Animating Autonomous Pedestrians

by

Wei Shao

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Courant Institute of Mathematical Sciences

New York University

January, 2006

Advisor: Demetri Terzopoulos

To my mother and father, and to my wife.

Acknowledgements

I would like to take this opportunity to express my gratitude to the people who have helped and supported me during my Ph.D. program.

First and foremost, I am particularly grateful to my adviser, Professor Demetri Terzopoulos. It was his guidance, encouragement and collaboration that lead me along the bumpy road of Ph.D. study to this final accomplishment. I am so fortunate to have had the experience of research and study with him for the past five years, which has changed me and will be influencing me for the rest of my life.

Next, I would like to thank Professors Ken Perlin, Davi Geiger, Yann LeCun, Denis Zorin and Chris Bregler for serving on my proposal and dissertation committees. Special thanks go to Ken for his insightful opinions and suggestions on my research work.

I owe a lot to my colleagues and lab mates, among them Mauricio Plaza who worked on the reconstructed Penn Station model with me, Alex Vasilescu, Sung-Hee Lee and Evgueni Parilov who shared their ideas, opinions, discussion and jokes with me, and everybody at the Media Research Lab for the discussions, laughter, food and drink.

The research reported herein was supported in part by grants from the Defense

Advanced Research Projects Agency (DARPA) of the Department of Defense and from the National Science Foundation (NSF). My work has benefitted from collaboration with Brown University, in particular with Dr. Eileen Vote who created the reconstructed model of the Petra Great Temple (Daniel Feliz provided a recent version of the model) and with Professor David Cooper, the PI of the NSF Information Technology Research (ITR) grant that helped support my work. My thanks also go to Dr. Donald Sanders of the Institute for the Visualization of History for his advice and feedback on the application of our pedestrian simulator to virtual archaeology.

Most of all, I thank my loving wife, Yi Fang, for her great care, unconditional support and continuous encouragement throughout this time, without which this dissertation would not have been possible.

Abstract

This thesis addresses the difficult open problem in computer graphics of autonomous human modeling and animation, specifically of emulating the rich complexity of real pedestrians in urban environments.

We pursue an artificial life approach that integrates motor, perceptual, behavioral, and cognitive components within a model of pedestrians as highly capable *individuals*. Our comprehensive model features innovations in these components, as well as in their combination, yielding results of unprecedented fidelity and complexity for fully autonomous multi-human simulation in large urban environments. Our pedestrian model is entirely autonomous and requires no centralized, global control whatsoever.

To animate a variety of natural interactions between numerous pedestrians and their environment, we represent the environment using hierarchical data structures, which efficiently support the perceptual queries of the autonomous pedestrians that drive their behavioral responses and sustain their ability to plan their actions on local and global scales.

The animation system that we implement using the above models enables us to run long-term simulations of pedestrians in large urban environments without manual intervention. Real-time simulation can be achieved for well over a thousand autonomous pedestrians. With each pedestrian under his/her own autonomous control, the self-animated characters imbue the virtual world with liveliness, social (dis)order, and a realistically complex dynamic.

We demonstrate the automated animation of human activity in a virtual train station, and we employ our pedestrian simulator in the context of virtual archaeology for visualizing urban social life in reconstructed archaeological sites. Our pedestrian simulator is also serving as the basis of a testbed for designing and experimenting with visual sensor networks in the field of computer vision.

Contents

List of Figures

List of Tables

List of Appendices

Chapter 1

Introduction

"Forty years ago today at 9 a.m., in a light rain, jack-hammers began tearing at the granite walls of the soon-to-be-demolished Pennsylvania Station, an event that the editorial page of The New York Times termed a "monumental act of vandalism" that was "the shame of New York." "

(Glenn Collins, The New York Times, 10/28/03)

Figure 1.1: The original Pennsylvania Train Station in New York City.

The demolition of New York City's original Pennsylvania Station (Figure 1.1), which had opened to the public in 1910, in order to make way for the Penn Plaza complex and Madison Square Garden, was "a tragic loss of architectural grandeur". Although state-of-the-art computer graphics enables a virtual reconstruction of the train station with impressive geometric and photometric detail (Figure 1.2), it does not yet enable the automated animation of the station's human occupants with anywhere near as much fidelity. This thesis addresses the difficult, long-term challenge of automated human animation.

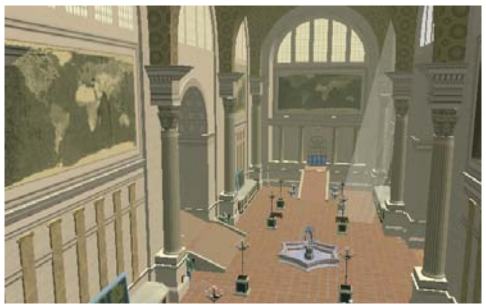
The creation of lifelike characters has been one of the most provocative topics in computer animation. From traditional expressive cartoon characters such as Disney's "Snow White"[1] to modern CG characters such as Pixar's "Buzz Lightyear"[2] we have experienced the revolutionary transition from manual drawing to computer-generated graphics. However, most lifelike characters on the big screen today remain hand-animated in labor-intensive fashion, using key-framing or a mixture of key-framing and motion capture techniques. As the latest modeling and rendering technologies begin to produce synthetic characters that look remarkably photorealistic, such as the virtual humans of the 2001 CGI film "Final Fantasy",[3] the animation research community is increasingly challenged to develop techniques that can automatically achieve compatibly realistic motions and even autonomous behaviors for such characters. The need for self-animated, lifelike virtual humans is even more acute in the rapidly evolving interactive game industry.

For more than a decade, research groups in academia and industry have been

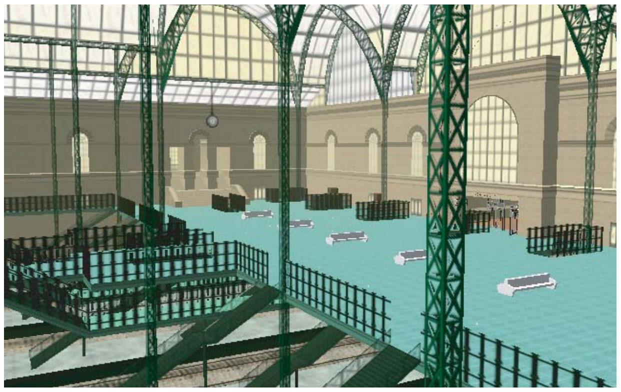
[1] *Snow White and the Seven Dwarfs*, 1937, Walt Disney Pictures.
[2] *Toy Story*, 1995, and *Toy Story 2*, 1999, Pixar.
[3] *Final Fantasy: The Spirits Within*, 2001, Square USA.

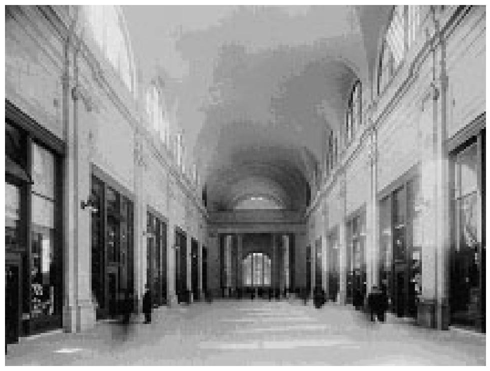
(a)

(b)


(c)

(d)


(e)


(f)

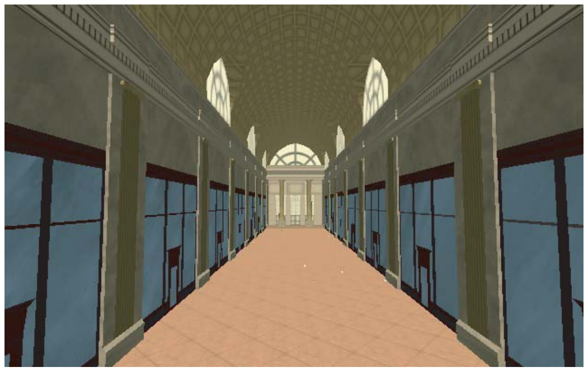

(g)

Figure 1.2: The original (a,b,d,f) and reconstructed (c,e,g) Pennsylvania Train Station, including (a,b,c) views of the main waiting room, (d,e) the upper concourse, and (f,g) the arcade.

pursuing an artificial life approach to synthesizing self-animated characters that are in some rudimentary sense "alive" [Terzopoulos 1999]. A variety of animation systems have been reported in the literature that can create lifelike characters patterned after lower animals for various practical purposes, from scientific simulation, to education, to entertainment including movies and interactive computer games. Recently, researchers have turned their ambitions towards the holy grail of creating lifelike autonomously self-animated humans that inhabit complex virtual worlds. Partial solutions aimed at various sub-problems toward this long-range goal have been proposed.

1.1 Research Focus

Ideally, to mimic a real human being, every aspect of a living human should be included in a character model—from the musculoskeletal system, to skin and clothes, from body motion to facial expression, from internal emotion to external behavior, from perception to interaction, and from learning to knowledge representation and reasoning. This may be a reasonable breakdown from a natural scientist's point of view. On the other hand, from a sociologist's perspective, an individual can play different roles in life, such as a graduate student, a violinist, a child, a parent, a spouse, a shopper, a driver, an office clerk, a teacher, an athlete, a conference delegate, etc. Although many of these roles can be played by the same biological human, each potentially requires different sets of motions, behaviors, and intelligent skills. Clearly, developing a general autonomous human model capable of the full repertoire of observed human complexity is a gargantuan, seemingly impossible task. To make progress, we must focus our attention to an interestingly complex yet tractable slice of the big picture that has not yet been fully addressed.

In this thesis, we take an artificial life approach to solving the difficult open problem of animating autonomous virtual pedestrians in large urban spaces. Our primary objective will be a computer model of a self-animated human pedestrian, including its motor, perceptual, behavioral, and cognitive abilities. Our secondary yet also important objective will be to develop external data structures and algorithms that can efficiently support numerous instances of our highly capable pedestrian model within a suitably rich synthetic urban environment. As a preview of our results, Figure 1.3 illustrates the virtual Penn Station environment populated by our autonomous virtual pedestrians.

1.2 Methodology and Architecture

Though much simpler than modeling a general human being, modeling a pedestrian is still a very tough problem. To tackle the problem, we pursue an artificial life modeling approach [Terzopoulos 1999], constructing a fully autonomous artificial human that possesses external characteristics and internal control mechanisms consistent with those of a real pedestrian, including

- a geometric model of body shape and articulation;
- color and texture properties of the skin, eyes, hair, and clothes;
- various controllable body movements, primarily for locomotion;
- sensing of the external environment, mainly via visual perception;

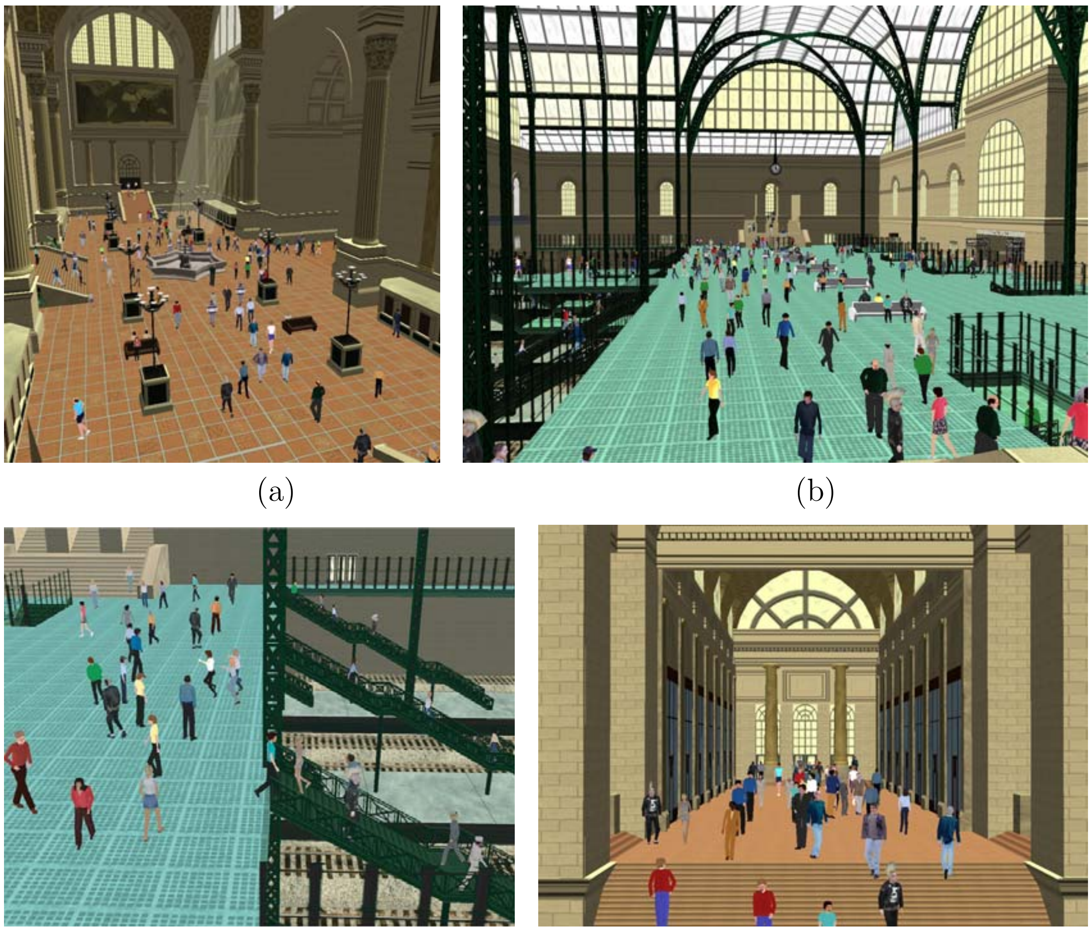

(a) (b) (c) (d)

Figure 1.3: A large-scale simulation of Penn Station populated by self-animated virtual humans. Rendered images of the main waiting room (a), concourses (b,c), and arcade (d).

- internal representation of individual physiological, psychological and social needs;
- a behavioral repertoire, including obstacle avoidance behaviors, following behaviors, etc;
- a reactive behavioral control mechanism that initiates, sequences, and terminates behaviors in accordance with internal motivators and external sensory stimuli;
- higher-level deliberative control mechanisms based on knowledge, which provide the intelligence to interact with the environment and achieve long-range goals.

Our virtual pedestrians are compound models spanning five distinguishable modeling levels, as is shown in Figure 1.4. This modeling hierarchy or pyramid is patterned after those depicted in [Terzopoulos 1999; Funge et al. 1999]. At the lowest level is the geometric human model, including an articulated skeleton with colored/textured skin and clothes. The second level is a kinematic motion layer which covers motion synthesis and motor control. Above that is the perceptual modeling level, including sensing processes for both external situation and internal states. The behavioral model on the next level controls a character's reactive and motivational behaviors in response to both internal and external stimuli obtained through perception processes. At the apex of the modeling pyramid, knowledge representation, reasoning, and planning are encapsulated in the cognitive model, which provides characters with deliberative intelligence and free will. Finally, the modeling hierarchy associated with the individual pedestrian is embedded within

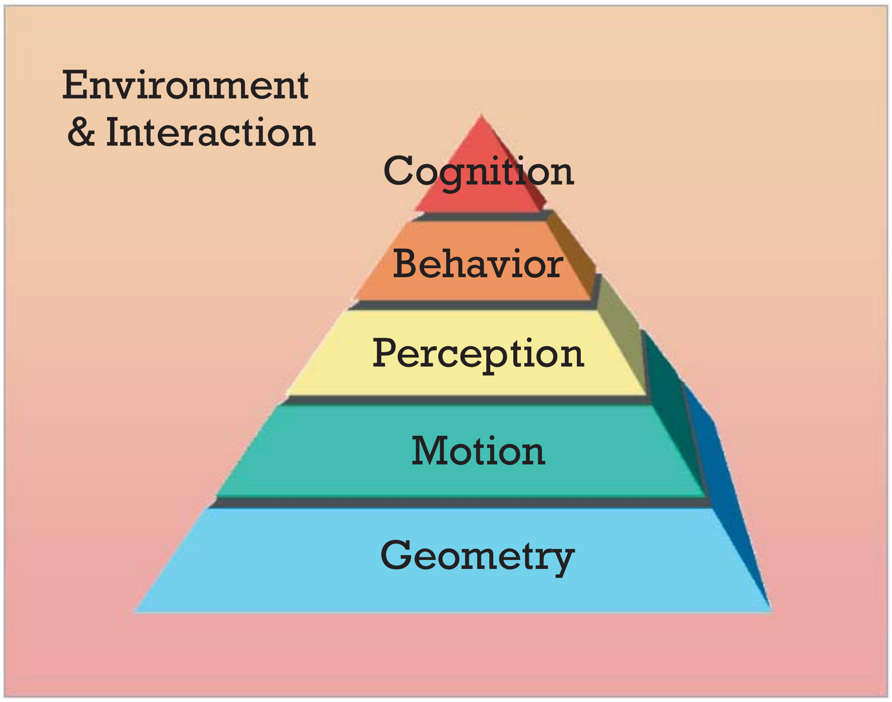


Figure 1.4: Modeling pyramid.

the environment and interaction model. The latter supports the relations and interactions between the pedestrians and their environment.

An important guideline for our work is that, as in real life, each virtual pedestrian should be an autonomous, intelligent individual. More explicitly, each pedestrian should be able to control itself across the motor, perceptual, behavioral and cognitive levels. Just like real people, it should be an autonomous agent that does not require any external, global coordination whatsoever, including control by any real human animators in order to cope with its highly dynamic environment. However, this criterion does not preclude the possibility for our virtual characters to be directed by animators at an abstract level in a manner that reflects certain animation objectives.

1.3 Contributions

In a departure from the substantial literature on so-called “crowd simulation”, the contribution of this thesis is a decentralized, comprehensive model of pedestrians as autonomous *individuals* capable of a broad variety of activities in large-scale synthetic urban spaces. While our work is innovative in the context of behavioral animation, it is very different from simple crowd animation, where one character algorithmically follows another in a stolid manner, which is relatively easy to accomplish with simple rules. We are uninterested in crowds *per se*. Rather, the goal of our work has been to contribute a comprehensive, self-animated model of individual human beings that incorporates nontrivial human-like abilities suited to the purposes of animating virtual pedestrians in urban environments. In particular, we pay serious attention to deliberative human activities over and above the reactive behavior level. Our artificial life approach to modeling humans spans the modeling of pedestrian appearance, locomotion, perception, behavior, and cognition. Our approach has been inspired most heavily by the work of [Tu and Terzopoulos 1994] on artificial animals and by [Funge et al. 1999] on cognitive modeling for intelligent characters that can reason and plan their actions. We have taken this comprehensive artificial life approach further, adopting it for the first time to the case of an autonomous virtual human model that can populate large-scale urban spaces.

Following the above described methodology and architecture, our overall contribution is a novel framework for human simulation, an implementation that can efficiently animate numerous pedestrians autonomously performing a rich variety of activities in large-scale urban environments, and its application. In more detail, our specific major contributions are as follows:

1. A sophisticated new model of autonomous pedestrians that has innovations in each of its components as well as in their combination. The pedestrian model, which was reported in [Shao and Terzopoulos 2005a], includes:

   (a) a motor control interface that hides the details of the low-level motion implementation;

   (b) a perception paradigm that considers not only sensing processes, but situation interpretation as well;

   (c) a robust repertoire of reactive behavior routines and a novel way of combining them;

   (d) a variety of navigational behaviors with concerns from both global and local perspectives; and

   (e) cognitive planners and a memory model suitable for pedestrians.

2. A new, hierarchical environment model, which was reported in [Shao and Terzopoulos 2005b], that supports natural interactions among pedestrians in extensive and highly dynamic virtual worlds through:

   (a) efficient perceptual processing; and

   (b) fast online path planning at various scales.

3. A human simulation system combining implementations of the two models that can automatically produce real-time animation on high-end PCs involving well over a thousand simulated pedestrians in large-scale urban environments.

4. Novel applications of our simulator to virtual archeology for the purposes of visualizing urban social life in reconstructed archaeological sites, as well as a potential computer-based archaeological tool in developing and testing theories about ancient site usage.

Regarding the latter contribution, in addition to the aforementioned Penn Station environment, we have applied our pedestrian simulator to a reconstruction of the Great Temple archaeological site of ancient Petra in Jordan (Figure 1.5). Our work has also enabled research in the domain of computer vision where our pedestrian simulator is being used as a developmental testbed for visual surveillance sensor networks.

1.4 Thesis Overview

The remaining chapters of this dissertation are organized as follows: In Chapter 2, we review related prior work. In addition to the voluminous relevant literature in computer graphics and animation, we cover some related work in robotics and the scientific study of pedestrian traffic.

Chapter 3, presents a technical overview of our pedestrian simulation framework, including the virtual environment model and the virtual pedestrian model.

Chapter 4 and Chapter 5 proceed with a full explanation of the autonomous pedestrian model, covering the details of the behavioral control and cognitive control sub-models, respectively. After briefly describing the low-level human modeling package that we use and the specific implementation of pedestrian's appearance and lower-level motor control, Chapter 4 presents the various behavior modules,

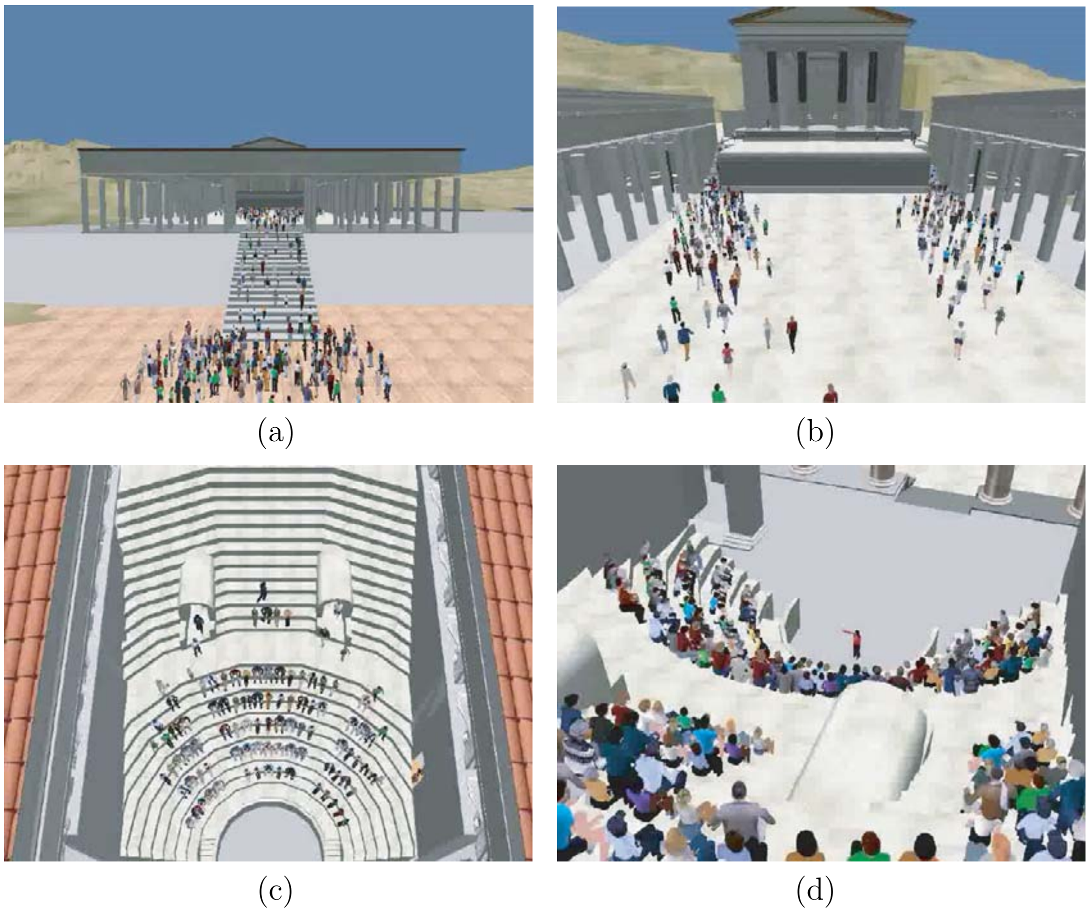


Figure 1.5: Filling the Petra Great Temple amphitheater.

how they alter the motor control commands, how they are combined to give optimal performance, and how appropriate behaviors are triggered both by the pedestrian's external environment and internal states through sensory processes. Chapter 5 presents the heuristic guidelines underlying the cognitive model and how it enables pedestrians to form plans, execute them, and update them, and how these plans affect the behavioral layer.

