodal Interfaces*, Nagoya, Japan, ACM Press. New York, USA: 5–7. [doi:10.1145/1352922.1352924](https://doi.org/10.1145/1352922.1352924)
- Tecchia, F., Loscos, C., Conroy, R. and Chrysanthou, Y. (2001). Agent Behavior Simulator (ABS): A Platform for Urban Behavior Development. In. *Proc. ACM/EG Games Technology Conference*.
- Teknomo, G. (2002). Microscopic Pedestrian Flow Characteristics: Development of an Image Processing Data Collection and Simulation Model. *Department of Human Social Information Sciences*, Graduate School of Information Sciences, Tohoku University.
- Teller, S. (1992). Visibility Computations in Densely Occluded Polyhedral Environments, UC Berkeley, CA, USA.
- Thalmann, D., Musse, S. R. and Kallmann, M. (1999). Virtual Humans' Behavior: Individuals, Groups, and Crowds. In. *Proc. Digital Media Futures*: 13–15.

- Thomas, G. and Donikian, S. (2000). Virtual Humans Animation in Informed Urban Environments. In. *Proc. Computer Animation*, IEEE Computer Society. Washington, DC, USA: 112–121. doi:10.1109/CA.2000.889057
- Thompson, P. A. and Marchant, E. W. (1994). Simulex: Developing New Computer Modelling Techniques for Evaluation. In. *Proc. Symposium Fire Safety Science* 613–624. doi:10.3801/IAFSS.FSS.4-613
- Thompson, P. A. and Marchant, E. W. (1995). Testing and Application of the Computer Model Simulex. *Fire Safety Journal* **25**: 149–166. doi:10.1016/0379-7112(95)00020-T
- Torrens, P. M. (2007). Behavioral intelligence for geospatial agents in urban environments. In. *Proc. IEEE Intelligent Agent Technology*, Los Alamitos, CA, IEEE: 63–66.
- Trappl, R. and Petta, P. (1997). *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*. Springer Verlag, Berlin, Germany.
- TRB (1994). Highway Capacity Manual Special Report, Transportation Research Board (TRB).
- Treuille, A., Cooper, S. and Popovic, Z. (2006). Continuum Crowds. In. *Proc. ACM Transactions on Graphics (SIGGRAPH 2006)*: 1160–1168. doi:10.1145/1179352.1142008
- Tsuji, Y. (2003). Numerical simulation of pedestrian flow at high densities. In. *Proc. Pedestrian and Evacuation Dynamics (PED)*, CMS Press, London, UK: 27–38.
- Tu, X. and Terzopoulos, D. (1994). Artificial Fishes: Physics, Locomotion, Perception, Behavior. In. *Proc. ACM SIGGRAPH*, ACM Press. New York, USA: 43–50. doi:10.1145/192161.192170
- Turner, A. and Penn, A. (2002). Encoding Natural Movement as an Agent-based System: an Investigation into Human Pedestrian Behaviour in the Built Environment. *Environment and Planning B: Planning and Design*. Pion Limited, London. **29**: 473–490. doi:10.1068/b12850
- Ulicny, B. and Thalmann, D. (2001). Crowd Simulation for Interactive Virtual Environments and VR Training Systems. In. *Proc. Eurographics Workshop on Animation and Simulation.*, Springer-Verlag. Berlin: 163–170.
- US Department of Labor. (Last visited 2008, 2008). Bureau of Labor and Statistics. from <http://www.bls.gov/>.
- Usoh, M., Catena, E., Arman, S. and Slater, M. (2000). Using Presence Questionnaires in Reality. *Presence: Teleoperators And Virtual Environments* **9**(5): 497–503. doi:10.1162/105474600566989
- Villamil, M. B., Musse, S. R. and Oliveira, L. P. L. d. (2003). A Model for Generating and Animating Groups of Virtual Agents. In. *Proc. Intelligent Virtual Agents*, Irsee, Germany: 164–169.
- Waldau, N., Schreckenberg, M. and Gatermann, P. (2003). Design Criteria Related to Orientation in Buildings during High-stress Situations Crowd Simulation Models and their Applications. In. *Proc. Pedestrian and Evacuation Dynamics. (PED)*: 307–318.
- Weaver, R., Silverman, B., Shin, H. and Dubois, R. (2001). Performance Moderator Functions for Modeling Adversary Organizations in Asymmetric Conflicts. In. *Proc. 10th Conference on Computer Generated Forces and Behavioral Representation*.