Chapter 6 presents the full details of the hierarchical environment model, which supports large-scale data organization, accurate perceptual queries, and efficient path search algorithms, enabling the real-time animation of well over a thousand pedestrians in large-scale virtual worlds.

With the environment model fully explained, we can then explain in Chapter 7 the details of the algorithms that support local and global path planning in the autonomous pedestrians.

In Chapter 8, we present several animation results and analyze the performance of our simulation system.

Chapter 9 draws conclusions from our work and discusses avenues for future research and application.

The main body of the dissertation is supplemented by three appendices. Appendix A presents the details of our scheme for optimally sequencing the reactive behavior routines. Appendix B presents additional motivational behavior routines for autonomous pedestrians in the Penn Station environment. Finally, Appendix C investigates motion analysis and synthesis, presenting new algorithms that can potentially serve as an alternate low-level motion layer for virtual pedestrians.

Chapter 2

Related Work

The creation of lifelike virtual creatures that emulate their counterparts in the real world is obviously a very challenging task, considering the complexity of the creature's biology and its dynamic physical surroundings. Such complexity increases to the extreme in the case of modeling human beings [Badler et al. 1993]. The focus of our research—to model pedestrians as self-controlled individuals—is a difficult open sub-problem of this task.

In this chapter, we will review previous work related to our research in the fields of artificial life, computer animation, scientific simulation, and robotics. We will start with an overview of motion synthesis techniques, continue with a detailed review of behavioral models and autonomous agents related to our work, and conclude with a brief discussion of environment models that support interaction between agents and their surroundings.

2.1 Natural Motion Synthesis

Creating realistic motion is a basic part of animating lifelike characters. Generally speaking, the most commonly used motion synthesis and editing techniques fall into the following three categories: kinematic, dynamic, and hybrid.

Kinematic techniques: Keyframing is the most traditional kinematic technique and it has proven successful for decades. It gives the artist full control over the character's action and thus, given enough skill and perseverance, it can yield very expressive animation. Many vivid cartoon characters have been created using this method. However, as key framing requires lots of interactive editing and usually takes a considerable amount of time, it is best used as a tool for offline animation production with a small number of characters. As vision technology has developed, a new motion creation method—motion capture—has become increasingly popular. Simply stated, motion capture is the recording of human body movement (or other movement) for immediate or delayed analysis and playback [Sturman 1994]. Ignoring inaccuracies during recording, the acquired motion data yields a realistic animation of the original motion if it is transferred to a "virtual copy" of the original character. To obtain animation with variation, several offline editing techniques (in addition to the most traditional keyframing), such as warping [Witkin and Popovic 1995], retargeting [Gleicher 1998], path editing [Gleicher 2001], transition generation [Rose et al. 1996], and motion signal processing [Bruderlin and Williams 1995], have been developed in order to adapt the original data to particular needs. Possible artifacts involving high-frequency details, such as foot skate, which are likely to be introduced in the above editing processes, can be removed by methods such as those described in [Kovar et al. 2002b]. Motion capture data can also enhance keyframing animations by adding in missing degrees of freedom and missing motion details [Pullen and Bregler 2002]. While motion capture is drawing increasing attention from the animation community, it still lacks the flexibility to be useful in applications featuring dynamic and interactive environments. Recently, several research groups [Kovar et al. 2002a; Arikan and Forsyth 2002; Lee et al. 2002] have started employing new techniques to analyze and abstract motion databases into graphs. Their abstractions transform the motion synthesis problem into graph walks. By drawing upon algorithms from graph theory and AI planning, particular graph walks that satisfy certain constraints can be extracted efficiently, thereby making motion synthesis fast and controllable [Kwon and Shin 2005].

Dynamic techniques: Motion synthesis via physical simulation usually requires a dynamic character model together with a set of actuators and controllers. Actuators emulate natural muscles and controllers activate and coordinate these actuators, driving the dynamic character model to produce desired motions. To synthesize realistic motions, physically-based models are used to compute the dynamics of passive moving bodies subject to kinematic constraints. This approach frees the animator from having to specify many low-level motion details, since motion is synthesized automatically by the procedural controllers through physical simulation. A substantial amount of research devoted to animating creatures using this method has already demonstrated its success in non-human animals, such as fishes [Tu and Terzopoulos 1994] and birds [Wu and Popovic; 2003]. However, humans have more elaborate musculoskeletal systems and thus need more complex control mechanism for coordinated motions. Consequently, controllers for physically based humans are difficult to construct and motion synthesis is usually computationally expensive. Although encouraging research results such as the animation of human athletics [Hodgins et al. 1995] and the virtual stuntman [Faloutsos et al. 2001b; Faloutsos et al. 2001a] have revealed the potential, human motions synthesized by dynamic simulation still tend to look robotic—they are physically valid but still lack the natural fluidity evident in biological humans.

Hybrid techniques: Seeing the respective advantages and drawbacks of kinematic and dynamic approaches, researchers have recently started combining the two methods, hoping to gain better performance out of the resulting hybrid techniques. Zordan and Hodgins [2002] have developed interactive character models that include trajectory tracking controllers that follow motion capture data and balance controllers that keep the character upright while modifying motion sequences to accomplish specified tasks, such as throwing punches or swinging a racket. Their simulated human characters can respond automatically to impacts and return smoothly to tracking motion data. Shapiro *et al.* [2003] have implemented a framework that employs both kinematic and dynamic controllers. They also developed transition methods between the two control methods for interactive character animation. In their framework, characters are able to perform natural-looking gaits and react dynamically to unexpected situations. Pollard and Zordan [2005] propose a controller for physically based hand grasping that draws from motion capture data. They show that a controller derived from a single motion capture example can be used to form grasps of different object geometries. Arikan *et al.* [2005] present an algorithm for animating characters being pushed by an external disturbance. Their technique uses a collection of motions of a real person responding to being pushed, and synthesizes new motions by selecting a motion from the recorded collection and modifying it so that the character responds to the push from the desired direction and location on its body. Similar approaches combining kinematic and dynamic techniques have also drawn the attention of the animation industry. Natural Motion, Inc.'s, Active Character Technology [Natural Motion, Ltd 2003] is capable of producing smooth "hand-over" transitions between pre-designed or imported character motions and physically simulated motions for real-time interactive character animation.

Each of the above three types of techniques has its own advantages and disadvantages. As we want our virtual humans to be animated online in real time, we need a motion synthesis technique that provides both speed and quality. To this end, recent advances in motion capture research have drawn our attention. We have designed an efficient data analysis and synthesis algorithm which can produce high quality motions as desired in real time. We have implemented our algorithm as an Alias Maya [Alias Systems Corp. 2005] real-time animation plug-in that is designed specifically for this purpose. However, in our current pedestrian simulation system, we employ the human animation package *DI-Guy* [Koechling et al. 1998] from Boston Dynamics, Inc. [Boston Dynamics, Inc. 2004], which also provides us with textured character models. Thus, we are restricted to using only the motion repertoire and blending algorithms that come with this software. Human motion synthesis, therefore, falls outside of the major focus of this thesis. For completeness, however, we provide the details of our motion analysis and synthesis algorithms in Appendix C, as an alternative for interested readers.

2.2 Behavioral Models and Autonomous Agents

Animating a character, whether human or non-human, with lifelike behaviors is anything but simple for an artist. Animation of groups of characters, such as flocks of birds or schools of fish, requires even more effort and time in addition to creativity. Not only does the overall motion of the whole group need to be highly coordinated, each individual creature should have its own distinct behavioral details as well. To tackle the problem, researchers developed self-animating character models that efficiently generate animation for groups of characters with much less effort than before. A variety of behavioral animation models and systems have appeared in both research and industry. We list a few representative samples in the following discussion.

2.2.1 Behavioral Characters in Animation

Boids: In his seminal work, Reynolds [1987; 1999] proposed a computational model of a distributed (multi-agent) behavioral system. In his approach, each animated character, called a "boid", is able to carry out a behavior from a small repertoire (separation, alignment, and cohesion) based on the location of its neighboring boids at any given time. The organized flock is an emergent property of the autonomous interactions between individual behaviors. Using a similar model, Brogan and Hodgins [1997] have reproduced behaviors for groups of simulated creatures traveling fast enough that dynamics plays a significant role in determining their movement. Their algorithm is evaluated in three different simulated systems—legged robots, humanlike bicycle riders, and point-mass systems.

Artificial Fishes: Tu and Terzopoulos [1994] developed a physics-based, virtual marine world inhabited by lifelike artificial life forms that emulate the appearance, motion, and behavior of fishes in their natural habitats. Each artificial fish is an autonomous agent that can demonstrate a repertoire of piscine behaviors, including collision avoidance, foraging, preying, schooling, courting, and mating. Despite their rudimentary brains, compared to the biological brains of real animals, these artificial fishes can learn basic motor functions and carry out perceptually guided motor tasks [Terzopoulos et al. 1994; Terzopoulos et al. 1996].

Improv: "Improv" is an authoring system developed by Perlin and Goldberg [1996] for creating interactive worlds inhabited by believable animated actors. Authors can use an *Animation Engine* to create layered, continuous, non-repetitive motions and smooth transitions, and can use the *Behavior Engine* to create rules governing how actors communicate, change, and make decisions. Characters created using the system are behavior-based as well as interactive. They can respond to each other and to users in real-time, with personalities and moods consistent with user-defined goals and intentions.

Mr. Bubb: Similarly, Loyall *et al.* [2004] presented an innovative system that enables the authoring of rich procedural knowledge related to autonomous interactive characters. Their system is composed of a programming language with emotion/expression models and a motion synthesis system that can combine hand-animated motion data with artistically authored procedures. “Mr. Bubb” is a character with emotion and rich personality that was created for a simple interactive game using their system.

EMOTE: Badler and his collaborators introduced a 3D character animation system called “EMOTE” (Expressive MOTion Engine) [Chi et al. 2000], which was inspired by movement observation science, specifically Laban Movement Analysis [Maletic 1987] and its “Effort and Shape” components. With the help of this system, they are able to create and parameterize specific agent behaviors (such as upper-body motions during walking, arm gestures, and facial expressions [Ashida et al. 2001; Badler et al. 2002]) more naturally.

AlphaWolf: Focusing on the synthetic social behavior for interactive virtual characters, Tomlinson *et al.* [2002] have built a computational model that captures a subset of the social behaviors of wild wolves, involving models of learning, emotion, and development. By howling, growling, whining or barking into a microphone in their interactive installation “AlphaWolf”, users can play the role of wolf pups and therefore influence other members of a pack of autonomous and semi-autonomous virtual wolves.

Social Learning in Synthetic Characters: Buchsbaum *et al.* [2005] have created animated characters that are able to observe, recognize and imitate what others are performing. In addition, their characters can identify simple goals and motivations from perceived behavior, and can learn about new objects by observing and correctly interpreting interactions performed by others with these objects.

Cognitive Modeling: Funge *et al.* [1999] have developed a cognitive modeling language CML, which is based on a logic formalism from artificial intelligence known as the situation calculus. They used it in different animation settings (a prehistoric world with dinosaurs and an undersea world with a shark and a "merman"), imbuing synthetic characters with domain knowledge and deliberative character behavior in terms of goals. With their cognitively empowered characters, the animator need only specify a brief high-level "script" and, through reasoning, the character will automatically work out a detailed sequence of actions satisfying the specification. This approach allows behavioral animation to be directed more intuitively, more succinctly and at a much higher level of abstraction than would otherwise be possible.

2.2.2 Low-level Pedestrian Simulation in Science

Next, we direct our attention to pedestrian models. In the scientific studies of pedestrian traffic, there are two major categories of models, microscopic and macroscopic. The former focuses on traffic characteristics of individual units, such as individual speed and individual interaction, while the latter studies the collective flow of aggregated pedestrian movement by examining average speed, covered area, etc. Since we are interested in detailed models of pedestrians as individuals, the microscopic model is more relevant to our work. There are several different categories of simulation models in microscopic pedestrian studies: Cellular models, physical force models, and queuing models. Teknomo [2002] provides a more detailed comparison among different microscopic pedestrian simulation models.

Cellular Models: Gipps and Marksjo [Gipps and Marksjo 1985] simulated pedestrians as particles in a grid of cells, each of which is assigned a cost score based on proximity to the pedestrian and a gain score based on distance to the target. According to these two scores, a pedestrian can pick the most beneficial neighboring cell to proceed toward. Later, Blue and Adler [1998; 2000] developed a cellular automata model for pedestrian simulation, which was originally applied to car traffic simulation. In their model, two parallel stages—lane change and cell hopping—are applied in each simulation time step for a pedestrian.

Physical Force Models: In 1979, Okazaki [1979] developed a magnetic model in which pedestrians and obstacles are assigned positive poles and goal locations assume negative poles. The overall magnetic effect causes pedestrians to move to their goals and avoid collisions. Helbing and Molnar [1995] introduced a model for pedestrian simulation based on "social forces", which is a measurement for the internal motivations of the individuals to perform certain actions (movements). Their model can be further applied to describe group dynamics and other social phenomena.

Queuing Models: These models, which were developed by several researchers [Watts 1987; Lovas 1993; Thompson and Marchant 1995; Schreckenberg and Sharma 2001], are often used for evacuation simulation in architecture design and urban planning.

2.2.3 Pedestrian Models in Animation

Hierarchical ViCrowd: Musse and Thalmann [2001] introduced a simulation system called "ViCrowd". As the name indicates, the goal of their system is to generate crowd simulation based on large groups rather than individuals. Autonomy is developed for their crowd model and by means of pre-programmed, scripted and interactive control, their system can generate crowd behaviors with various levels of realism.

Reactive Massive Crowd: Based on artificial life technology, Massive Software, Inc., [Massive Software, Inc. 2005] has developed a 3D animation tool called "Massive" for generating crowd-related visual effects and character animation for film and television. Using this software, artists can design characters with a set of reactions to what is going on around them. These reactive characters possess a special vision process, and a sense of hearing and touch that allows them to respond naturally to their environment.

Reactive Navigation using Iterative Optimization: Lamarche and Donikian [2004] developed a system that enables reactive pedestrian navigation within a hierarchical topological structure built from the geometric database of a virtual environment. In their approach, an iterative optimization process is used for pedestrians to avoid collisions and reach targets at the same time.

Situation-based Behaviors for Scalable Crowd Simulation: Sung *et al.* [Sung et al. 2004] developed a situation-based animation system that embeds composable behaviors into the environment. As characters enter a specific place/situation, behaviors stored there can temporarily be added to the character's behavior repertoire and they can be composed with the existing behaviors on the fly to enable characters to respond to the situation more appropriately. These augmented situation-specific behaviors are removed once the character leaves the situation behind.

2.2.4 Other Related Models

Subsumption Architecture: The subsumption model introduced to the robotics community by Brooks [1986; 1991; 1995] had an unconventional control architecture. The traditional approach decomposed the control problem into a series of functional units and information flowed sequentially through each unit until the final control commands can be picked after last unit. However, in Brooks' model "task achieving behaviors" are used to replace functional units. They have parallel access to perceptual input and can simultaneously control the same set of robot actuators. To resolve the conflicts in control competence, higher level function units, when they wish to take control, can subsume the roles of lower level ones. The construction of the whole system is incremental from the bottom up, and at any level a complete operational control system can be formed by using only the layers below that level.

Combining Deliberation, Reactivity, and Motivation: Stoytchev and Arkin [2001] described a hybrid mobile robot architecture that combines deliberative planning, reactive control, and motivational drives to deal with dynamic and unpredictable environments and high-level human commands. Using this architecture, they were able to control a mobile robot to accomplish a fax delivery mission in a normal office environment.

2.2.5 Comparative Summary

Considering the substantial research effort devoted to behavioral models and autonomous agents from both academia and industry, our review above was by no means exhaustive. However, as a good representative subset, our selection spans the spectrum from simple particle systems (such as those in Section 2.2.2) to visually convincing systems with realistic artificial creatures that resemble their counterparts in the real world (such as those in Section 2.2.1). With each of them having its own focus, they demonstrate various advantages. Some of the examples are fully autonomous characters that do not require any direction from users (e.g., artificial fishes [Tu and Terzopoulos 1994]) while others are more friendly to authoring, guidance and interaction (e.g., Improv [Perlin and Goldberg 1996]); some put more emphasis on individuals (e.g., EMOTE [Chi et al. 2000; Ashida et al. 2001; Badler et al. 2002]), and others may focus more on large groups (e.g., ViCrowd [Musse and Thalmann 2001]); some of the characters are simple and purely reactive (e.g., [Lamarche and Donikian 2004]) and therefore can be used to animate crowds in real time, while there are intelligent ones that can do reasoning, planning, or even learning (e.g., the cognitive creatures [Funge et al. 1999] and social-learning characters [Buchsbaum and Blumberg 2005]). A comparison of the aforementioned models and systems is summarized in Table 2.1.

Our work is greatly inspired by the existing systems, especially those that aspire to fully autonomous characters. Our goal is to achieve realtime animation, at the

	Level of Autonomy	**Level of Intelligence**	**User Interaction**	**Learning or Development**
Boids	Medium	Low	No	No
Artificial Fishes	High	Medium	No	Learning
Improv	Medium	Medium	Authoring & Interaction	Development
Mr. Bubb	High	Medium	Interaction	No
Emote	Low	Low	Authoring	No
Cognitive Creatures	High	High	No	No
AlphaWolf	High	Medium	Interaction	Development
Social learning Characters	High	Medium/High	Interaction	Development
Low Level Pedestrian Models	Low/Medium	Low	No	No
ViCrowd	Various	Medium	Authoring	No
Massive	Medium	Low	Authoring	No
Reactive Navigation	Medium	Low	No	No
Situation-based Behavior	High	Low	Authoring	No
Subsupmtion Architecture	Incremental	Incremental	No	No
Deliberation +Reactivity +Motivation	High	Medium	No	No
Autonomous Pedestrians	High	High	User-Controllable Characters	No

Table 2.1: Comparing different animation models and systems

level of individual behaviors, of numerous virtual pedestrians in large-scale urban spaces. Although this may sound similar to the research of "crowd animation", which has been a hot topic for the past decade in both academia [Musse and Thalmann 2001; Tecchia et al. 2002; Loscos et al. 2003; Ulicny et al. 2004; Sung et al. 2005] and industry [Prasso et al. 1998; Autodesk 3ds Max 2005; Massive Software, Inc. 2005; BioGraphic Technologies, Inc. 2005], it is actually quite different. Crowd animation usually focuses on the collective movement of a group of characters, each one being a simple reactive agent with comparably low intelligence. On the contrary, we focus on modeling *intelligent individuals*. The goal of our work is to develop a comprehensive, self-animated model of *individual* humans that incorporates nontrivial abilities suited to the purposes of animating virtual pedestrians in urban environments. Our characters must be fully autonomous so they can perceive, behave, reason, and plan in a manner similar to real pedestrians. Consequently, the animation produced from these virtual pedestrians will be more natural and purposeful than that generated by conventional "crowd animation" systems.

While our work is innovative in the context of behavioral animation, the approach taken here is inspired most heavily by the work of [Tu and Terzopoulos 1994] on artificial animals and by [Funge et al. 1999] on cognitive modeling for intelligent characters that can reason and plan their actions. We develop this comprehensive artificial life approach further and adopt it for the first time to the case of an autonomous virtual human model suitable for large-scale urban spaces.

2.3 Environmental Modeling

To enable artificial characters to interact with their virtual environment, a suitable representation of the environment is indispensable in supporting efficient sensing, navigation, or path planning. While most of the currently available environmental modeling techniques were developed in the robotics community for robot navigation, novel environmental models have also appeared in the field of computer animation with the growth in interest in autonomous agents and behavioral animation. We briefly survey the different techniques in the remainder of this section. (A more detailed survey on robotic mapping can be found in [Thrun 2002].)

2.3.1 Map Representations in Robotics

The *Occupancy Grid* mapping technique [Elfes 1987; Moravec 1988] represents the environment with a grid of fixed resolution. This multidimensional grid maintains the occupancy state of each cell. This model has been widely used since its introduction because of its simple yet rich representation, but it has two main drawbacks: 1) the grid can hardly capture the topological relationships between regions; 2) it usually suffers from high cost in both time and space when fine grids are used.

Another metric approach is the *feature map* [Leonard and Durrant-Whyte 1991], in which the environment is represented with parametric features such as points, lines, cylinders, corners, etc., and is augmented with information of the features such as position, geometry, color, etc. It is a highly applicable model for visual processing of sensory data. However, such a technique is not useful for an unstructured environment, where it is not always possible to find clearly distinct geometries.

Topological approaches [Kuipers and Byun 1991] use graphs to represent the environment. Nodes and arcs in the graph indicate relations between regions, such as adjacency and connectivity. On the one hand, this compact abstraction facilitates certain tasks, such as path planning, but on the other hand, it does not support detailed navigation due to its lack of metric information, such as absolute position.

Given the advantages and disadvantages of the various approaches, researchers have started to combine different techniques to form *Hybrid Maps* [Thrun and Buecken 1996; Guivant et al. 2004]. The combined models can benefit from every technique used and are able to show satisfactory performance. Our virtual environment model falls into this hybrid category.

2.3.2 Environment Models in Animation

In the field of computer animation, as synthetic characters are getting more and more sophisticated, modeling their surrounding world is drawing increasing attention from researchers.

Lamarche and Donikian [2004] developed a system that can build an accurate hierarchical topological structure from the geometric database of a virtual environment. Based on this structure, they were able to develop visibility computation, close neighbor detection, collision avoidance and optimized path planning algorithms.

Noser *et al.* [Noser et al. 1995] used synthetic vision as the only sensing channel for their actors to perceive their environment. A dynamic octree serves as the internal representation for actors to memorize what they see and to determine where to go.

In the *Informed Environment* research reported by Farenc *et al.* [1999], an urban scene is decomposed into a hierarchy of *Environment Entities* that provide not only geometrical information but semantic notions as well, allowing a more realistic simulation of character behavior.

Sung *et al.* [2004] proposed a more "meaningful" environment model, which embeds composable character behaviors in the regions where they are relevant. These behaviors can be added into characters' behavior repertoire when they enter the associated regions. While this model is unnatural, it provides an interesting solution to the problem of interpreting and interacting with the environment after it is perceived.

Even from the brief review above, it is becoming increasingly clear that, in the context of modeling artificial characters, an environment model must go well beyond the representation of maps and objects. It also needs to facilitate all perceptual processes, interpretation, and interaction, which makes it a difficult research problem.

Chapter 3

Framework

This chapter overviews our autonomous pedestrian simulation framework, whose two major components include the environment model and the autonomous pedestrian model. The details of each of these two components are deferred to later chapters.

3.1 Virtual Environment Model

The interaction between a pedestrian and his/her environment plays a major role in the animation of autonomous virtual humans in synthetic urban spaces. This, in turn, depends heavily on the representation and (perceptual) interpretation of the environment. We have devoted considerable effort to developing two large-scale urban environment models, one is an (indoor) model of a train station, the other is an (indoor/outdoor) archeological site model. To make the bulk of the ensuing discussion in this dissertation more concrete, we will briefly describe the train station model at this juncture.

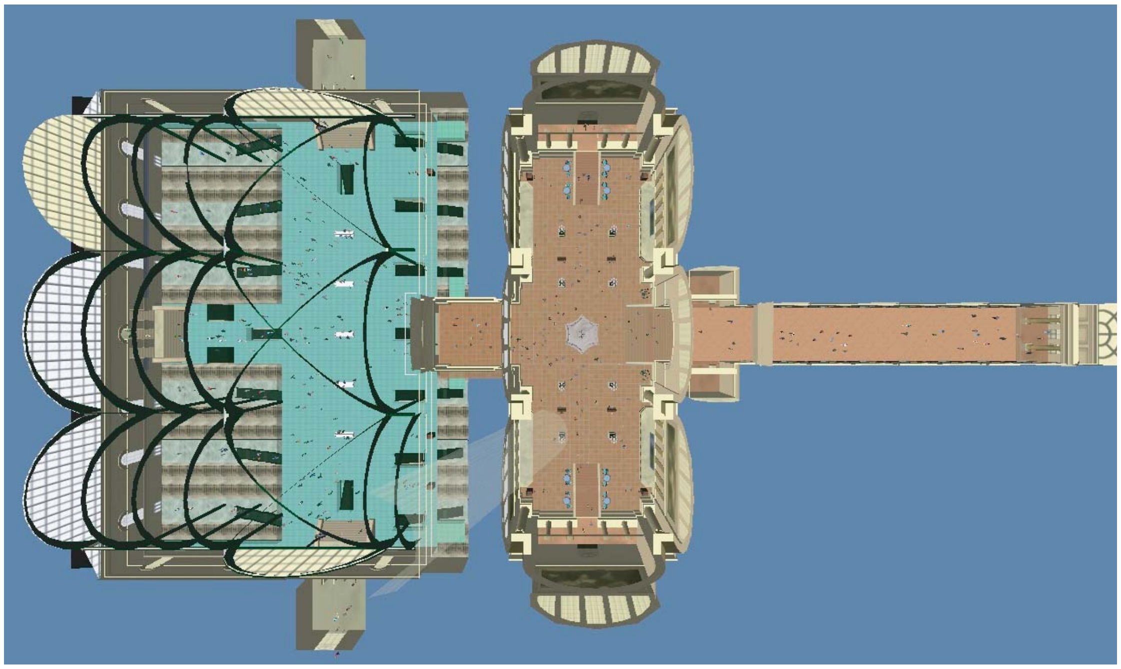


Figure 3.1: Plan view of the Penn Station model with the roof not rendered, revealing the 2-level concourses and the train tracks (left), the main waiting room (center), and the long shopping arcade (right).

The original Pennsylvania Train Station of New York City was a monumental architectural masterpiece that was originally built in 1910 and, sadly, demolished in 1963. Figure 1.1 and the left column of Figure 1.2 show historical photographs of the exterior and interior of the station, respectively. The reconstructed 3D model of Penn Station, which includes raw geometry and textures, (see the right column of Figure 1.2) is available in MultiGen-Paradigm Inc.'s OpenFlight file format [MultiGen-Paradigm, Inc. 2005] and was distributed to us by Boston Dynamics, Inc. [Boston Dynamics, Inc. 2004]. Figure 3.1 shows a roofless plan view of the rendered Penn Station geometric model with the two-level concourse at the left, the main waiting room at the center, and the long arcade at the right. Geometrically, the station is a large 3D space (200m (l) $\times$ 150m (w) $\times$ 20m (h)) that is architecturally complex, featuring hundreds of objects, including levels of floors, stairs, walls, doorways, big columns, ticket booths, platforms, train tracks, etc. These objects, together with a collection of non-architectural objects that we have added to the model such as lamps, fountains, benches, vending machines, tables, sporadic trash on the floors, etc., sum up to over 500 in total.

We represent the virtual environment by a hierarchical collection of maps. As illustrated in Fig. 3.2, the model comprises (i) a *topological map* which represents the topological structure between different parts of the virtual world. Linked within this map are (ii) *perception maps*, which provide relevant information to perceptual queries, and (iii) *path maps*, which enable online path-planning for navigation. Finally, on the lowest level, are (iv) *specialized objects* that support quick and powerful perceptual queries.

In the topological map, nodes correspond to environmental regions and edges represent accessibility between regions. A region is a bounded volume in 3D-space (such as a room, a corridor, a flight of stairs or even an entire floor) together with all the objects inside that volume (e.g., ground, walls, benches). The representation assumes that the walkable surface in a region may be mapped onto a horizontal plane without loss of essential geometric information. Consequently, the 3D space may be adequately represented within the topological map by several 2D, planar maps, thereby enhancing the simplicity and efficiency of environmental queries.

The perception maps include grid maps that represent stationary environmental objects on a local, per region basis, as well as a global grid map that keeps track of mobile objects, usually virtual pedestrians. Each cell of the stationary object grid maps, whose cell size is typically $0.2 \sim 0.3$ meters, stores information that identifies

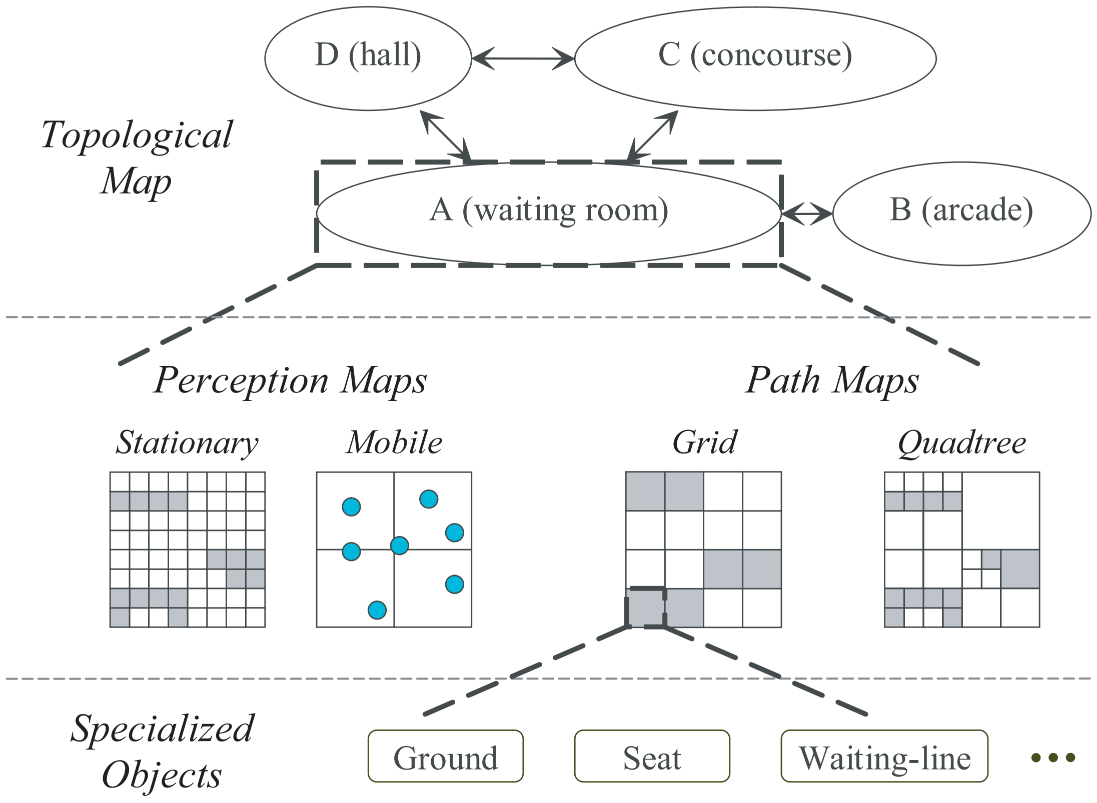


Figure 3.2: Hierarchical environment model.

all of the objects occupying its cellular area. Each cell of the mobile grid map stores and updates identifiers of all the agents currently within its cellular area. Since it serves simply to identify the nearby agents, rather than to determine their exact positions, it employs cells whose size is commensurate with the pedestrian's visual sensing range (currently set to 5 meters). The perception process will be discussed in more detail in Section 3.2.2.

The path maps include a quadtree map which supports global, long-range path planning and a grid map which supports short-range path planning. Each node of the quadtree map stores information about its level in the quadtree, the position of the area covered by the node, the occupancy type (ground, obstacle, seat, etc.), and pointers to neighboring nodes, as well as information for use in path planning, such as a distance variable (i.e., how far the node is from a given start point) and a congestion factor (the portion of the area of the node that is occupied by pedestrians). The quadtree map supports the execution of several variants of the A* graph search algorithm, which are employed to compute quasi-optimal paths to desired goals (cf. [Botea et al. 2004]). Our simulations with numerous pedestrians indicate that the quadtree map is used for planning about 94% of their paths. The remaining 6% of the paths are planned using the grid path map, which also supports the execution of A* algorithm and provides detailed, short-range paths to goals in the presence of obstacles, as necessary. A typical example of its use is when a pedestrian is behind a chair or bench and must navigate around it in order to sit down.

The specialized objects at the lowest level of the environment hierarchy are able to provide answers to queries that can not be handled directly by perception maps. They make it easy for behavioral controllers to acquire higher level perceptual information from the virtual world.

We will present the full details of our environment model and path planning algorithms in Chapters 6 and 7, respectively. Our Penn Station environment model is efficient enough to support the real-time (30fps) simulation of about 1400 pedestrians on a 2.8GHz Xeon PC with 1GB memory. More detailed performance statistics are presented in Chapter 8.

3.2 Virtual Pedestrian Model

Analogous to real humans, our synthetic pedestrians are fully autonomous. They perceive the virtual environment around them, analyze environmental situations, make decisions and behave naturally. As described in Section 1.2, our autonomous human characters are structured in accordance with a hierarchical character model similar to the one advocated by [Funge et al. 1999] for an aquatic "merman" and dinosaurs. Progressing through levels of abstraction up the "modeling pyramid", our model incorporates appearance, motor, perception, behavior, and cognition sub-models, each of which will be summarized in the following sections.

3.2.1 Appearance and Motor Control

DI-Guy: Human Appearance and Movement

As an implementation of the appearance and motor levels of the pyramid, we employ a human animation package called *DI-Guy*, which is commercially available from Boston Dynamics Inc. [Koechling et al. 1998; Boston Dynamics, Inc. 2004]. *DI-Guy* provides a variety of textured geometric models that portray different people. These character models are capable of basic motor skills, such as strolling, walking, jogging, sitting, etc., implemented using conventional IK and motion capture techniques. *DI-Guy* is intended as an application that enables users to script the actions of human characters manually in space and time. To facilitate this task, it provides an interactive scripting interface called *DI-Guy Scenario*, which we do **not** use in our work. However, it also provides an SDK that enables each character's motor repertoire to be controlled by external user-specified $C/C++$

programs. We build upon this interface our own control programs at the motor, perceptual, behavioral, and cognitive levels.

Emulating the natural appearance and movement of human beings is a highly challenging problem and, not surprisingly, *DI-Guy* suffers from several limitations. In particular, the *DI-Guy* geometric human models are insufficiently detailed to provide pleasing renderings of humans for close-up viewing. We have made no attempt to ameliorate this situation as the *DI-Guy* appearance code is not open-source and remains a proprietary "black box". More importantly, although the most advanced *DI-Guy* characters have reasonably broad action repertoires, they cannot synthesize the full range of motions needed in a busy urban environment that produces frequent close encounters between independently locomoting pedestrians. Specifically, *DI-Guy* characters are limited in how tightly they can execute turns and how quickly they can change locomotion gaits and speeds, sometimes requiring up to a couple of seconds before completing a transition, depending on the exact instant within the locomotion cycle. Fortunately, the most recent release of the software package provides a *Motion Editor* interface that enables users to modify and supplement the motion repertoires of *DI-Guy* characters. We have used this interface to decompose *DI-Guy* motions, which typically span one entire cycle of a gait, into more elementary motions that include only single steps and other motion primitives. The decomposition enables the *DI-Guy*-based pedestrians to make faster transitions so that they can better deal with highly dynamic urban environments.

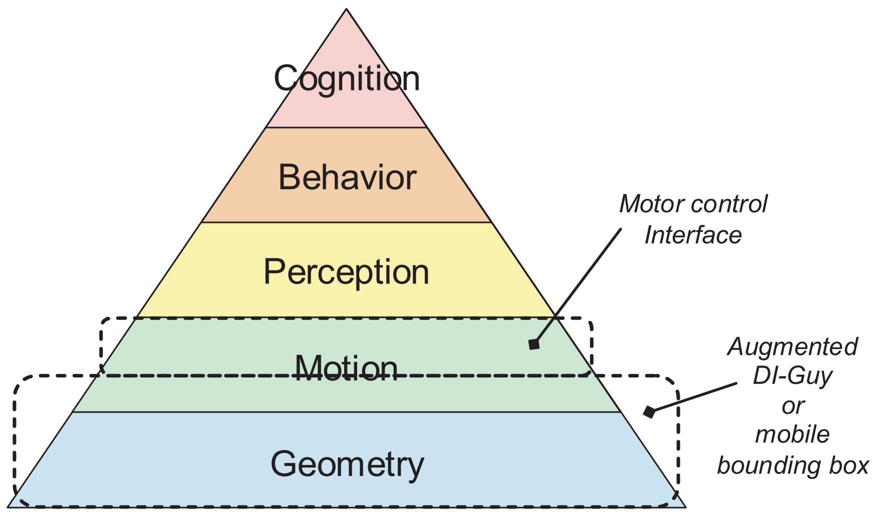


Figure 3.3: Appearance, movement and motor control.

Motor Control Interface

We have implemented a motor control interface (see Figure 3.3) between the low level kinematic layer of DI-Guy, and the higher-level behavioral controllers, which will be discussed later in this chapter and presented in full detail in Chapter 4. This interface accepts motor control commands from behavior modules, selects the appropriate direction, speed, acceleration, and gait for a pedestrian in accordance with its kinetic limits, and updates the posture and position of the pedestrian using the low-level kinematic control mechanism of *DI-Guy*. As motor control commands issued by the higher-level behavior system are not necessarily physically achievable, our motor control layer is responsible for verifying and correcting them (e.g., large magnitude speeds or accelerations are trimmed). Given a desired locomotion speed, the motor controller will choose the gait that has the closest speed and, if necessary, call upon the kinematic layer to make as smooth as possible a transition from the current gait to the new one.

Hence, this motor control layer is a seamless interface that hides the details of the underlying kinematic layer from the higher-level behavior modules, enabling the latter to be developed more or less independently. Despite its limitations, we have found that, at least for the time being, our "Augmented *DI-Guy*" low-level humanoid model adequately supports our research into autonomous human animation through perceptual, behavioral, and cognitive modeling, as we shall describe in subsequent chapters. It is important to note, however, that our higher-level modeling abstractions are designed to be more or less independent of the lower levels; hence, any other suitable low-level API can easily replace *DI-Guy* in our future work. As case in point, in addition to "Augmented *DI-Guy*", we have implemented a simpler kinematic layer for testing our higher level algorithms, where each pedestrian is represented by a simplistic mobile bounding box (see Figure 3.3). We have used both the Augmented *DI-Guy* and the "mobile bounding box" to produce the various animated demonstrations of our pedestrian simulation system.

3.2.2 Perception

In a highly dynamic virtual world, an autonomous intelligent character is confronted with two classes of sensory problems. First, to interact with its environment, it must have a keenly perceptive regard for the external world. Second, as a virtual creature, it should also be able to experience internal stimuli, that convey its tendencies, emotions, and desires, that once perceived can trigger fulfilling behaviors. In this chapter, we will focus on sensory processes of these two types.

External World

The environment model presented in Chapter 6 is used extensively for pedestrians to perceive the external world. Through various perceptual queries, the hierarchical environment model can provide not only the raw sensed data, but sometimes the interpretation of the perceived situation as well, which is more important and useful to a pedestrian. Next, we summarize the major perceptual processes whose goals are to provide answers to three questions, as follows:

Where Am I? In real life, people are capable of using surrounding objects and spatio-temporal context together with optional prior knowledge to determine their current position, which can be either relative or absolute. Despite its seeming simplicity, such human ability is not easy to simulate. This problem, which is referred to as map building and localization, has been a long-standing research topic in robotics [DeSouza and Kak 2002]. In a virtual world such as the one we have for our pedestrians, however, we can simply query the environmental model in accordance with the pedestrian's absolute position and orientation. Additional queries are also designed to allow a pedestrian to "recognize" in which region he is and what type of region it is (a staircase, a passageway, a portal, etc). Such information will later be used to decide the appropriate navigational behaviors employed (such as passageway navigation).

What is Around Me? This question is answered by the perceptual queries of static and mobile objects described in Section 6.2.

Static Objects: In sensing static objects, a pedestrian emits a probing eye-ray to detect obstacles in a specific direction. This technique is also exploited for visibility (or direct-reachability) check—a visual test frequently used in navigational behaviors—in which a pedestrian tests whether obstacles exist between him and his target.

Dynamic Objects: Dynamic objects in our simulations are mostly pedestrians. Obtaining perceivable information of neighboring pedestrians is crucial to collision avoidance behaviors to be described in the next chapter. Such information not only includes a nearby pedestrian's current position, orientation, velocity, acceleration, turning speed, etc., but also includes his observable intention in the next move, as pointed out by Goffman [1971]. The latter can help others estimate the future trajectory of this pedestrian, thus predicting collisions more accurately. For our pedestrians, a limited number of nearby humans can be detected by interrogating tiers of neighboring perception map cells and further information can be obtained by simply querying each individual's "observable" data.

What is Happening There? In this question, "there" can be anywhere within a pedestrian's visual range. Such questions are relevant when a pedestrian needs a high level interpretation of a perceived situation before making a decision. For instance, when a tired pedestrian sees a chair and another pedestrian, he should be able to tell whether that pedestrian is sitting on the chair, is going to sit on the chair, is going to leave the chair, or is just passing by and is uninterested in the chair. Different interpretations will probably lead to different decisions. In real life, such interpretations may require continuous observation for several seconds and abilities such as shape estimation, pattern recognition, etc. As they are nontrivial problems, we believe that in our system it is not affordable nor necessary to let every pedestrian do it for itself. Instead, we use specialized objects (see Section 6.4) within the environment model to keep track of evolving situations that may be important to pedestrians' decision making. Such situations are usually found at places where resources (e.g., space, transaction points, etc.) are limited and pedestrians may need to compete for them, such as:

- seats, chairs and benches;
- vending machines;
- ticket booths;
- entrances and exits;
- stairs;
- narrow passageway or portals; and
- the artist performance area.

With the help of specialized objects, pedestrians can efficiently determine "what is happening there" by querying the environment model.

Internal States

The evolution of the world is also indirectly affected by the evolution of an individual. The latter is in turn reflected by the changing of the individual's internal states, among which are its physiological, psychological or social needs. To encode them, we maintain a set of internal mental state variables for each pedestrian, including tiredness, thirst, curiosity, the propensity to get attracted by performers, the need to acquire a ticket, etc. The values of these variables are initialized either randomly or through a configuration file at the beginning of a simulation, and are updated at each simulation step in accordance with the nature of the variables. Thirst, for instance, increases continuously, while the propensity to get attracted by performers increases and decreases now and then.

3.2.3 Behavior

Modeling of realistic human behavior in general is a challenging task. Considerable literature in psychology, ethology, artificial intelligence, robotics, and artificial life is devoted to the subject. Following [Tu and Terzopoulos 1994], we adopt a bottom-up strategy that uses primitive reactive behaviors as building blocks that in turn support more complex motivational behaviors, all controlled by an action selection mechanism.

We developed six key reactive behavior routines that cover virtually all the local obstacle situation that a pedestrian can encounter. Given that a pedestrian possesses a set of motor skills, such as standing still, moving forward, turning in several directions, speeding up and slowing down, etc., these routines are responsible for initiating, terminating, and sequencing the motor skills on a short-term basis guided by sensory stimuli and internal percepts. The timely and accurate perceptual interpretation of afferent sensory data is crucial for meaningful behavioral response.

While the reactive behaviors enable pedestrians to move around freely, almost always avoiding collisions, navigational and motivational behaviors enable them to go where they desire, which is crucial for pedestrians.

When the value of a mental state variable exceeds a specified threshold, an action selection mechanism chooses the appropriate behavior to fulfill the need. Once a need is fulfilled, the value of the associated internal state variable begins to decrease asymptotically to zero.

We classify pedestrians in the virtual train station environment as commuters, tourists, law enforcement officers, performers, etc. Each pedestrian type has an associated action selection mechanism with appropriately set behavior-triggering thresholds associated with mental state variables. For instance, law enforcement officers on guard will never attempt to buy a train ticket and commuters will never act like performers.

Chapter 4 will present the details of the pedestrian behavioral model.

3.2.4 Cognition

At the highest level of autonomous control, a cognitive model [Funge et al. 1999] is responsible for creating and executing plans, as is necessary for a deliberative human agent. Such a model must be able to make reasonable global navigation plans in order for a pedestrian to travel purposefully and with suitable perseverance between widely separated regions of the environment. During the actual navigation, however, the pedestrian must have the freedom to decide whether or not and to what extent to follow the plan, depending on the real-time situation. On the other hand, in a highly dynamic environment such as a train station, the pedestrian also needs the ability to decide whether and when a new plan is needed. These decisions require a proper coupling between the behavioral layer and cognitive layer.