- Wiggins, J. S. (1996). *The Five-Factor Model of Personality: Theoretical Perspectives*. The Guilford Press, New York.
- Wolfram, S. (1983). Statistical Mechanics of Cellular Automata. *Reviews of Modern Physics*. **55**(3): 601–644. doi:[10.1103/RevModPhys.55.601](https://doi.org/10.1103/RevModPhys.55.601)
- Wray, R., Chong, R., Phillips, J., Rogers, S. and Walsh, B. A Survey of Cognitive and Agent Architecture. from <http://ai.eecs.umich.edu/cogarch0/>.
- Wren, C. R., Ivanov, Y. A., Leigh, D. and Westhues, J. (2007). The MERL Motion Detector Dataset: 2007 Workshop on Massive Datasets, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA. doi:[10.1145/1352922.1352926](https://doi.org/10.1145/1352922.1352926)
- Wright, W. (2008). The Sims. I. Electronic Arts.
- Yu, Q. and Terzopoulos, D. (2007). A Decision Network Framework for the Behavioral Animation of Virtual Humans In. *Proc. ACM SIGGRAPH/Eurographics symposium on Computer animation*, San Diego, California Eurographics Association: 119–128.

Author Biographies

Jan M. Allbeck is a PhD candidate in the Department of Computer and Information Science, which is a part of the School of Engineering and Applied Science of the University of Pennsylvania. Dr. Norman I. Badler is her advisor. She is also the Associate Director of the Center for Human Modeling and Simulation (HMS), where she coordinates and participates in the research projects affiliated with HMS as well as coordinates the operational aspects of the laboratory facility. Allbeck has bachelor's degrees in mathematics and computer science from Bloomsburg University and a master's degree in computer and information science from the University of Pennsylvania. She has explored many aspects of computer graphics, but is most drawn to research at the crossroads of animation, artificial intelligence, and psychology in the simulation of virtual humans. Her current research focuses on the creation and simulation of functional crowds.

Nuria Pelechano-Gomez is an associate professor of *Llenguatges i Sistemes Informàtics* at the *Universitat Politècnica de Catalunya, UPC* (Spain), where she is a member of the MOVING and Event Lab groups. Her research interests include modeling and simulation of large crowds with heterogeneous behaviors, interaction between virtual agents and real people in virtual environments, real-time 3D graphics, and presence. Pelechano received a BSc in computer science engineering from the *Universitat de València (Spain)* in 2001, an MSc with honors in vision, imaging and virtual environments (computer science) at the University College London (UK) in 2002, and a PhD in computer and information science at the University of Pennsylvania (USA) in 2006 as a Fulbright Scholar. Pelechano also completed her postdoctoral research at the University of Pennsylvania in the Center for Human Modeling and Simulation and in the T. C. Chan Center for Building Design and Energy Studies (School of Design), where she carried out numerous studies on building design based on pedestrian movement. She has published papers on crowd simulation in a number of international journals and conferences.

Norman I. Badler is a professor of computer and information science at the University of Pennsylvania and has been on that faculty since 1974. Active in computer graphics since 1968 with more than 200 technical papers, his research focuses on human modeling and animation control

with real-time 3D graphics. His current research interests include embodied agent animation and simulation, human-computer interfaces, crowd modeling and control, and computational connections between language and action. Badler received his BA degree in creative studies mathematics from the University of California at Santa Barbara in 1970 and his MSc in mathematics and PhD in computer science from the University of Toronto in 1971 and 1975, respectively. He is the co-editor of the Elsevier journal *Graphical Models*. He was the Cecilia Fitler Moore Department Chair of Computer and Information Science from 1990 to 1994. He directs the SIG Center for Computer Graphics and the Center for Human Modeling and Simulation at the University of Pennsylvania. Among the Center's achievements are the human modeling software system *Jack*, which was the basis for a spin-off company in 1996; the software is now marketed by Siemens. He is the Director of the Digital Media Design undergraduate degree program in computer science at the University of Pennsylvania. During 2001–2005, he was also the Associate Dean of the School of Engineering and Applied Science.