Chapter 5 will present the details of the pedestrian cognitive model.

Chapter 4

Behavioral Control

"(Julian) Huxley likened the human to a ship also commanded by many captains, all of whom stay on the bridge continuously, each giving his own commands without consideration of any of the others. Sometimes the conflict caused by their countermanding commands leads to complete chaos, but sometimes they jointly succeed in choosing a course which none of them would have arrived at alone. "

(K. Lorenz. 1981. The Foundations of Ethology. Page 242.)

Realistic human behavioral modeling, whose purpose is to link perception to appropriate actions in an intelligent virtual human, is an enormous hurdle since, even for pedestrians, the complexity of any substantive behavioral repertoire is high. In this chapter, we first present the architecture of our behavioral model, and then fill in the details of the various behavioral routines and control mechanisms.

4.1 Human Behavior Modeling Methodology

Ever since people started to record their thoughts, the behaviors of living systems have been a topic of study for both natural and social scientists, as well as philosophers. Evidence from a vast number of observations and experiments have led scientists to believe that the control mechanism behind observable behaviors takes hierarchical form in its organizational structure [Tinbergen 1951; Eibl-Eibesfeldt 1975] and different levels in this hierarchy can be activated either sequentially or in parallel (as pointed out by Huxley's parable above) [Lorenz 1981]. Drawing upon these hypotheses and speculation, researchers in computer animation and robotics have proposed various architectures for building autonomous behavioral systems [Tu and Terzopoulos 1994; Maes 1991; Brooks 1986].

We design our behavioral architecture in accordance with these pioneering works (see Figure 4.1). In a bottom-up strategy, we start with a collection of carefully chosen primitive reactive behaviors at the bottom, and use lower level behaviors as building blocks that in turn support more complex higher level behaviors. During execution, we employ several activation schemes at different levels to

- either selectively chain together simple behavioral modules in order to achieve complex goals,
- or exclusively activate one behavior and inhibit others on the same level according to external stimuli and internal percepts,
- or effectively incorporate several modular controllers simultaneously to allow a combined response to complex environmental situations.

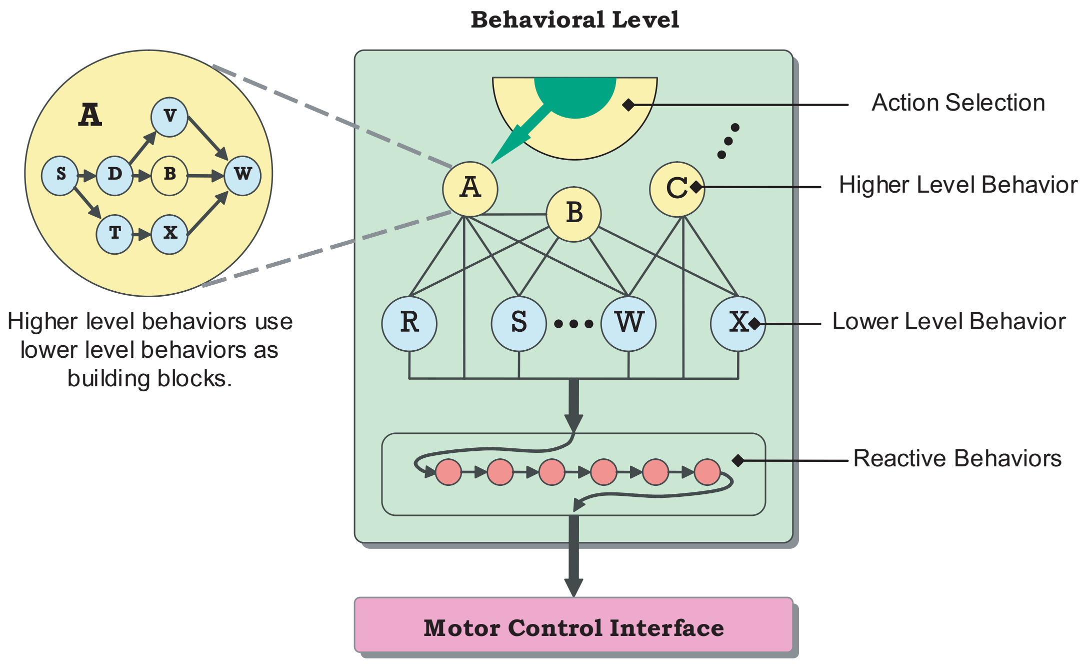


Figure 4.1: Behavioral architecture. Despite the hierarchical structure, behavioral execution takes three forms in our system: (1) higher level routines sequentially draw upon simple ones at lower levels; (2) an action selection mechanism activates high level behaviors one at a time; (3) at every simulated human step, usually not one but multiple reactive behaviors get activated.

In particular, the second scheme is used in a high level action selection mechanism, while the third is for combining primitive reactive behaviors. We now start our description with reactive behaviors at the lowest level and will move upward to navigational and motivational behaviors.

4.2 Basic Reactive Behaviors

"Take for example, techniques that pedestrians employ in order to avoid bumping into one another. These seem of little significance. However, there are an appreciable number of such devices; they are constantly *in*

use and they cast a pattern on street behavior. Street traffic would be a shambles without them."

(E. Goffman. Relations in public: microstudies of the public order, Page 6.)

Reactive behaviors appropriately connect perceptions to immediate actions. For our synthetic pedestrians, we have developed six key reactive behavior routines, each suitable for a different set of situations related to the navigation of a densely populated and highly dynamic environment. We will first explain each of these behavior routines in detail, and then explain how they are combined.

4.2.1 Six Key Routines

Routine A: Avoid stationary obstacles ahead (Figure 4.2). With the help of perception map discussed in Section 6.2, a virtual pedestrian can perceive objects within a predefined field of view and is able to determine the identity of and estimate the distance to these perceived objects. If there is a nearby obstacle in the direction of locomotion, a set of lateral directions within a specified angular extent (currently set to 90 degrees) to the left and right are tested for obstacles and the less cluttered direction is chosen. If none of the alternative directions seems better than the current heading, the pedestrian will slow down and perform the aforementioned test again using larger lateral search angles (currently set to 150 degrees). This behavioral response mimics that of a real pedestrian, who will normally slow down and turn his head to look around before proceeding when he encounters a tough (set of) obstacle(s).

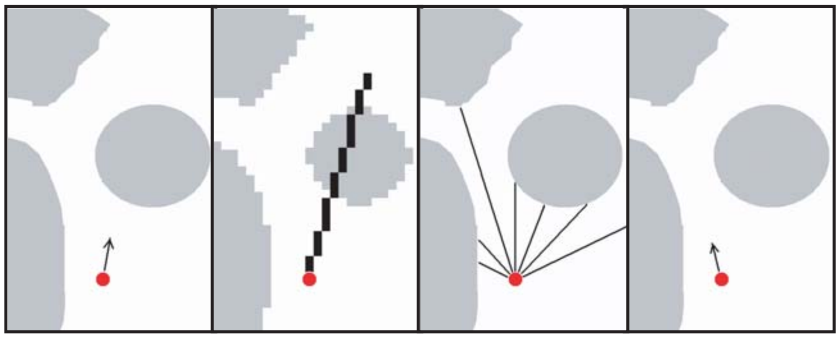


Figure 4.2: Routine *A*: avoid stationary obstacles. 1) Original travel direction. 2) Detect obstacle ahead by examining grid entries along the rasterized eye ray. 3) Perceive situation around by shooting out more eye rays (rasterization not shown). 4) Change direction and move on.

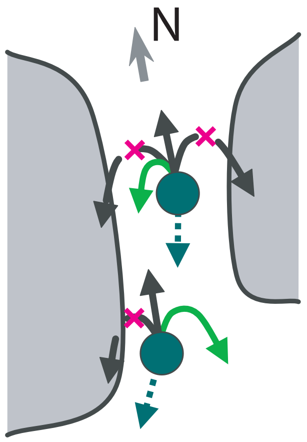


Figure 4.3: Routine *B*: avoid stationary obstacles in a big turn. Initially heading north, the two pedestrians want to turn southward. They use routine *B* and pick the best turning curves.

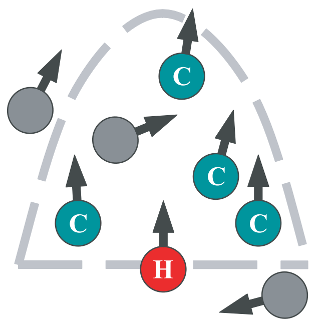


Figure 4.4: Routine C: maintain separation in a moving crowd. Other pedestrians in H's front cut-off parabola and are traveling in similar directions as H (labeled "C") are considered to be in H's *temporary crowd*.

Routine B: Avoid stationary obstacles in a complex turn (Figure 4.3). A pedestrian employs this routine whenever it needs to make a turn that cannot be finished in one step. In this routine, turns with increasing curvatures are considered in both directions, starting with the side that permits the smaller turning angle, until a collision-free turn is found. If the surrounding space is too cluttered, the curve is likely to degenerate causing the pedestrian to stop and turn on the spot. The turn test is implemented by checking sample points along a curve with interval equal to the distance of one step of the pedestrian moving with the anticipated turn speed.

Routine C: Maintain separation in a moving crowd (Figure 4.4). For a pedestrian H, other pedestrians are considered to be in H's *temporary crowd* if they are moving in a similar direction to H and are situated within a parabolic region [Goffman 1971] in front of H defined by

$$y = -\frac{4}{R}x^2 + R, \tag{4.1}$$

where R is the sensing range, y is oriented in H's forward direction and x is oriented laterally. To maintain a comfortable distance from each individual H_i in this temporary crowd, a directed repulsive force (cf. [Helbing and Molnar 1995]) given by

$$\vec{f_i} = r_i \times \frac{\vec{d_i}/|\vec{d_i}|}{|\vec{d_i}| - d_{min}} \tag{4.2}$$

is exerted on H, where $\vec{d_i}$ is the vector separation of H_i from H, and d_{min} is the predefined minimum distance allowed between H and other pedestrians (usually $2.5 \times H$'s bounding box size). The constant r_i is H_i's perceived "repulsiveness" to H (currently set to -0.025 for all pedestrians). The repulsive acceleration due to H's temporary crowd is given by

$$\vec{a} = \sum_i \frac{\vec{f_i}}{m}, \tag{4.3}$$

where m is the inertia of H. The acceleration vector is decomposed into a forward component $\vec{a_f}$ and a lateral component $\vec{a_l}$. The components $\Delta t \times \vec{a_f}$ and $\Delta t \times c_i \times \vec{a_l}$ are added to H's current desired velocity. The crowding factor c_i determines H's willingness to "follow the crowd", with a smaller value of c_i giving H a greater tendency to do so (currently $1.0 \leq c_i \leq 5.0$).

Routine *D*: Avoid oncoming pedestrians. In a crowded public area, such as a plaza, where pedestrians are moving in various directions, the likelihood of collisions increases dramatically. To mitigate the situation, pedestrian H estimates its own velocity $\vec{v}$ and the velocities $\vec{v_i}$ of nearby pedestrians C_i. It then considers two types of threats. By intersecting its own linearly extrapolated trajectory T

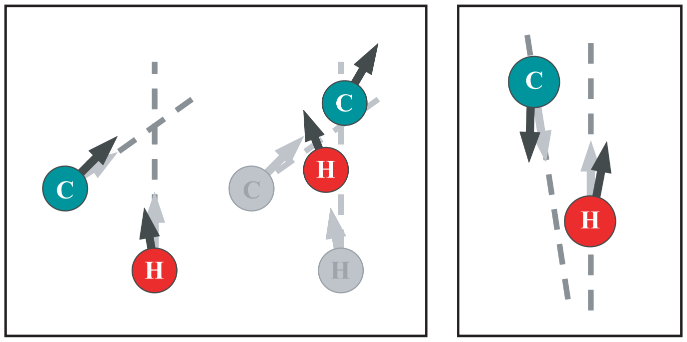


Figure 4.5: Routine D: avoid oncoming pedestrians. 1) To avoid *cross collision*, H slows down and turns toward C while C does the opposite (left) until collision is cleared (middle). 2) To avoid *head-on collision*, both pedestrians turn slightly away from each other (right).

with the trajectories T_i of each of the C_i, pedestrian H identifies potential collision threats of the first type: *cross collision* (Figure 4.5, left). In the case where trajectories of H and C_i are almost parallel and will not intersect imminently, a *head-on collision* (Figure 4.5, right) may still occur if their lateral separation is too small; hence, H measures its lateral separation from oncoming pedestrians. The details are given in Table 4.1.

Note that each pedestrian evaluates the possibility of collision with respect to the size of its own bounding box $B(H)$ and the size of the bounding boxes $B(C_i)$ of the nearby pedestrians. Bounding box sizes vary from 0.3 to 0.6 meters for different pedestrians, depending on body size. Among all collision threats, H will pick the oncoming pedestrian C^* that poses the most imminent one. If C^* poses a *head-on collision* threat, H will turn slightly away from C^*. If C^* poses a *cross collision* threat, H will estimate who will arrive first at the anticipated intersection point p. If pedestrian H determines that it will arrive sooner at p than C^*, it will increase its speed and turn slightly away from C^*; otherwise, it will decrease its speed and

```
if $T$ and $T_i$ intersect then
    let $p$ be the anticipated point of intersection
    let $t$ be $H$'s travel time to $p$
    let $t_i$ be $C_i$'s travel time to $p$
    $\Delta t_{safe} = (B(H) + B(C_i))/min(|v|, |v_i|)$
    if $|t - t_i| < \Delta t_{safe}$ then
        return "$C_i$ is a cross collision threat"

if $T$ and $T_i$ almost parallel then
    if the lateral separation $< B(H) + B(C_i)$ then
        return "$C_i$ is a head-on collision threat"

return "$C_i$ is not a collision threat"
```

Table 4.1: Algorithm for checking threats.

turn slightly towards C^*. This behavior will continue for several footsteps, until the potential collision has been averted.

Routine E: Avoid dangerously close pedestrians. This is the fail-safe behavior routine, reserved for emergencies due to the occasional failure of Strategies C and D, since in highly dynamic situations predictions have a nonzero probability of being incorrect. Once a pedestrian perceives another pedestrian within his forward safe area (Figure 4.6), it will resort to a simple but effective behavior–brake as soon as possible to a full stop, then try to turn to face away from the intruder, and proceed when the way ahead clears.

Routine F: Verify new directions relative to obstacles. Since the reactive behavior routines are executed sequentially as will be explained momentarily in

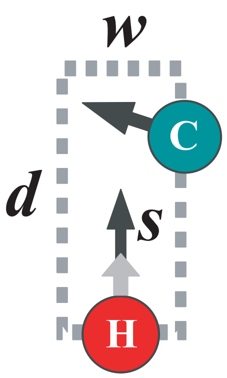


Figure 4.6: Routine E: avoid dangerously close pedestrians. H's forward safe area is shown in dotted line. Both the width w and the depth d depend on H's bounding box size. In addition, d is also determined by H's current speed s (black arrow). Here H encounters C and slows down (light gray arrow) to let C pass.

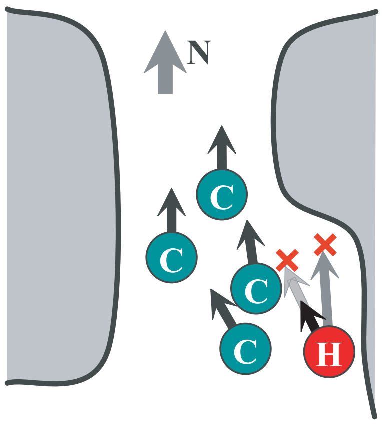


Figure 4.7: Routine F: verify new directions relative to obstacles. When confronted by both static and dynamic threats, H abandons the direction (dark gray arrow) issued by Routine C, uses the direction (light gray arrow) picked by Routine A and slow down (black arrow) to let others to pass before proceed.

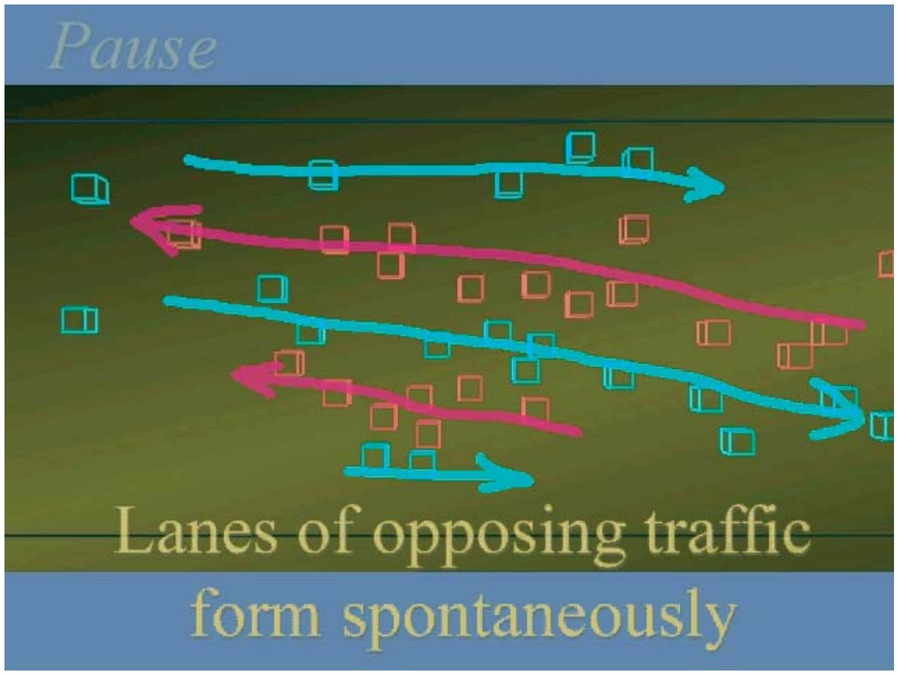


Figure 4.8: Lanes of opposing traffic form in a busy area. (Snapshot from animation.)

Section 4.2.2, motor control commands issued by Routines C, D or E to avoid pedestrians may counteract those issued by Routines A or B to avoid obstacles, thus steering the pedestrian towards obstacles again. To avoid this, the pedestrian checks the new direction against surrounding obstacles once more. If the way is clear, it proceeds. Otherwise, the original direction issued by either the higher-level path-planning modules or by Routine A, whichever was executed most recently prior to the execution of Routine F, will be used instead. However, occasionally this could lead the pedestrian toward future collisions with other pedestrians (Figure 4.7) and, if so, it will simply slow down to a stop, let those threatening pedestrians pass, and proceed.

Some remarks regarding the above routines are in order: The intuition behind the strategy of Routine F is that whenever a pedestrian is faced with threats of collision from both static and mobile obstacles, it is a bad idea to steer toward the former, as that will definitely lead to a collision. Instead, the pedestrian should hesitate until the mobile collision threats pass and then proceed in the direction that is free of static obstacles. Obviously, the fail-safe strategy of Routine E suffices *per se* to avoid nearly all collisions between pedestrians. However, our experiments show that in the absence of Routines C and D, Routine E makes the dynamic obstacle avoidance behavior appear awkward–pedestrians stop and turn too frequently and they make slow progress. As we enable Routines C and D, the obstacle avoidance behavior looks increasingly more natural. Interesting multi-agent behavior patterns emerge when all the routines are enabled. For example, pedestrians will queue to go through a narrow portal. In a busy area, lanes of opposing pedestrian traffic will tend to form spontaneously after some time passes (see Figure 4.8), since this pattern enables more pedestrians to make faster progress. These self-organizing patterns of movement, which the virtual pedestrians are not explicitly programmed to form, resemble those of real human crowds in urban environments.

4.2.2 Combining the Routines

A remaining issue is how best to activate the aforementioned six reactive behavior routines. Since the situation encountered by a pedestrian is always some combination of the six key situations that are covered by the behavior routines, it is necessary to employ each applicable routine. Therefore, we have chosen to activate the routines in a sequential manner, as illustrated in Figure 4.9, giving each the

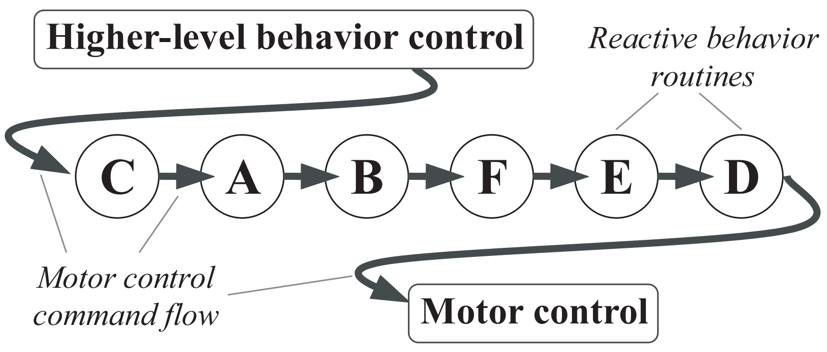

Higher-level behavior control
Reactive behavior routines
C
A
B
F
E
D
Motor control command flow
Motor control

Figure 4.9: Flow of control for reactive behaviors. Shown here is the best permutation order $C - A - B - F - E - D$.

chance to alter the currently active motor control command, comprising speed, turning angle, etc. For each routine, the input is the motor command issued by its predecessor, either a higher-level behavior module (possibly goal-directed navigation) or another reactive behavior routine. The sequential flow of control affords later routines the advantage of overwriting motor commands issued by earlier routines (cf. [Brooks 1986; Brooks 1995]), but this may cause the pedestrian to ignore some aspect of the situation covered by those routines, resulting in a collision. The problem can be mitigated by finding a "best" permutation ordering for the six routines (cf. [Reynolds 1993]). We have run many extensive simulations (with each one longer than 20 minutes in virtual time) in the Penn Station environment with different numbers of pedestrians (333, 666, and 1000), exhaustively evaluating the performance of all 720 possible permutation orderings. Permutations that result in fewer collisions and more progress are considered better. Finally, the best permutation of the reactive behavior routines that we have found is $C-A-B-F-E-D$ as shown in Figure 4.9.

Appendix A presents the details regarding the exhaustive search and an analysis of the results.

4.3 Navigational and Motivational Behaviors

While the reactive behaviors enable pedestrians to move around freely, almost always avoiding collisions, navigational and motivational behaviors enable them to go where they desire, which is crucial for pedestrians.

As we must deal with online simulations of numerous pedestrians within large, complex environments, we are confronted with many navigational issues, including the speed and scale of path planning, the realism of actual paths taken, and pedestrian flow control through and around bottlenecks. We have found it necessary to develop a number of novel navigational behavior routines to address these issues. These behaviors rely in turn on a set of conventional navigational behavior routines. The latter include moving forward, wandering about, turning in place, steering towards a target, proceeding towards a target, and arriving at a target (see [Reynolds 1999] for detailed description of these primitive behaviors).

4.3.1 Guided Navigation

The realism of synthetic human navigation depends on the presence of rational behavior details at different levels (or scales). Globally, a pedestrian should be able to make reasonable travel plans. As will be explained in Chapter 5, such plans usually have certain favorable properties, such as taking time-saving paths. During navigation, however, the pedestrian shall have the freedom to decide whether or not, and to what extent to follow the plan, depending on the real-time situation.

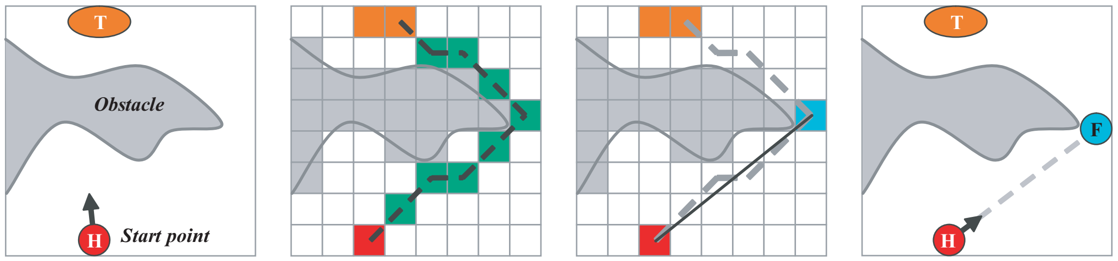


Figure 4.10: Perception-guided navigation. (1) To reach target T, (2) pedestrian H will plan a jagged path on a path map (either grid or quadtree), (3) pick the farthest visible point (in blue) along the path as an intermediate target (marked F in (4)) and (4) proceed toward it.

More attention shall be drawn to the local (in both space and time) settings of the dynamic environment in order to keep the pedestrian safe while still making progress. In this section, we present behavior routines that use a planned path and a selective set of reactive behaviors as the global and local guide, respectively, and combine them together to make navigation as natural as possible.

Perception-guided navigation among static obstacles. Given a path P (the global planning of paths will be explained in Chapter 7), a farthest visible point p on P—*i.e.*, the farthest point along P such that there is no obstacle on the line between p and the pedestrian's current position—is determined using perceptual queries and set as an intermediate target (see Figure 4.10). As the pedestrian progresses toward p, it may detect a new farthest visible point that is even further along the path. This enables the pedestrian to approach the final target in a natural, incremental fashion. During navigation, motor control commands for each footstep are verified sequentially by the entire set of reactive behavior routines in their aforementioned order so as to keep the pedestrian safe from collisions.

Detailed "arrival at target" navigation. Before a pedestrian arrives at a target, a detailed path will be needed if small obstacles intervene that are not resolved on ordinary path maps. Such paths can be found on a fine-scale grid path map. Unlike the previous situation, a pedestrian will follow the detailed path more strictly as it approaches the final target, because accuracy becomes an increasingly important factor in the realism of the navigation as the distance to the target diminishes. As some part of an obstacle may also be a part of the target or be very close by (e.g., the front part of a vending machine, or the back of a bench in which the pedestrian wants to sit), indiscriminately employing reactive behaviors for static obstacle avoidance (Routines A and B) will cause the pedestrian to avoid the obstacle as well as the target, thereby hindering or even preventing the pedestrian from reaching the target. One possible solution is to remove the target portion of the obstacle from the perception map before applying the routines, but this becomes difficult to manage in an online simulation. We choose a simpler alternative—temporarily disable Routines A and B and let the pedestrian accurately follow the detailed path, which already avoids obstacles. Note that the other reactive behaviors, Routines C, D, and E, remain active, as does Routine F, which will continue to play the important role of verifying that modified motor control commands never lead the pedestrian into obstacles.

4.3.2 Bottleneck Flow Control

"The members of an orderly community do not go out of their way to aggress upon one another. Moreover, whenever their pursuits interfere, they make the adjustments necessary to escape collision and make them

according to some conventional rule."

(Edward Alsworth Ross. 1908. Social Control, Page 1. [Ross 1908])

In the Penn Station environment, stairways from the concourse to train platforms are less than 1.8 meters in width, which allows only two pedestrians to advance comfortably side by side. The two major portals connecting the main waiting room and the concourses are less than 2.8 meters wide, making it difficult for four pedestrians to pass simultaneously. In a space that is expected to accommodate the hustle and bustle of hundreds or even thousands of pedestrians, these narrow passageways can easily become bottlenecks that will cause queuing crowds to accumulate. The crowds will hinder opposing traffic, exacerbating the situation, and pedestrians may experience lengthy delays.

To our knowledge, available techniques cannot completely tackle the problem. The queuing behavior introduced by Reynolds [1999] is not well-suited to situations involving two-way traffic, because in narrow passageways oncoming pedestrians often cause one another to slow down or stop. In the cramped space, this quickly leads to blockage. Self-organization, as discussed by Helbing and Molnar [1995] and by us in the last section, yields lanes of opposing traffic which increases throughput, but it takes time and space to manifest itself. A crowd will quickly grow beyond control in narrow passageways, well before self-organization can help. Global crowd control techniques, such as those proposed by Musse and Thalmann [2001], are useful for directing a given group of agents traveling around as a whole, but they lack the flexibility to handle highly dynamic groups of individual autonomous pedestrians.

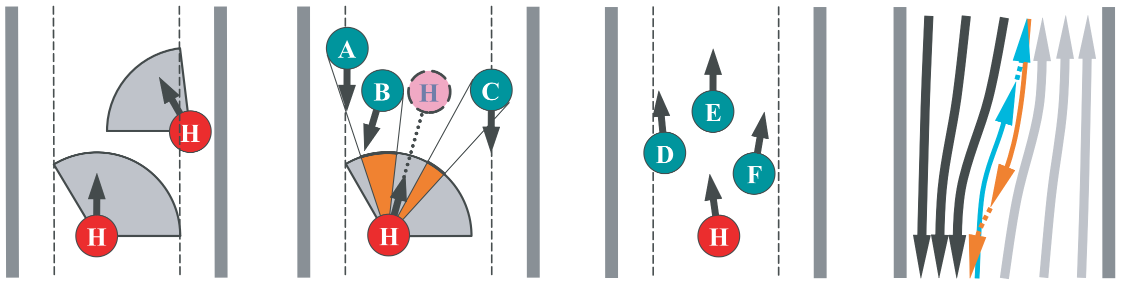


Figure 4.11: Passageway navigation. (1) Two imaginary boundaries (dashed lines) and the safety fan. (2) Pedestrian H searches for a safe direction interval when confronted by oncoming traffic. (3) Spread out when no oncoming traffic is observed. (4) Typical flow of pedestrians in a passageway—big flows on the sides with small unblocking streams intermingling in the middle.

In our solution, we have determined that the application of two behavioral heuristics maximizes pedestrian flow in bottleneck situations. First, pedestrians inside a bottleneck should move with on-going traffic while trying not to impede oncoming traffic. Second, all connecting passageways between two places should be used in balance. These two behaviors are detailed next.

Passageway navigation: Inside a narrow passageway, real people demonstrate different behavior patterns under different conditions. If all people are traveling in the same direction, they will tend to spread out in order to see further ahead and maximize their pace. However, if people are in busy two-way traffic, they will compromise and quickly form opposing lanes of pedestrian traffic to maximize the throughput. When inside a passageway, our pedestrians employ a similar behavior:

First, two imaginary boundaries are computed parallel to the walls of the passageway with an offset of about half the pedestrian H's bounding box size (Figure 4.11(1)). Space between these boundaries is considered to be safe. Hence, restricting H's travel direction within a safety fan, as shown in the figure, guarantees that H stays clear of the walls.

Second, if H detects that its current direction is blocked by oncoming pedestrians, it will search within the safety fan for a safe interval to get through (Figure 4.11(2)). The search starts from H's current direction and continues clockwise. During the search, H attempts to steer to the right of every blocking on-comer, testing whether there is enough room to get through. If the search succeeds, H will move in the safe direction found. Otherwise, H will slow down and proceed in the rightmost direction within the safety fan. This strategy allows non-blocking traffic to intermingle without resistance. However, in a manner that reflects the preference of real people in many countries, a virtual pedestrian will tend to squeeze to the right if it is impeding or impeded by oncoming traffic (Figure 4.11(4)).

Finally, pedestrians apply reactive behavior Routine C described in the previous section in order to maintain a safe separation between one another in the moving crowd. In the event that no more oncoming traffic is observed, pedestrians increase their crowding factor c_i (which lowers the willingness to follow the crowd—see section 4.2.1) to spread out, allowing faster walkers to overtake slower ones (Figure 4.11(3)). Upon encountering oncoming traffic, they will decrease their crowding factor, which will draw the crowd more tightly, making more space for oncoming pedestrians.

Passageway selection: In most urban environments there exist several options for transiting from one location to another. Real people are usually motivated enough to weigh the options and choose a route that promises to maximize their pace. Their choice depends on both personal preferences and the real-time situ-

```
Given a list of candidate passageways $P_i$, together with the
density $D_i$ (unit: $\#pedestrians/m^2$) and proximity $T_i$ (unit:
meters) of the entrance of $P_i$ that closer to the pedestrian:

let $c =$ the index of the passageway picked in previous step,
        or NULL if none is picked previously
let $S_i = T_i + 25 \times D_i$, a value that reflects $P_i$'s situation
        (Smaller $S_i$ indicates a better situation.)
let $m =$ the index of the passageway that has the smallest $S_i$

if ($c$ is NULL)
  then pick $P_m$        (no previous choice, so pick the best)
  else if ($S_m < S_c - 5.0$)
    then pick $P_m$      (a significantly better one found, so pick it)
    else pick $P_c$      (stick to the previous choice)
```

Table 4.2: Algorithm to pick a passageway.

ation in and around those access facilities. Likewise, our pedestrians will assess the situation around stairways and other passageways, pick a preferred one based on proximity and density of pedestrians near it, and proceed toward it. As crowd density is always changing, however, our pedestrian may be motivated to modify its choices too frequently. Hence, our selection routine dictates that the pedestrian maintain its original choice unless a significantly more favorable traffic density condition develops in a different passageway (see Table 4.2 for algorithm of the behavior). This behavior, although executed at the level of an individual, has a global effect, balancing the loads at different passageways.

With the above two passageway behaviors, we are able to increase the number of pedestrians within the Penn Station model from under 400 to well over 1000 without any long-term blockage in bottlenecks.

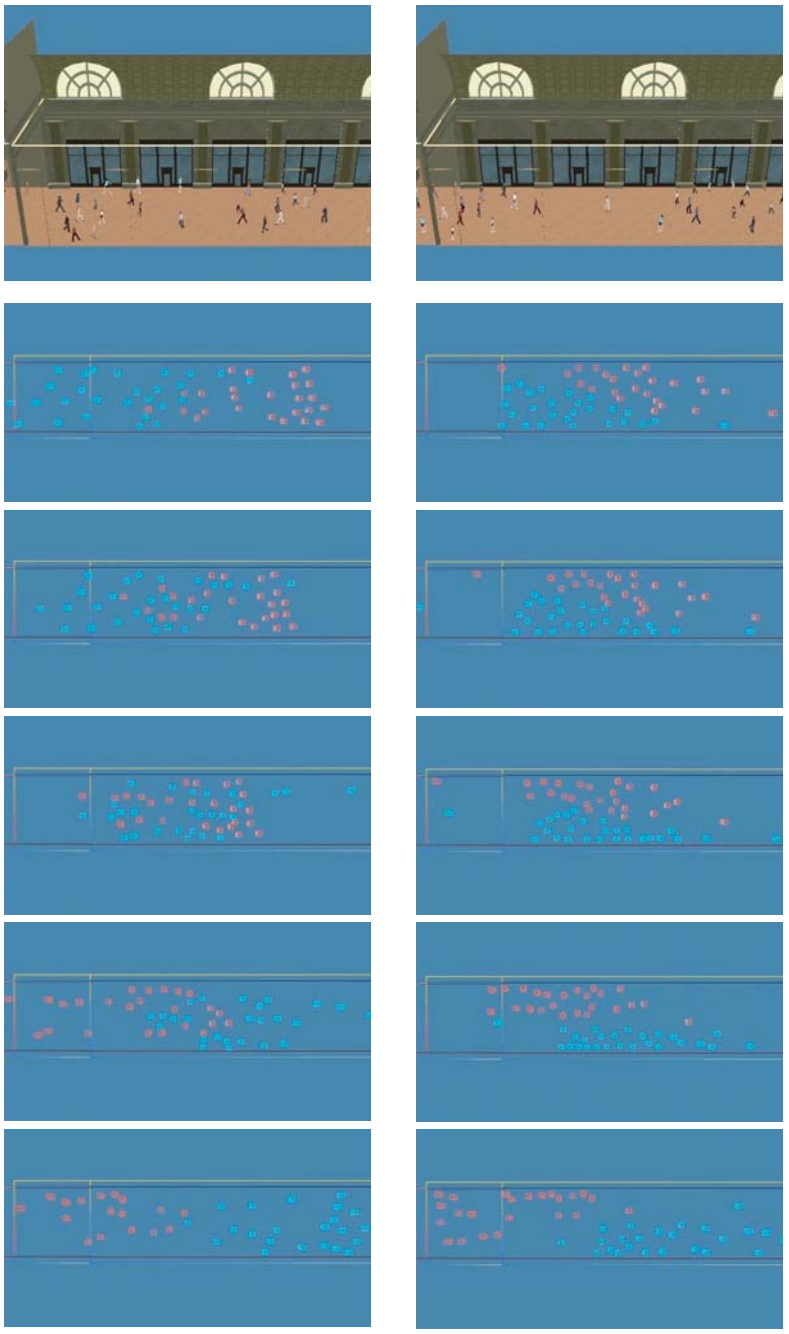


Figure 4.12: Crowds interacting in the arcade. Images in the left column show emergent group patterns due to the use of a combination of purely reactive behaviors. Images in the right column show motivational behaviors suitable for corridors and passageways.

4.3.3 Crowds Interacting in the Arcade

The self-organizing patterns described in Section 4.2 as well as the passageway navigation behavior covered in Section 4.3.2 are able to improve pedestrian traffic flow. Next, two simulations will illustrate the similarity and differences between them (see Figure 4.12).

In the first simulation (left column of Figure 4.12), two crowds of pedestrians heading in opposite directions along the arcade encounter each other. The pedestrians execute purely reactive behaviors described in Section 4.2 in order to avoid bumping into one another. Even with such simple behaviors, the virtual humans show interesting emergent group patterns. Though they are not explicitly programmed to do so, they spontaneously form interleaved lanes of pedestrians moving in opposing directions, as this tends to maximize the pace of everyone's progress.

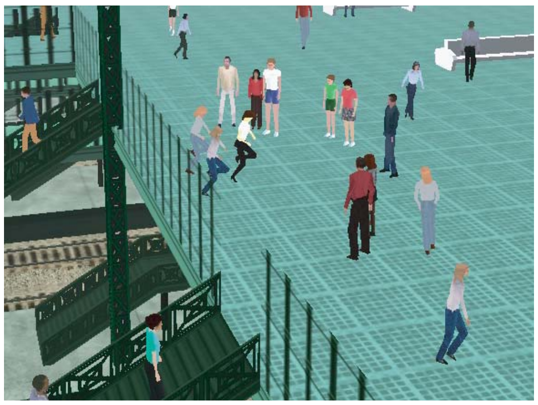
The scenario in the second simulation (right column of Figure 4.12) is similar to the first example, but this time the pedestrians perceive the arcade as a corridor that can conveniently accommodate only two major opposing traffic flows. Once oncoming people are observed, a pedestrian will search for safe direction in a clockwise fashion as described in the passageway navigation behavior (Section 4.3.2). As a result, the crowds squeeze past each other to their respective right. As soon as the oncoming crowd is cleared, the pedestrians tend to spread out again in order to maximize their view and pace.

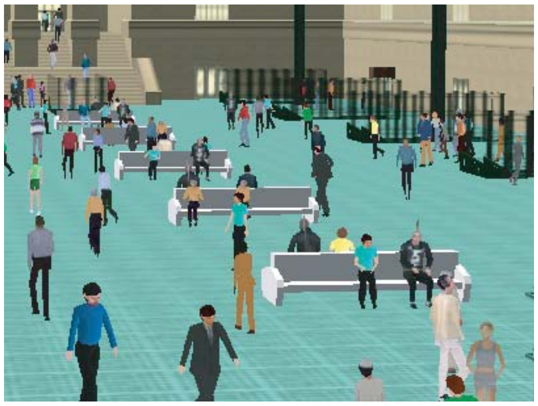
4.4 Other Interesting Behaviors

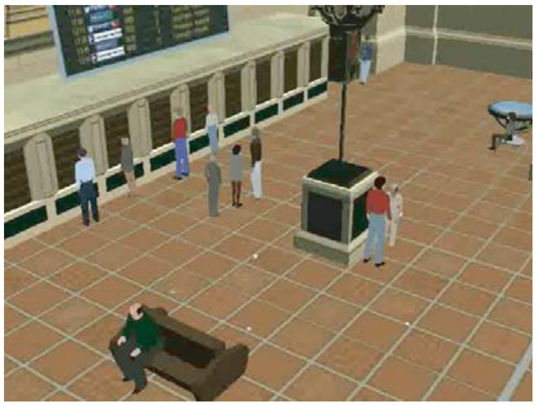
The previously described behaviors comprise an essential portion of the pedestrian's behavioral repertoire. To make our pedestrians more interesting, however, we have augmented their repertoire with a set of non-navigational behavior routines including, among others, the following:

- Stand and wait impatiently
- Select an unoccupied seat and sit down
- Approach a performance and watch
- Meet with friends and chat
- Queue at a vending machine and make a purchase
- Queue at ticketing areas and purchase a ticket

In this last behavior, for example, a pedestrian joins the queue and stands behind its precursor pedestrian until it comes to the head of the queue. Then, it will approach the first ticket counter associated with this queue that becomes available. This routine and some of the other aforementioned routines belong to a class of behaviors often observed in pedestrians. Such behaviors involve competition among pedestrians for available resources. In Appendix B, we will detail the description of these behavior routines to show how our pedestrians resolve potential conflicts like real humans. Figure 4.13 illustrates some of the aforementioned behaviors in our virtual pedestrians.


(a)


(b)


(c)

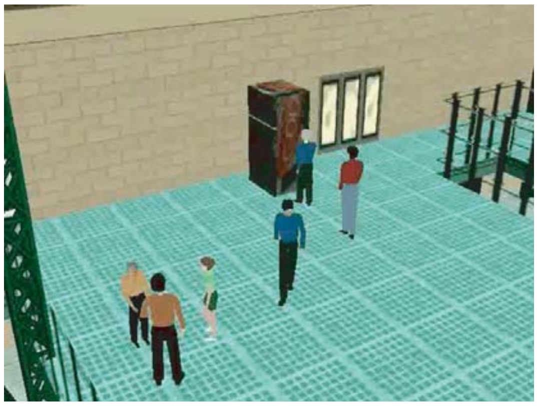

(d)

Figure 4.13: Examples of non-navigational behaviors. (a) Approaching a dance performance and watching. (b) Many pedestrians have selected unoccupied seats and are seated on the benches while others are moving about. (c) Queueing at ticketing areas and purchasing tickets. (d) Queuing at a vending machine and making a purchase. Note that the three pedestrians at the left are meeting and chatting.

Note that the non-navigational behaviors listed above depend on the basic reactive behaviors and navigational behaviors to enable the pedestrian to reach its targets in a collision-free manner. In addition, most of them also need the help from specialized environment objects (see Section 3.2.2, Section 6.4, and Appendix B), which provide quick and accurate answers to higher level perceptual queries, such as "Who is at the end of the line?" or "Is the purchase window available?", etc. These answers, which can be thought of as abstract interpretations of the situation, are crucial to the accomplishment of these behaviors and are well beyond the capability of perception maps.

4.5 Action Selection

Behaviors are responses to both external situations and internal mental states. Through sensory processes, as presented in Chapter 3.2.2, a pedestrian can acquire information about its surrounding environment as well as its evolving internal states. Given this information, an action selection mechanism controls the behavioral hierarchy at a high level. When a certain combination of a mental state variable value and proper external situation is perceived, the action selection mechanism will initiate an appropriate behavior to attempt to fulfill the need indicated by the mental state. For example, in a pedestrian whose thirst variable exceeds a predetermined threshold, behaviors will be triggered to locate a vending machine. Once a vending machine is found, the pedestrian will approach it, queuing up if necessary, to obtain a drink. If a tired pedestrian sees an empty seat, he will probably go and sit down to take a rest. In case more than one need awaits fulfillment, the most important need ranked by the action selection mechanism receives the highest priority. Hence, if a person is in a hurry to catch a train and passes by an attractive street show, he will likely ignore his interest in the performance and walk directly to the track. Finally, once a need is fulfilled, the value of the associated internal state variable begins to change back asymptotically to its nominal value.

In the virtual Penn Station environment, we classify pedestrians as commuters, tourists, law enforcement officers, performers, etc. Each pedestrian type has an associated action selection mechanism with behavior-triggering combinations of mental state thresholds and situation patterns set accordingly. This allows pedestrians to behave in a rational and meaningful way—for instance, law enforcement officers on guard will never attempt to buy a train ticket and commuters will never act like performers. As a representative example, Figure 4.14 illustrates the action selection mechanism of a commuter.

4.6 Summary

To summarize, the three sets of behaviors described in this chapter reflect different levels of sophistication. The non-navigational behaviors described in Section 4.4 direct our pedestrians to fulfill goals and enable them to demonstrate appropriate high level pedestrian behaviors. These behaviors require navigational and motivational behaviors covered in Section 4.3, which enable pedestrians to move around and reach target locations in a rational and natural manner. However, when the pedestrian is confronted with an imminent threat of collision or other danger, the motivational behaviors will be preempted by (and must immediately relinquish

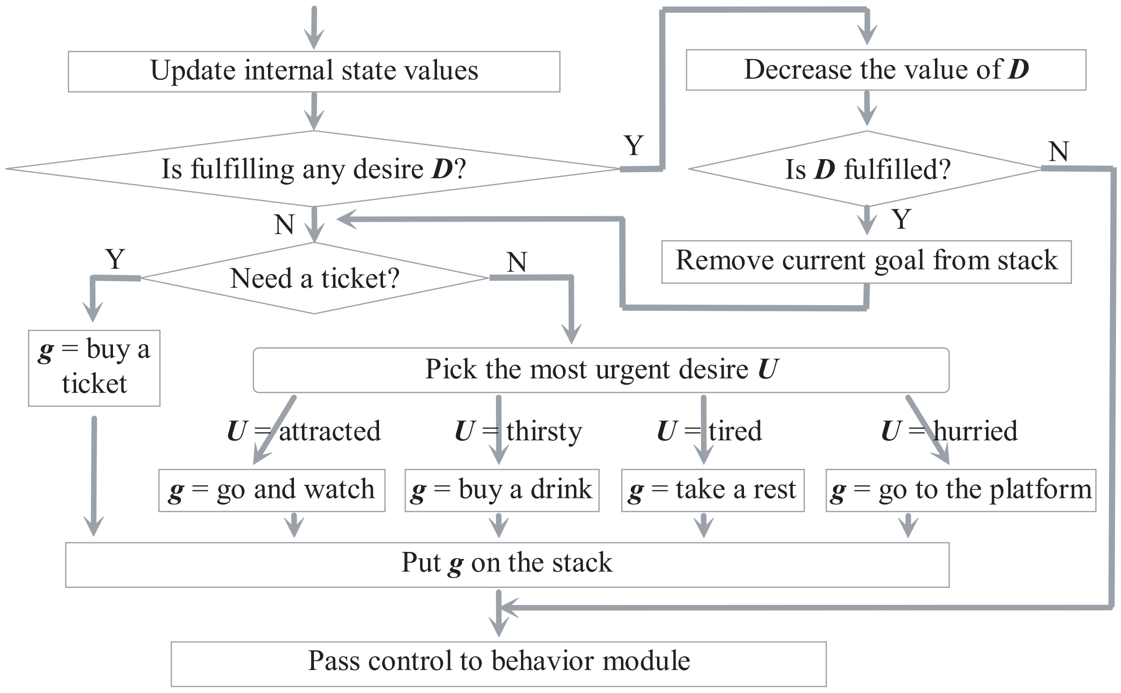


Figure 4.14: Action selection in a pedestrian commuter.

control to) the primitive reactive behaviors presented in Section 4.2. Finally, both the reactive and the motivational behaviors depend on the basic motor skills which drive the pedestrian's body to move appropriately.

In our behavioral hierarchy, an action selection mechanism at a high level is in charge of initiating and terminating behaviors for making progress, while primitive behavioral controllers at the low level protect pedestrians from danger. The behavioral hierarchy, in turn, supports higher-level cognitive control, which we will discuss in the next chapter.

Chapter 5

Cognitive Control

At the highest level of autonomous control, a cognitive model is responsible for creating and executing plans suitable for autonomous pedestrians. Whereas the behavioral substrate described in Chapter 4 is mostly *reactive*, the cognitive modeling layer makes our pedestrian a *deliberative* autonomous agent.

The realism of pedestrian animation depends on the execution of rational actions at different levels of abstraction and at different spatio-temporal scales. At a high level of abstraction and over a large spatio-temporal scale, a pedestrian must be able to make reasonable global navigation plans that can enable it to travel deliberatively (and with suitable perseverance) between widely separated regions of the environment, say from the end of the arcade through the waiting room, through the concourses, and down the stairs to a specific train platform. Such plans should exhibit desirable properties, such as being relatively direct and saving time when appropriate. During the actual navigation, however, the pedestrian must have the freedom to decide to what extent to follow the plan, depending on the realtime situation. Priority must be given to the local (in both space and time) situation in the dynamic environment in order to keep the pedestrian safe while still permitting progress towards a long-range goal. In addition, in an environment like Penn Station, a pedestrian should have the ability to handle not only a single goal, but a list of goals. With these insights, we follow three heuristics throughout the design of our pedestrian model, including the cognitive level:

- first, divide and conquer—always decompose complex tasks into simple ones;
- second, think globally and act locally—a plan is always needed at the global level but on the local level it will only serve as a rough guide; and
- finally, be flexible—always be prepared to change local sub-plans in response to the real-time situation.

In the subsequent sections, we will present the details of our cognitive model and explain how these heuristics are applied, starting with a brief description of the internal knowledge representation.

5.1 Knowledge of the Virtual World

To make plans, especially path plans, a pedestrian must have a world representation from its own egocentric point of view. In the interest of both time and space efficiency, we assume that the environment model is also the internal knowledge representation of the static portion of the virtual world for each pedestrian. Hence, except for wandering tourists, every pedestrian knows the layout of the environment and the position of static objects (such as walls, stairs, tracks, ticket booth, etc.) as well as the pedestrian's current position (which is referred as *localization* and has been an essential and longstanding problem in robotics [DeSouza and Kak 2002]). On the other hand, each pedestrian keeps, and updates at every simulation step, a representation of the dynamic aspect of the current world in the form of a list of perceived objects, including nearby pedestrians, and current events (such as "the ticket window is available", "that seat is taken", etc.).

This approach allows us to keep only one copy (or a small number of copies, in the case of multiprocessing) of the huge representation of the static world and still be able to support diverse cognitive and behavioral control for different pedestrians. Although this is an over-simplification, it is reasonable for modeling pedestrians who are usually familiar with the urban space around them, and it is a sensible choice for the purpose of simulating hundreds of pedestrians in large scale environments in real time.

5.2 Planning

Planning is one of the most essential parts of a cognitive model. For pedestrians, in particular, path planning is probably the most important planning task. Path planning directs a pedestrian to proceed through intermediate areas and reach ultimate destinations. In our pedestrian model, path planning is broken down into sub-tasks at different scales. Since the path planning algorithms are intimately dependent on the details of the environment model, which we will cover fully in the next chapter, we defer their details to Section 6.5 and Chapter 7.

Generally speaking, however, the breakdown of general path planning into sub-tasks allows the problem to be solved in a top-down manner. With unique specialty, each sub-task can attack its own part of the problem in a suitable and efficient way, giving its contribution to the final solution. In this way, the complete path planning process has the properties of both global extent and local accuracy as well as economic cost.

In addition to the path-planner, the cognitive level incorporates other planning routines (planners). Generally speaking, planners are like manuals that guide one through the steps needed to accomplish final goals. Each step may also require further instructions. For instance, to "meet a friend", a pedestrian must first reach the meeting point and then wait for the friend. Accomplishing the first step requires a path planner. Once the planner finishes its work, control is handed over to the behavioral level to carry out the plan until either it is done or new plan arises. Continuity is maintained by the memory model, which will be presented next.

5.3 Memory

A pedestrian can have a lot of different activities inside a big train station. He may have to meet with a friend before getting on a train. The route from station entrance to train track may be complicated and may require different strategies to navigate. On the way to the platform, he may want to get a drink or to stop and watch a dance by street artists, and so on. To keep certain things in mind while doing others, a pedestrian requires memory. Such a memory model enables a pedestrian to:

- store information—let the pedestrian memorize tasks (ultimate or intermediate);
- retrieve information—remind the pedestrian of pending tasks; and
- remove information—once accomplished, tasks shall be forgotten.

Unlike the knowledge of the virtual world which represents one's internal view of external things, the memory model stores internal results directly from thinking or planning, including, for instance, a list of pre-planned goals (first meet friend and then catch the train), a sequence of sub-tasks decomposed from a major task (to catch a train, one need to go through the arcade, cross the waiting room, pass one of the gates, cross the upper concourse, and go downstairs to the platform), interruption and resumption (after stopping by the vending machine and street performance, I still need to continue to the train platform), and so on. Therefore, such a memory model is highly dynamic and usually long-term, and can vary greatly in size.

To create the memory model, we again take a simple approach—use a stack as the memory. Such simplification does not sacrifice any functionality except that it restricts the operations to be on the topmost memory item only. The restriction certainly affects the capabilities of pedestrians, depriving them of being able to accomplish multiple tasks in flexible (non-deterministic) orders. Instead, a randomly-accessed list is among the more sophisticated mechanisms that can use to solve the problem. However, the observable difference of behaviors on pedestrians due to such a choice tends to be small, as probably pedestrians themselves do not make use of such flexibility. In addition, the stack data structure has simple

```
. if the goal is accomplished then remove the goal from memory and return.
. if the goal expires then remove the goal from memory and return.
. if multiple sub-tasks are required to achieve the goal then
    . make a plan containing these multiple sub-tasks base on current situation;
    . put the first sub-task on memory stack with a proper expiration time; and
    . return.
. determine the appropriate action needed to attain the goal.
```

Table 5.1: Typical skeleton of routines for processing memory items.

constant-time operations which permit easy and fast maintenance regardless of its size. This offers a big advantage toward our ultimate goal—creating a real-time simulation of hundreds or even thousands of pedestrians.

5.3.1 Memory Items

Before we describe how the memory stack works, let us first look at memory items. Each item on the memory stack stands for a goal and has a list of properties including a descriptor of the goal type (e.g., catch a train, have a rest, reach a specific target point), the level of complexity (ultimate or intermediate), goal parameters (e.g., platform 10, position and orientation of the seat, position of the target point), criteria of accomplishment(e.g., target must be within 0.5 meters, rest until recovered), expiration time (e.g., for 3 seconds, until finished).

5.3.2 Memory Item Processing

The top item on the stack is always the current goal. It will be processed by a cognitive or behavioral routine designed specifically for its type, typically in the way given in Table 5.1. As the table shows, complex goals will be decomposed into multiple simple ones. The reason why only the first sub-task gets pushed onto the stack is because the task decomposition is usually done according to the real-time situation (reflected, for instance, in the choice of goal parameters in sub-tasks). At the time, when the first sub-task is accomplished the remaining sub-tasks of the original decomposition may no longer be optimal as lots of things may have changed. Pushing all the sub-tasks at once will pin down the pedestrian to a static plan, which is not what real humans do. But if we just keep the first sub-task, when it is finished and gets removed from the memory stack, the original goal becomes the top one and gets processed again and a new decomposition can be determined then. To make it possible for pedestrians to be persistent to their previous choice, the original goal (the complex one) on the top of the memory stack will get updated after decomposition but before the pushing of the first sub-task. The update will store concise information of this decomposition into the then-top memory item (which stands for the original goal) so that later when it gets exposed again it can remind the pedestrian of the previous choice, giving him the chance of sticking to it. Of course, the decision of whether or not to stick to it is up to the cognitive or behavioral routine that will process the goal.

Sometimes, even the time needed to finish the first sub-task is long. In such cases, the expiration time attached to the memory item of the first sub-task can force the sub-task to expire after a certain amount of time, exposing the original goal again, and thus guarantee frequent plan update. The expiration time also forces replanning when a task resumes after an interruption by another one. For example, consider that a pedestrian feels thirsty on the way to the platform and detours to reach a vending machine. When he gets the drink, he needs to continue the trip to the platform. As the "get a drink" task is an interruption, there may be sub-goals of "get to the platform" underneath it on the memory stack reflecting an original plan of "get to the platform". Due to the expiration time of the sub-goals, they are likely to be invalid by then, which effectively forces a replanning of the "get to the platform" task.

5.4 Putting Things Together

Having described all the different components, let us put them together to construct an autonomous pedestrian. Figure 5.1 shows the layered relationship of the various components within the pedestrian model together with the world model. There are seven major components in the figure coded in different colors—the world model (Chapter 6), sensing (Chapter 3.2.2), internal states (Chapter 3.2.2 & 4), cognition (this chapter), behavior (Chapter 4), motor control interface (Chapter 3.2.1), and DI-Guy (Chapter 3.2.1). Components within some of the parts are also illustrated. Between the various components, there are arrows indicating data exchange during simulation. Next, we will use the figure to explain what happens at every *human simulation step*, which comprises several *simulation steps* or *frames*.

At the beginning of each *human simulation step*, a pedestrian H gets the top memory item as the current goal g (arrows **2** and **7**). If g is a complex goal (such as "meet a friend"), it will be decomposed by a planner routine into sub-tasks (e.g., "reach the meeting point" and "wait") and the first sub-task will become the

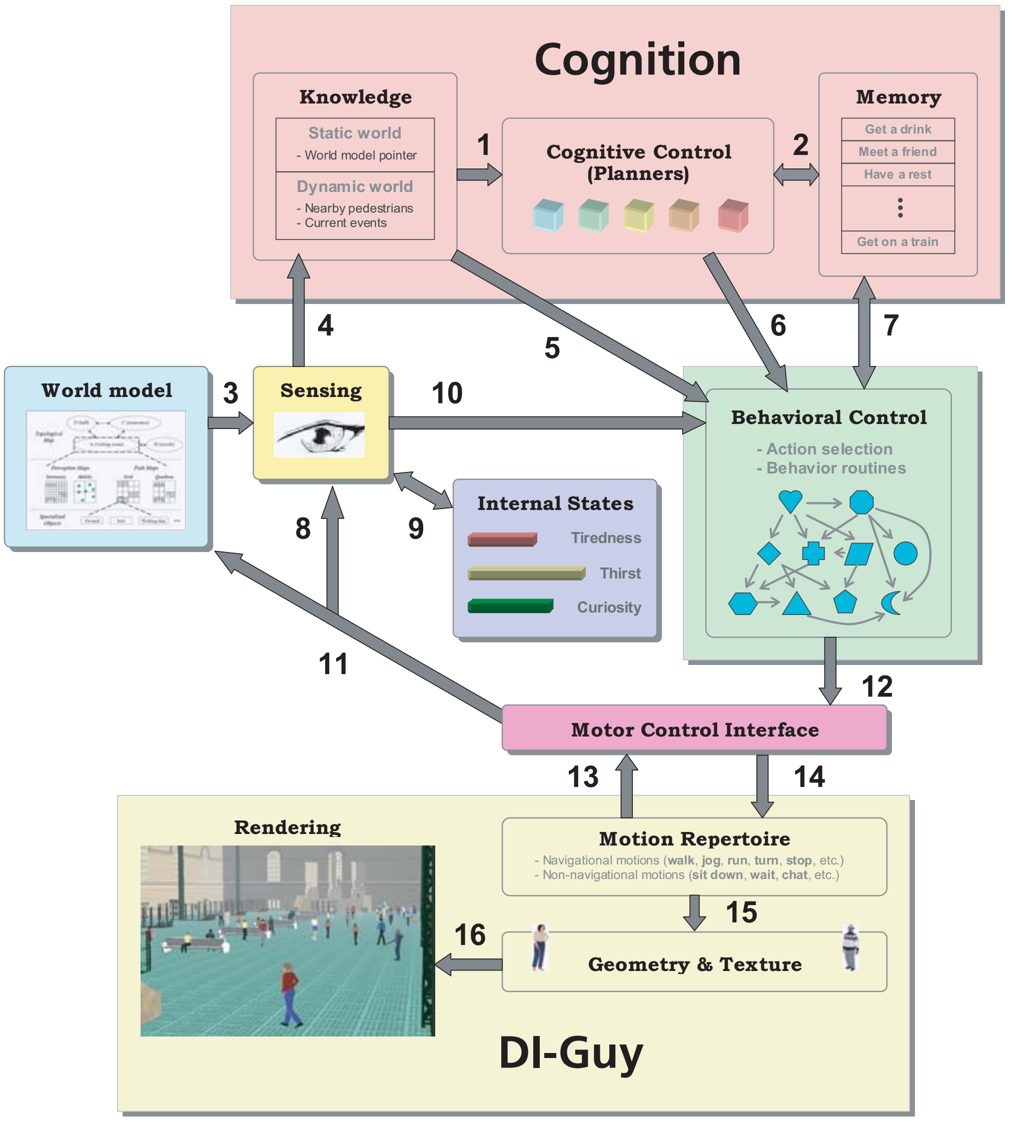


Figure 5.1: The complete autonomous pedestrian model.

current task and will be pushed onto memory stack (arrow **2**). Proper knowledge (usually of the static world) will be used (arrow **1**) during planning when needed (say, in planning a path). If the task is simple enough (arrows **6** and **7**), the action selection mechanism will choose a proper behavioral routine to handle it. The necessary information of the realtime external situation will be obtained directly or indirectly from sensing processes and internal knowledge representation (arrows **3**, **4**, **5** and **10**). Motor control commands (e.g., go straight forward with speed $1.2m/s$ in the next step, turn 15 degrees left with walk speed $0.5m/s$ in the next step, stand still, sit down, look at watch, look left, etc.) issued at the behavioral level are sent down (arrow **12**) to the motor control interface where they will first be verified and corrected in accordance with the motion abilities and kinetic limits of the pedestrian. Then they are translated into actual motions and/or transitions available in the motion repertoire. These motions and/or transitions will be picked (arrow **14**) from the repertoire to configure the pedestrian's postures (arrow **15**) for the next several *frames* (since each *human simulation step* comprises several *simulation steps* or *frames*). The textured geometric human model will be used to render the scene in each *frame* (arrow **16**). Feedback from the motion level (such as change of position and orientation, current speed, already sitting down or not, etc.) are obtained (arrow **13**) and relayed to the world model (arrow **11**) and the sensing component (arrow **8**) in order to perform a proper update. This update leads to a slightly different perceived world (arrow **3**, **4**, **5**, and **10**) by the pedestrian in the next *human simulation step.* The update also causes the internal state variables to change (arrow **9**), which may in turn trigger (arrow **10**) behavioral controllers to initiate or terminate certain behaviors in order to fulfill desires. Finally, after several *simulation steps*, a new *human simulation step* will be processed.

Note from the figure that the motor control interface effectively hides the *DI-Guy* software from our higher level models, which makes it easy, in principle, to replace it with other human animation packages.

5.5 Summary

In this chapter, we have presented the cognitive model of our virtual pedestrians, which completes the description of our autonomous pedestrian model except for the path-planning algorithms, which will be detailed in Chapter 7. Before we can explain the path planning aspects of the cognitive model, we must first present the full details of the environment model in the next chapter.

Chapter 6

Environmental Modeling

As outlined in Section 3.1, we represent the virtual environment by a hierarchical collection of data structures, including a topological map, two types of maps for perception, two types of maps for path planning and a set of specialized environment objects (see Figure 3.2 on page 35). With each of these data structures specialized to a different purpose, the combination is able to support accurate and efficient environmental information storage and retrieval.

6.1 Topological Map

At the highest level of abstraction, a graph serves to represent the topological relations between different parts of a virtual world. In this graph, nodes correspond to environmental regions and edges between nodes represent accessibility between regions.

A region is a bounded volume in 3D-space (such as a room, a corridor, a flight of stairs or even an entire floor) together with all the objects inside that volume (for example, ground, walls, ticket booths, benches, vending machines, etc.). We assume that the walkable surface in a region may be mapped onto a horizontal plane without loss of essential geometric information, such as the distance between two locations. Consequently, a 3D-space may be adequately represented by several planar maps, thereby enhancing the simplicity and efficiency of environmental queries, as will be described momentarily.

Another type of connectivity information stored at each node in the graph is *path-to-via* information. Suppose that $L(A, T)$ is the length in the number of edges of the shortest path from a region A to a different target region T, and $P(A, T)$ is the set of paths from A to T of length $L(A, T)$ and $L(A, T) + 1$. Then $V(A, T)$, the *path-to-via* of A associated with T, is a set of pairs defined as follows:

$$V(A,T) = \{\ (B, C_B) \ \ | \ \ \begin{array}{ll} B \text{ is a region} & \& \\ \exists p \in P(A,T) & \& \\ C_B = \text{length of } p & \& \\ B \text{ is next to } A \text{ on } p & \}. \end{array} \tag{6.1}$$

As the name suggests, if (B, C_B) is in $V(A, T)$, then a path of length C_B from A to T via B exists. In other words, $V(A, T)$ answers the question "To which region shall I go, and what cost shall I expect if I am currently in A and want to reach T"?

Given a graph, the *path-to-via* information is computed offline in advance using the incremental algorithm shown in Table 6.1. Note that after *Step 3* in the algorithm, only those entries are stored whose cost is *minimal* or ($minimal + 1$). In this way we can avoid paths with cycles. To understand this, consider $V(A, C)$

```
   Given $G(N, E)$, a graph with $N$ nodes and $E$ edges:

1. Initialization:
   for each node $A$
     for each target node $T$
       if $(A == T)$
         then $V(A,T) \leftarrow \{(A,0)\}$
         else $V(A,T) \leftarrow \{\}$

2. Collect information associated with paths of length $L$ based
    on the information associated with paths of length $L-1$:
   for length $L = 1$ to $N-1$
     for each node $A$
       for each target node $T$
         for every neighbor node $B$ of $A$ and any node $X$ in $G$
           if $(X, L-1) \in V(B,T)$
             then add $(B, L)$ to $V(A,T)$

3. Keep only low cost entries:
   for each node $A$
     for each target node $T$ and any node $Y$ in $G$
       let $C_{min}$ be the minimal cost in $V(A,T)$
       for each entry $E(Y,C)$ in $V(A,T)$
         if $(C > C_{min} + 1)$
           then remove $E$ from $V(A,T)$
```

Table 6.1: Algorithm for computing *path-to-via* information.

for the graph in Figure 3.2. C is a direct neighbor of A; so $(C, 1)$ is clearly an entry of $V(A, C)$. $(B, 3)$ is another entry as A-B-A-C is also a possible path from A to C. Obviously, A-B-A-C is not desirable as it contains a cycle. Such paths will automatically be removed from the *path-to-via* set after *Step 3*.

Linked within each node of the graph, are perception maps and path maps together with a list of objects inside that region. Next we will describe them in turn.

6.2 Perception Maps

Mobile objects and stationary objects are stored in two separate perception maps, which form a composite grid map. Hence, objects that never need updating persist after the initialization step and more freedom is afforded to the mobile object (usually virtual pedestrian) update process during simulation steps. Table 6.2 summarizes their similarities and differences, and the next two subsections present the details.

6.2.1 Stationary Objects

Our definition of a region assumes that we can effectively map its 3D space onto a horizontal plane. By overlaying a uniform grid on that plane, we make each cell correspond to a small area of the region and store in that cell identifiers of all the objects that occupy that small area. Thus, the grid defines a rasterization of the region. This rasterized "floor plan" simplifies visual sensing. The sensing query shoots out a fan of line segments whose length reflects the desired perceptual range

Type	Cell Size	Update Cost for the Entire World	Query Cost per Pedestrian
stationary	small ($\sim 10^{-1}m$)	0	constant, given the sensing range and acuity
mobile	large ($\sim 10^{1}m$)	linear in the number of pedestrians	constant, given the sensing fan and max number of sensed pedestrians

Table 6.2: Comparison of perception maps.

and whose density reflects the desired perceptual acuity (cf. [Tu and Terzopoulos 1994; Maes et al. 1995]). Each segment is rasterized onto the grid map (see the left and center panels of Figure 6.1). Grid cells along each line are interrogated for their associated object information. This perceptual query takes time that grows linearly with the number of line segments times the number of cells on each line segment. Most importantly, however, it does not depend on the number of objects in the virtual environment. Without the help of grid maps, the necessary line-object intersection tests would be time consuming given a large, complex virtual environment populated by numerous pedestrians. For high sensing accuracy, small sized-cells are used. In our simulations, the typical cell size of grid maps for stationary object perception is $0.2 \sim 0.3$ meters.

6.2.2 Mobile Objects

Similarly, a 2D grid map is used for sensing mobile objects (typically other pedestrians). In this map, each cell stores and also updates a list of identifiers of all the pedestrians currently within its area. To update the map, for each pedestrian H (with C_{old} and C_{new} denoting the cells in which H was and is, respectively):

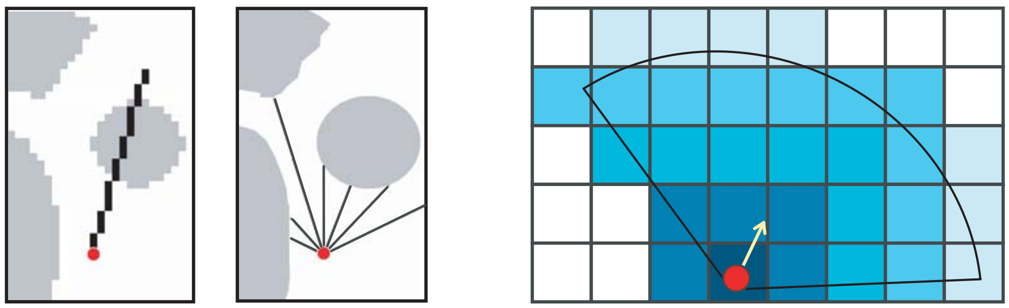


Figure 6.1: Perception maps and visual sensing. Left: Sensing stationary objects by examining map entries along a rasterized eye ray. Center: Perceive situation around by shooting out a fan of such eye rays (rasterization not shown). Right: Sensing mobile objects by examining (color-coded) tiers of the sensing fan.

1. If ($C_{old} == C_{new}$) then do nothing.

2. Otherwise, remove H from C_{old} and add him to C_{new}.

A hash map is used to store the list of pedestrians within each cell. As the update for each pedestrian takes negligible constant time, the update time cost for the entire map is linear in the total number of pedestrians, with a small coefficient.

The main purpose of this perception map is to enable the efficient perceptual query by a given pedestrian of nearby pedestrians that are within its sensing range. The sensing range here is defined by a fan as illustrated in the right part of Figure 6.1. In the mobile object perception map, the set of cells wholly or partly within the fan are divided into subsets, called "tiers", based on their distance to the pedestrian. Closer tiers will be examined earlier. Once a maximum number (currently set to 16) of nearby pedestrians are perceived, the sensing is terminated. This is intuitively inspired by the fact that usually people can simultaneously pay attention only to a limited number of other people, especially those in close proximity. Once the set of nearby pedestrians is sensed, further information can be obtained by referring to finer maps, by estimation, or simply by querying a particular pedestrian of interest. Given the sensing fan and the upper bound on the number of sensed pedestrians, perception is a constant-time operation.

6.3 Path Maps

Goal-directed navigation is one of the most important abilities of a pedestrian, and path planning enables a pedestrian to navigate a complex environment in a sensible manner. To facilitate fast and accurate online path planning, we use two types of maps with different data structures–grid maps and quadtree maps, which will be briefly presented next. The path planning algorithms will be detailed in a subsequent chapter, after we have finished describing the environment representation hierarchy.

6.3.1 Grid Path Map

Grid maps, which are useful in visual sensing, are also useful for path planning. We can always find a shortest path on a grid map, if one exists, using the well-known A^* graph search algorithm [Rabin 2000; Stout 2000].

In our system, grid path maps are used whenever a detailed path is needed. Suppose D is the direct distance between pedestrian H and his target T. Then a detailed path is needed for H if D is smaller than a user-defined constant D_{max} and there are obstacles between H and T. This occurs, for instance, when one wants to move from behind a chair to its front and sit on it. Clearly, the accuracy of the path in this instance depends on the size of the cells in the grid path maps. A small cell size results in a large search space and, likely, low performance. However, detailed paths are usually not needed unless the target is close to the starting point. Therefore, chances are that paths are found quickly, after the search covers only a small portion of the entire search space. Roughly speaking, in most cases the space that must be searched is bounded by $4(D_{max}/c)^2$, where c is the cell size. The typical values for these constants in our current system are $1 \leq D_{max} \leq 10$ meters and $10^{-1} \leq c \leq 1$ meters.

In addition, grid path maps are used when path search fails on a quadtree path map. This will be explained later in the section on path planning (Chapter 7).

When creating grid maps, special care must be taken to facilitate efficient updates and queries. Polygonal bounding boxes of obstacle objects represented on grid maps are enlarged by half of the size of a pedestrian's bounding circle. If the center of a pedestrian never enters this "buffer" area, collisions will be avoided. This enables us to simplify the representation of a virtual pedestrian to a single point, which makes most queries simpler and more efficient.

6.3.2 Quadtree Path Map

Every region has a quadtree map, which supports fast online path planning [Botea et al. 2004]. Each quadtree map comprises

1. a list of nodes N_i $(i = 0, 1, 2, \cdots, m)$, which together cover the whole area of the region (see Step (3) in Figure 6.2);
2. C, the number of levels; i.e., the number of different node cell sizes appearing in the map (which is 3 for the quadtree map in Figure 6.2); and

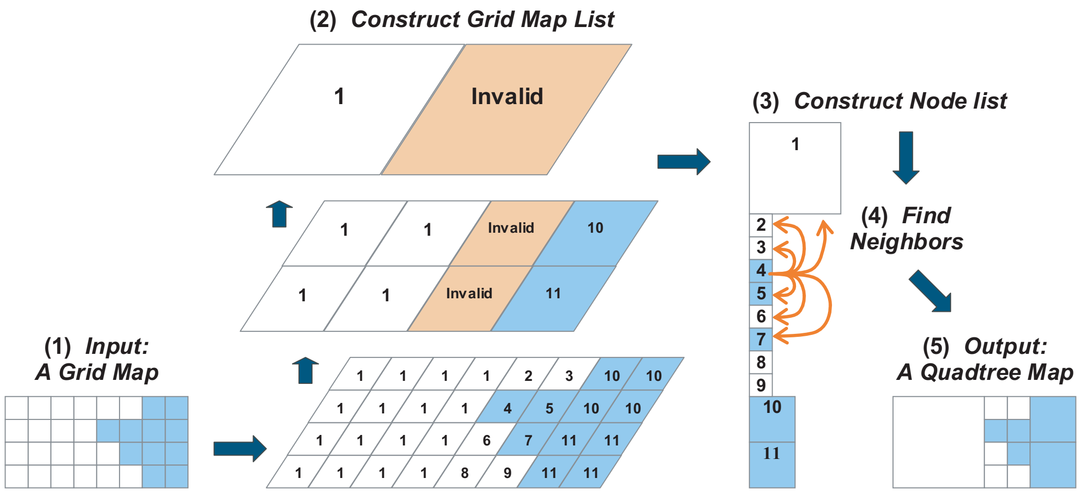


Figure 6.2: Construct a quadtree map.

3. a pointer to an associated grid map with small cell size (see Step (1) in Figure 6.2).

Each node N_i of the quadtree [Samet 1989] stores the following variables:

1. L_i, where $0 \leq L_i < C$, the level of the node in the quadtree (which also indicates the cell size of N_i);
2. the center position of the area covered by this node;
3. the occupancy type (ground, obstacle, etc.);
4. a list of pointers to neighboring nodes;
5. a congestion factor g_i, which is updated at every simulation step and indicates the portion of the area covered by this node that is occupied by pedestrians; and
6. a distance variable, which indicates how far the area represented by the node is from a given start point, and will be used at the time of path-planning,

especially during back-tracking as a gradient reference to find the shortest way back to the start point.

As Figure 6.2 illustrates, given a grid map with small cells, the algorithm for constructing the quadtree map first builds the list of map levels containing nodes representing increasing cell sizes, where the cell size of an upper level node is twice as large as that of lower level nodes. Higher level nodes, which aggregate lower level nodes, are created so long as the associated lower level cells are of the same occupancy type, until a level is reached where no more cells can be aggregated. Quadtree maps typically contain a large number of lower level nodes (usually over 85% of all nodes) that cover only a small portion (usually under 20%) of the entire region. Such nodes significantly increase the search space for path planning. Thus, in the final stage of construction, these nodes are excluded from the set of nodes that will participate in online path planning. As the area that they cover is small, their exclusion does not cause significant accuracy loss. However, in occasions when path planning fails because of this exclusion, grid maps will be used to find the path, as will be described in Chapter 7.

6.4 Specialized Environment Objects

At the lowest level of our environmental hierarchy are environment objects. Cells of grid maps and quadtree maps maintain pointers to a set of objects that are partly or wholly within their covered area. Every object has a list of properties, such as name, type, geometry, color/texture, functionality, etc. Many of the environment objects are specialized to support quick perceptual queries. For instance, every ground object contains an altitude function which responds to ground height sensing queries. A bench object keeps track of how many people are sitting on it and where they sit. By querying nearby bench objects, weary virtual pedestrians are able to determine the available seat positions and decide where to sit without further reference to the perceptual maps. Other types of specialized objects include queues (where pedestrians wait in line), purchase points (where pedestrians can make a purchase), entrances/exits, etc. In short, these objects provide a higher level interpretation of the world that would be awkward to implement with perception maps alone, and this simplifies the situational analysis for pedestrians when they perform autonomous behaviors.

6.5 Processing Queries

The environment representation, together with the algorithms designed on it, efficiently provides accurate perceptual data as well as planning results in response to the various queries that may be issued by an autonomous pedestrian. Typical queries are explained next in order of increasing abstractness.

Sensing ground height To ensure that a virtual pedestrian's feet touch the ground in a natural manner, especially when climbing stairs or locomoting on uneven ground, the pedestrian must query the environment model in order to sense the local ground height so that the feet can be planted appropriately. Each grid map cell contains the height functions of sometimes a few (most often a single) ground objects. The greatest height at the desired foot location is returned in constant time.

Visual sensing As stated earlier in section 6.2 (see also Figure 6.1), our data structures dramatically increase the efficiency of the sensing processes when a pedestrian must perceive static obstacles and nearby pedestrians, which is a crucial component of obstacle avoidance. On the perception map for static objects, rasterized eye rays are used to detect static obstacles. On the perception map for dynamic objects, a constant number of neighbor cells are examined to identify nearby pedestrians. Both of the algorithms are localized and do not depend on the size of the world, the number of objects or pedestrians, or anything else that increases with world size.

Locating an object Given a location identifier (say, "Track 9"), a search at the object level can find the virtual object. This is accomplished in constant time using a hash map with location names as keys. As the virtual object has an upward reference to its region, it can be located quickly (say, "under the lower concourse") by referring to the node in the topological graph, as can nearby objects in that region (say, "Platform 9" and "Platform 10") by referring to the perception maps linked within the node.

Acquiring high level interpretation of specific situation Abstract interpretation of the environment is indispensable for creating high level behaviors such as "to get a ticket" to be described in Chapter 4. Take it as an example. In order to get a ticket, a pedestrian needs to figure out (1) the length of wait line at each booth to pick a fast one, (2) the last person on the wait line in order to go and wait on the line, (3) when he becomes the first person on the line, (4) whether a window is available for him to make the purchase. Such queries are all answered by specialized objects, in this case a *queue* object and several *purchase point* objects associated with the ticket booth, which keep track of the evolving situation.

Planning paths between regions Here, the *path-to-via* information is useful in identifying intermediate regions that lead to the target region. Any intermediate region can be picked as the next region and, by applying one of the path-searching schemes described in Chapter 7, a path can be planned from the current location to the boundary between the current region and that next region. The process is repeated in the next region, and so on, until it can take place in the target region to terminate at the target location. Although the extent of the path is global, the processing is local.

6.6 Summary

In this chapter, we presented a large-scale environment model, which includes a sophisticated set of hierarchical data structures that support fast and accurate perceptual queries and path search algorithms. In the next chapter, we will describe the algorithms employed for online path-planning on path maps in the environment model.

Chapter 7

Path Planning

Having presented the details of our environment model, we can now fill in the missing details of the pedestrian cognition model which we covered in Chapter 5, especially as they relate to path planning.

7.1 Path Planning Sub-Tasks

As we mentioned in Chapter 5, path planning in our pedestrian model is broken down into sub-tasks at different scales:

- ***Task A:*** Given two regions R_S and R_T, find an R_S's neighboring region R_N through which a pedestrian can reach R_T. This is done by first exploiting the topological map at the top level of the environment model. A set of optimal neighboring regions leading to R_T can be determined simply by a table-lookup operation (using the precomputed *path-to-via* information in the topological map (see Section 6.1)). From them, the best one can then be picked through passageway selection behavior or the like at the behavioral level, based on both the proximity and the pedestrian density (see Section 4.3.2).

- ***Task B:*** Find a rough path connecting the start point S and the target T, which are within the same region R but far away from each other. Here we first employ one of the search algorithms presented in Section 7.4 to search for a path on the quadtree path map of R. If the attempt fails for the reason explained later in this section, we will plan a path on one of the grid path maps of R. The path found, whether on quadtree map or grid map, is usually jagged or full of spikes. It will be taken as a rough guide by the pedestrian during visually-guided navigation, as covered in Section 4.3.1.

- ***Task C:*** In the cases when S and T are close to each other but have obstacles in between, plan a detailed path from S to T. Now the finest grid path map of this region is used here for the path search in order to increase accuracy. The path will then be slightly smoothed, and the pedestrian will follow the processed path strictly when approaching the target as we describe in Section 4.3.1.

With these sub-task solvers at hand, and given a pedestrian's current location S in region R_S and a target location T in region R_T, the general path planning can be achieved as follows:

1. Use ***Task A*** to pick the right neighboring region R_N of R_S.

2. Determine B_{SN}, the boundary area or portal area between R_S and R_N.

3. Either go directly toward B_{SN} or use ***Task B*** to plan a path and follow the path to B_{SN}.

4. Repeat Steps 1, 2, and 3 in the new region until region R_T is reached.

5. Either go directly toward T or use ***Task B*** to approach T.

6. Either go directly toward T or use ***Task C*** to arrive at T.

Note that at the places where ***Task B*** or ***Task C*** is used, the current target location (whether ultimate or not) will first be checked to determine whether or not it is directly reachable. If it is, a pedestrian will simply move straight toward it without further ado.

Autonomous virtual pedestrians are capable of automatically planning paths around static and dynamic obstacles in the environment. When used for path planning, quadtree maps have the advantage of quick response due to their smaller search space, while grid path maps are good for detailed plans as they have higher resolution. Thus, quadtree maps are always used when detailed paths are not required. Sometimes, however, quadtree maps may fail to find a path even though one exists. This is because low level nodes on a quadtree map are ignored during path search in order to decrease the search space. Therefore, paths that must go through some low level nodes cannot be found on a quadtree map. To resolve this, we turn to grid path maps whenever search fails on a quadtree map. The standard A^* algorithm [Rabin 2000; Stout 2000], is used to find a path on grid maps. For efficiency reasons, a list of grid maps with decreasing cell sizes is kept for each region. Grid map path search starts at the coarsest level and progresses down the list so long as the search fails on coarser levels. Although grid maps with large cells are likely to merge separate objects due to aliasing, and thus cause the search to fail, the multiresolution search ascertains that, as long as the path exists, it will always be found eventually on maps with smaller cells.

In the remaining part of this section, we describe the path planning algorithm for quadtree path maps. To find a path on a quadtree map M from a given starting point S to a target T, the main phases of the algorithm are as follows:

1. insert T into M and expand it if necessary (see Section 7.5);

2. from S, try to find any node in the expanded target using one of the several available search schemes (see Sections 7.2, 7.3, and 7.4);

3. if the search reaches an expanded target node C, then (1) back-track through the visited nodes to construct a path P_{cs} from C to S and (2) back-track through expanded target nodes to build a path P_{ct} from C to a real target node in T (see Section 7.6);

4. link P_{cs} and P_{ct} together to construct an initial path P_{st} from S to T; and

5. post-process P_{st} to compute the final path P.

The details of the algorithm will be discussed in the next several sections.

7.2 Set Distance Variables

The search schemes mentioned in *Step 2* employ several variations of the A^* search algorithm. In the conventional A^* algorithm, the search procedure iteratively gets an unvisited (ground) node from a queue, visits it, marks it as visited, adds its

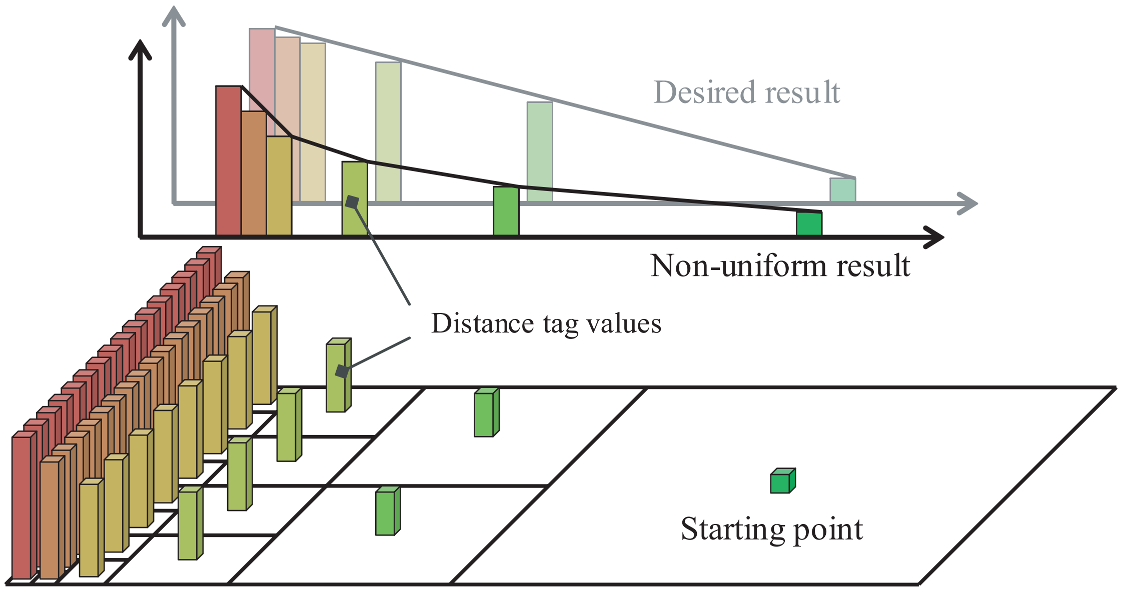


Figure 7.1: Non-uniform distance field caused by constant increment on a quadtree map.

neighbors to the queue, and repeats until the target is reached or the algorithm fails to reach the target. As the algorithm progresses, it updates a distance variable in each node which indicates the approximate distance of the node from the start point. After the search succeeds, the distance tags of all the visited nodes form a distance field, which back-tracking uses to find a shortest path along the distance gradient from the target point back to the start point.

When A^* algorithm is applied to plan paths on uniform grids, the increment of distance variables at each search step is always the same due to the constant cell size. The situation with a quadtree map is different, however, as the nodes of a quadtree map can represent different sized regions. Constant-size increment of distance variable at each step will produce a non-uniform distance field on quadtree maps (see Figure 7.1). In updating the distance variables, we therefore employ the

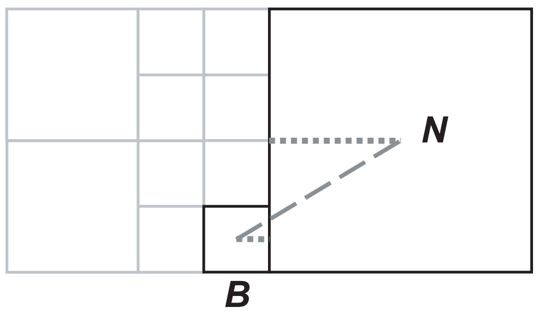


Figure 7.2: Distance approximation on quadtree map. Distance (dashed line) between centers of neighboring nodes is approximated by the sum of the half of the sizes of those two nodes (dotted lines).

actual distances $Dist_c(N, B)$ between the centers of adjacent nodes N and B. To make it efficient, the half sum of N's size and B's size is used to approximate the actual distance (see Figure 7.2). Notice that node sizes increase or decrease by a factor of 2 from level to level in the quadtree, so we have:

$$Dist_c(N, B) \approx 0.5 \times S_0 \times (2^{L_N} + 2^{L_B}) \tag{7.1}$$

where S_0 is the size of nodes at level 0 (those smallest nodes) and L_N and L_B are levels of the nodes N and B, respectively. Further simplification can be achieved by (1) omitting the constant part $0.5 \times S_0$ and (2) keeping 2^x in a precomputed list, so that Equation 7.1 becomes:

$$Dist_c(N, B) \sim Power_2(L_N) + Power_2(L_B) \tag{7.2}$$

where $Power_2 = \{1, 2, 4, 8, 16, 32, 64, 128, \cdots\}$ is a list of powers of 2. In this formula, the sum of two increments at level i is equal to one increment at level

$i+1$, which exactly reflects the fact that the cost of going through two consecutive nodes at level i is the same as that of going through one node at level $i+1$. However, note that the centers of regions associated with large ground nodes in the quadtree are generally further from obstacles than those of the small ground nodes. When planning paths, virtual pedestrians can keep further away from obstacles simply by favoring bigger nodes in their path searches. The search algorithm does this by appropriately weighting the internodal distances from level to level in the quadtree:

$$Dist_c(N, B) \sim w(L_N) \times Power_2(L_N) + w(L_B) \times Power_2(L_B) \qquad (7.3)$$

where $w(i)$ is a choice of the weight function. To be more specific, our current choice of weights effectively change the $Power_2$ series into a $Fibonacci$ series, which gives limited favor to bigger nodes:

$$Dist_c(N, B) \sim Fib(L_N) + Fib(L_B) \qquad (7.4)$$

where $Fib = \{1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \cdots\}$ is a $Fibonacci$ series. Clearly, it favors one node at level $i+1$ rather than the sum of two nodes at level i, but it treats the former the same as the sum of one at level i and one at level $i-1$. This way the path found by back-tracking, although might be longer than the shortest one, will not be too much longer. To be more precise, in worst case paths are $1/3$ longer than the shortest.

7.3 Congestion

The search algorithm also takes the congestion of a region into consideration during path planning. The congestion factor variables C_N and C_B of adjacent nodes N and B are incremented and decremented when a pedestrian transitions between the regions covered by these nodes. If the congestion factor C_i of node i exceeds a preset congestion threshold C_{thresh}, the internodal distances from that node are weighted with a weighting factor greater than unity. Thus, Equation 7.4 becomes:

$$Dist_c(N,B) \sim w_{cong}(C_N) \times Fib(L_N) + w_{cong}(C_B) \times Fib(L_B) \tag{7.5}$$

where $w_{cong}(x)$ is a choice of weight function and is currently defined as:

$$w_{cong}(x) = \begin{cases} 1 & \text{if } x < C_{thresh} \\ (\frac{1}{1-(x-C_{thresh})})^2 & \text{if } C_{thresh} \leq x < C_{thresh}+1 \\ infinity & \text{if } C_{thresh}+1 \leq x \end{cases} \tag{7.6}$$

According to the above formula, greater costs will be associated for more congested nodes in terms of their distance from a start node. This effectively pulls crowded nodes away from the starting point. As the backtracking process always trace back along the gradient, it will pick a low cost route and thus in a way automatically avoid crowded area.

In Figure 7.3, we compare the paths planned with and without congestion model. Part 1) shows a highly crowded sub-area (in gray circle) appearing in a simulation with 210 agents. In the picture, blue objects are obstacles, thin long green objects are walls and red dots are agents. In part 2), quadtree map

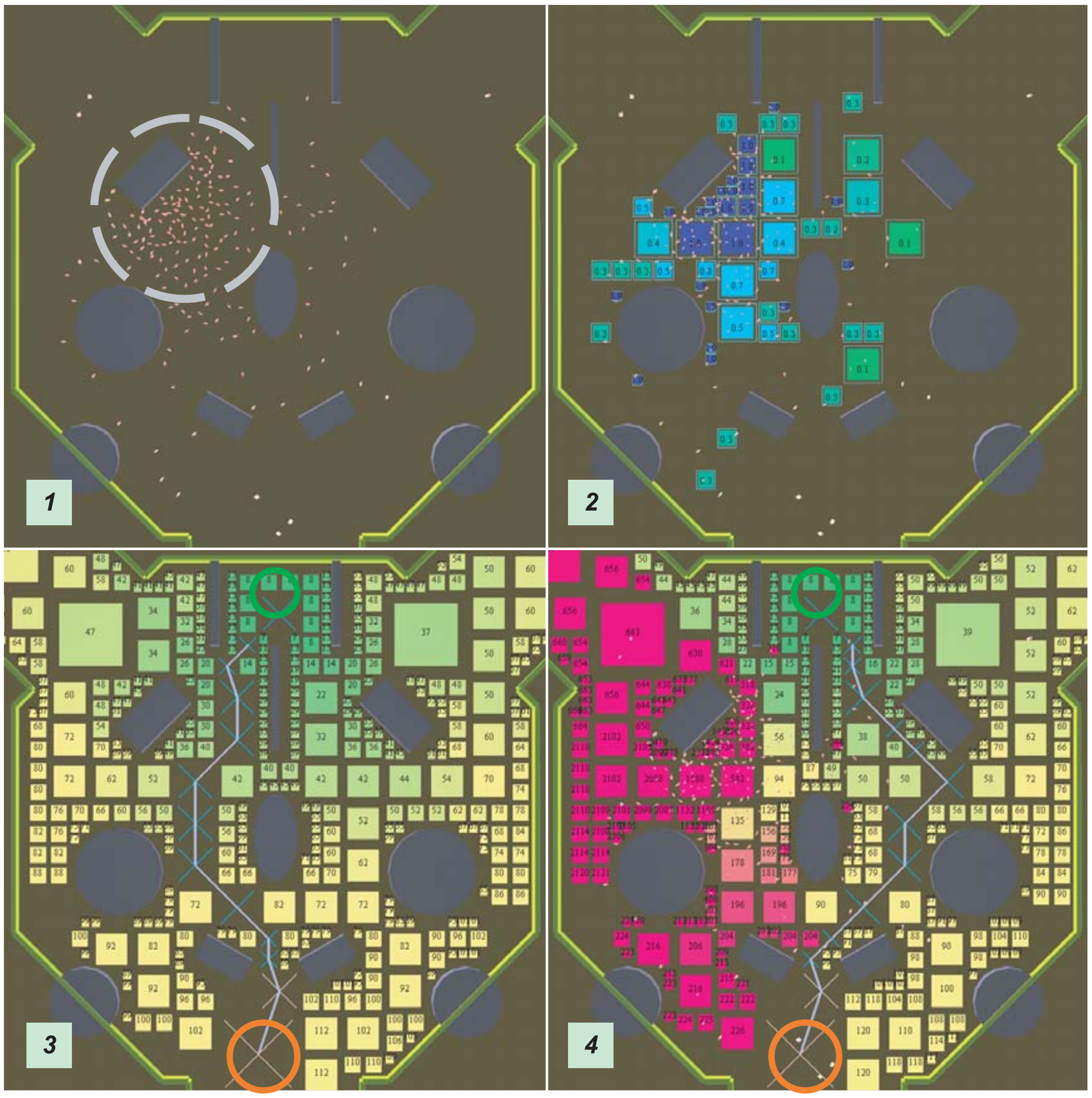


Figure 7.3: Congestion model on quadtree maps.

```
. Let $S$ = the start node. $EnQueue(S)$.
. Loop
  . $N = DeQueue()$
  . if ($N == NULL$) return "space exhausted"
  . for each neighbor $B$ of $N$, do
    . if ($B$ is a expanded target node) return "target reached"
    . if ($B$ is not a ground node) continue
    . if ($B$ has been visited) continue
    . $B_{dist\_val} = N_{dist\_val} + Dist_c(N, B)$
    . $EnQueue(B)$
. End loop.
```

Table 7.1: Algorithm outline shared by search schemes.

nodes whose congestion factor > 0.1 are shown in color, with more blueish squares indicating more congested area. Part 3) shows a path planned from green circle on the top to orange circle at the bottom **without** congestion model. And in part 4), another path is planned for the same pair of start and target points **with** congestion model. For both 3) and 4), color coded search space is also shown– visited nodes as colored squares with distance variable values printed in the center of the square. Note in 4), when search encounters highly crowded area, the distance variable values suddenly go up (indicated by the sudden color change to red). The red nodes act like a barrier, preventing back-tracking from going through them, and thereby give great favor to the new path.

7.4 Search Schemes

We have designed several variants of the A^* algorithm, each with its own emphasis. Some variants aim at high speed while others always attempt to find shortest paths.

Name	**Number of queues**	**Nodes are sorted in queues?**	**Queues are prioritized?**	**Path length**	**Search speed**
SortedQ	one	partly sorted by distance variable values	no	shortest	slowest
SingleQ	one	no	no	shorter	slower
MultiQ	multiple, one for nodes at each level	no	no	longer	faster
PmultiQ	multiple, one for nodes at each level	no	queues of nodes at higher level have higher priority	longest	fastest

Table 7.2: Comparing different search schemes (1): number and properties of queues.

The virtual pedestrians use all of the variants in order to increase the variety of their motions. Our A^* variants share the same algorithm outline as shown in Table 7.1 and differ in how they maintain the unvisited nodes in the queue(s) as shown in Table 7.2 and Table 7.3.

Variant **SingleQ** is the standard A^* algorithm for uniform grid maps. It does not find shortest paths in quadtrees because it expands quadtree nodes non-isotropically. **SortedQ** maintains a sorted queue, and thus it first visits the frontier node (frontier nodes are nodes in the boundary between visited and unvisited regions) which is closest to the start node. The expansion is more isotropic, but maintaining a sorted queue is relatively expensive. **SortedQ** tends to visit larger quantities of smaller nodes in the quadtree, since they are more plentiful; hence, the visited region tends to grow rather slowly. **MultiQ** maintains a separate queue for each level in the quadtree and it visits every level in turn; hence, more large

- SortedQ	
EnQueue	Insert node N into the queue so that the last K nodes in the queue are sorted in increasing order of distance variable values. K is a user-defined value.
DeQueue	Return a node removed from the head of the queue.
- SingleQ	
EnQueue	Append node N to the tail of the queue.
DeQueue	Return a node removed from the head of the queue.
- MultiQ	
EnQueue	Append node N to the tail of $queue(N.level)$
DeQueue	If last time *DeQueue* returned a node from $queue(i)$, then this time return a node removed from the head of the next queue – $queue(i+1)$.
- PmultiQ	
EnQueue	Append node N to the tail of $queue(N.level)$
DeQueue	For each *queue* from highest priority to lowest priority if *queue* is not empty then remove a node from the head of *queue* and return it.

Table 7.3: Comparing different search schemes (2): enqueue and dequeue operations.

nodes are visited sooner than in the **SingleQ** search, improving the search speed. **PmultiQ**, is a prioritized **MultiQ** scheme, which tries to visit the largest frontier node first and therefore exhibits the fastest growth speed among the four methods. If we consider the size of a node to be analogous to the size of a road, the **PmultiQ** scheme finds a path by searching along interstate highways first, then all state highways, then local roads, etc., until it finds the target. However, the path found will not necessarily be optimal. Thus, in the following order of the four schemes (**SortedQ, SingleQ, MultiQ, PmultiQ**), the length of the paths that they generate are increasingly less optimal, but the searches are increasingly more efficient.

In figure 7.4, we compare paths computed by the four search variants of the path planning algorithm. Part 1) is visualization of the quadtree map of the concourse's upper level in the Pennsylvania Train Station environment model. The white quads denote ground nodes and the blue ones denote obstacles. The green circle on the left is the start point and the orange circle on the right is the target. 2)-5) show results of the four path planning schemes 2) SortedQ, 3) SingleQ, 4) MultiQ, and 5) PmultiQ. The search space is color coded with the distance variable values increasing from green to orange. Note that, although the four paths are similar, the sizes of search space are different. (For clarity, obstacle quads are not shown in the lower images 2)-5).)

7.5 Target Expansion

To make the search even faster, every target is expanded on a quadtree map until it touches any node at a certain level L_t or above. The value of L_t is a trade-off between accuracy and efficiency and it is automatically determined during map construction. Nodes on and above this level shall cover a large portion of the entire region (currently we set the threshold to be 70%) and the number of them shall be significantly small (usually less than 30% of all nodes). Target expansion will likely shorten the time needed in the search step as nodes at levels lower than L_t probably do not have to be visited before the expanded target is reached (especially in the

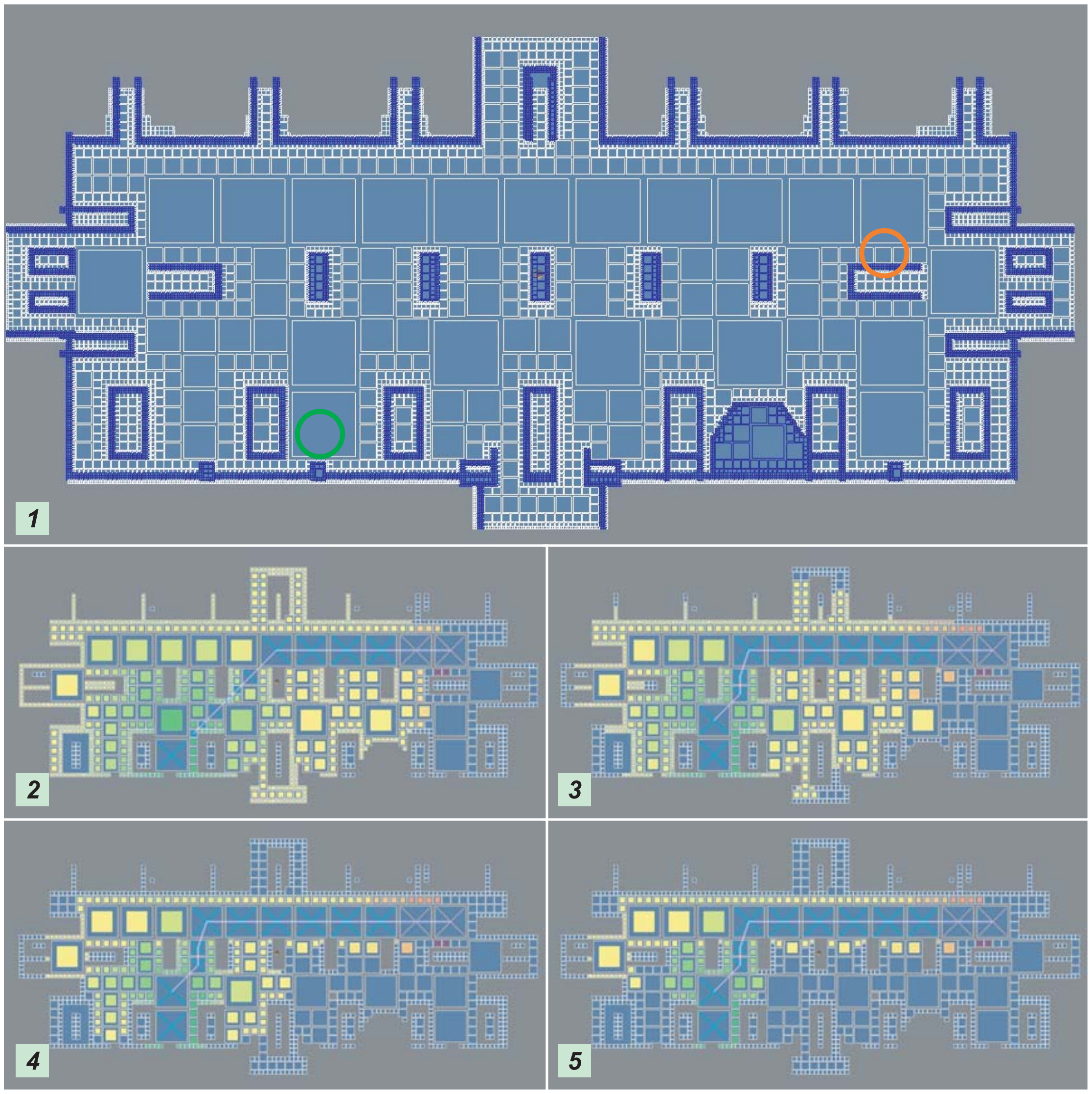


Figure 7.4: Comparison of path planning algorithms on quadtree maps.

Name	**Map Accuracy (smallest cell size in meters)**	**Total Search Time (in seconds)**	**Average Path Length (in meters)**	**Average Number of Nodes visited**
Grid	0.5	621.9	29	27078
Grid	1.0	141.1	31	6211
SortedQ	0.2	89.5	26	2817
SingleQ	0.2	62.3	28	2190
MultiQ	0.2	26.1	30	820
PmultiQ	0.2	19.5	32	535

Table 7.4: Comparison of path search on path maps. Results from a set of 50,000 searches in several different environments.

case of **PmultiQ**). In order for the expansion to be nearly isotropic, **SortedQ** search scheme is used to expand the target for it to reach a node on or above level L_t.

7.6 Path Construction

The search step successfully completes its task once it has found a node T_e that belongs to the expanded target. Using the aforementioned back-tracking method, the path planning algorithm constructs a path from the starting point to T_e within the searched area and a path from T_e to a real target node within the expanded target area. By linking these two paths together, we obtain a complete path to the target. Finally, the complete path will be refined for use by the virtual pedestrian, as discussed early when we describe pedestrian behaviors in Chapter 4.

Map Type	**% of Paths Planned**	**% of Time Used**
Quadtree	94%	8%
Grid	6%	2%

Table 7.5: Usage of path maps for non-detailed-path-planning. Results measured from a set of 20-minute-long simulations with various number of pedestrians (100 $\sim$ 1000) in Penn Station environment.

7.7 Performance Comparison

Table 7.4 compares the average performance of the two types of path maps and various search schemes of quadtree maps, as measured in our experiments. The experiments include 50,000 searches in several different environments. From the results, it is clear that quadtree maps generally out perform grid maps with a big margin and the four variants of path search algorithm on quadtree maps demonstrate exactly the characteristics we expected–as search time decreases path length increases. Therefore, in our simulations quadtree maps are used a lot more than grid maps for non-detailed path planning, in order to save computation time (see Table 7.5).

Chapter 8

Applications, Results and Analysis

Our pedestrian animation system, which comprises on the order of 50,000 lines of C++ code, enables us to run long-term simulations of pedestrians in large-scale urban environments without manual intervention.

8.1 Autonomous Pedestrians in Penn Station

We have employed the environmental modeling techniques described in Section 3.1 and Chapter 6 to represent the original Pennsylvania Train Station of New York City (Figure 1.1 and the left column of Figure 1.2) whose 3D reconstruction is illustrated in the right column of Figure 1.2 and in roofless plan view in Figure 3.1. The entire 3D space of the station (200m (l) $\times$ 150m (w) $\times$ 20m (h)) is manually divided into 43 regions. The initial graph map for the Penn Station model is constructed according to this division and the *path-to-via* information of every region is computed automatically. For each region, all the objects are automatically loaded and abstracted into the appropriate maps. For an obstacle, a polygonal bounding box is included, while for a ground object, a sampled height function is stored. Orientations are specified for objects such as chairs, newsstands, and vending machines. Special labels are stored for platforms/tracks and for trains to different destinations. A registered object also stores reference pointers to associated representations in coarser-resolution maps. At run time, our environment model requires approximately 90MB of memory to accommodate the station and all the objects.

A text configuration file is used for the initialization of pedestrians. To distinguish individual pedestrians not only by their appearance, we define, in this file the kinetic limits and personal preferences for each pedestrian. These personal traits include body inertia, bounding box size, preferred walking speed, maximal angular speed during turning, motion style and characteristic, preferred safe distance from others, sensing range, crowding willingness, internal state variable changing range and pattern, character type (commuter, policeman, etc.) and so on. In this file, we also specify in which of the four ways each pedestrian shall be initialized—either put the pedestrian at a random or a specific position that is valid (standing on the floor and not intersecting with any solid object), or cue the pedestrian to enter the station from a random or a specific entrance after the simulation starts.

In our simulation experiments, we populate the virtual station with five different types of pedestrians. Most of them are classified as commuters. In addition, there are tourists, performers, policemen, and patrolling soldiers. Our simulations demonstrate not only conventional crowd behaviors, in some cases involving over a thousand pedestrians, but various individual activities that are typical for real pedestrians in a train station as well. With every individual guided by his/her own autonomous control, they act out their volition, coordinate with other strangers based on both observation and personal preferences, and at the same time follow simple common-sense rules. These autonomous characters imbue the virtual train station with liveliness, social (dis)order, and a realistically complex dynamic.

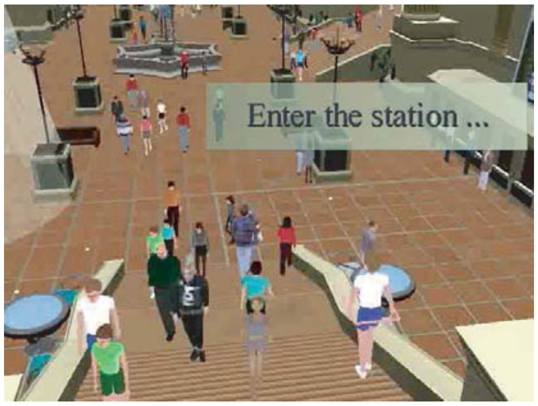
8.1.1 Animation Examples

We will now describe several representative simulations that demonstrate specific functionalities of our system. To help place the animation scenarios in context, refer to Figure 3.1.

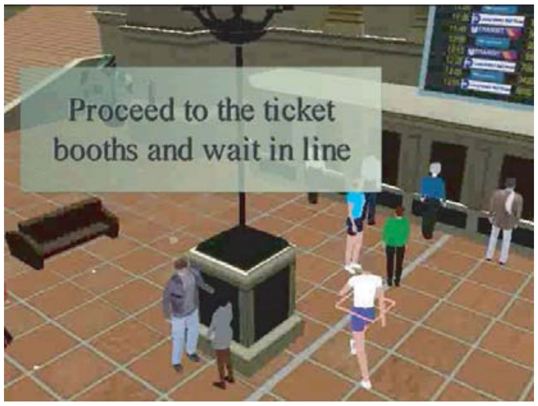
Following an Individual Commuter

As we claimed earlier in Section 2.2.5, an important distinction between our system and existing crowd simulation systems is that we have implemented a comprehensive human model, which makes every pedestrian a complete individual with a richly broad behavioral and cognitive repertoire. In this animation, therefore, we choose a commuter and follow him as he moves through the station.

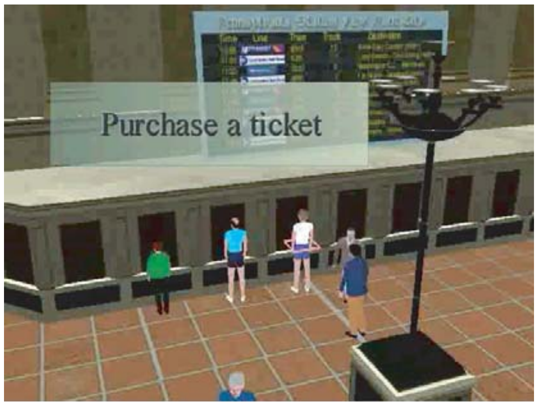
In the animation, our subject enters the station, proceeds to the ticket booths in the main waiting room, and waits in a queue to purchase a ticket at the first open booth. Having obtained a ticket, he then proceeds to the concourses through a congested portal. Next, our subject feels thirsty and spots a vending machine in the concourse. He walks toward it and waits his turn to get a drink. Feeling a bit tired, our subject finds a bench with an available seat, proceeds towards it, and sits



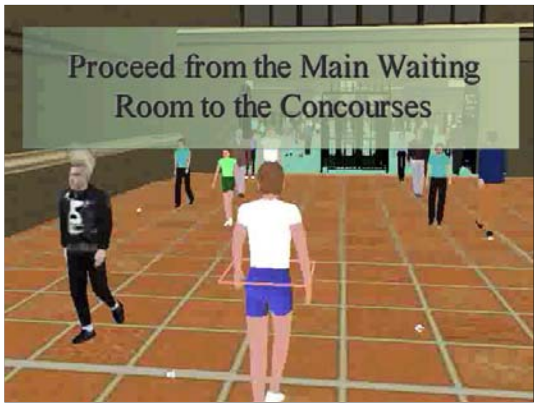
(a)



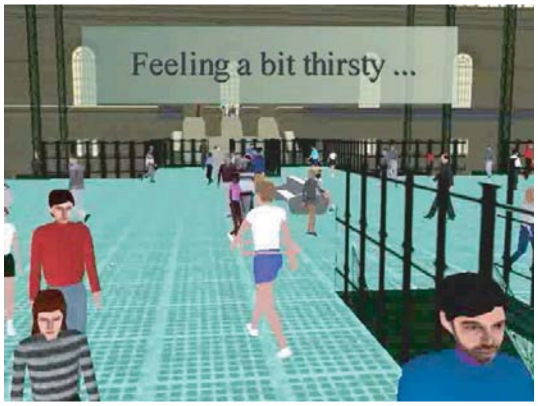
(b)



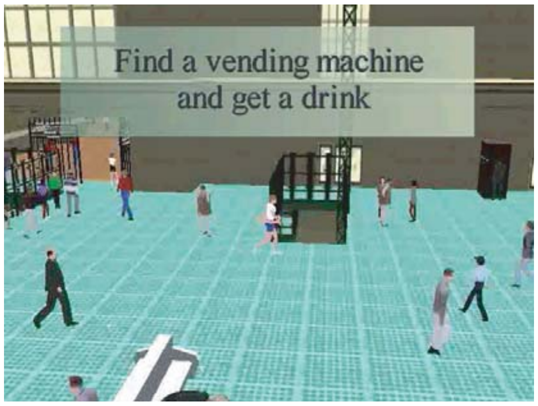
(c)



(d)

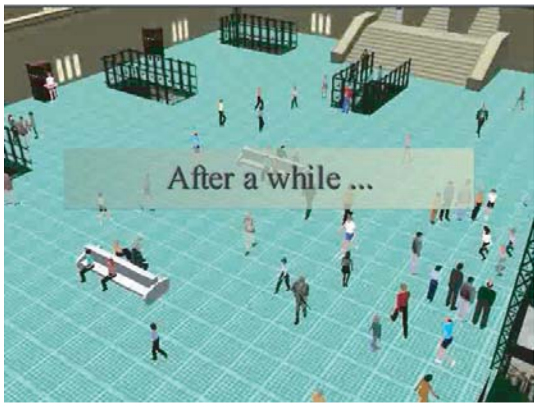


(e)



(f)

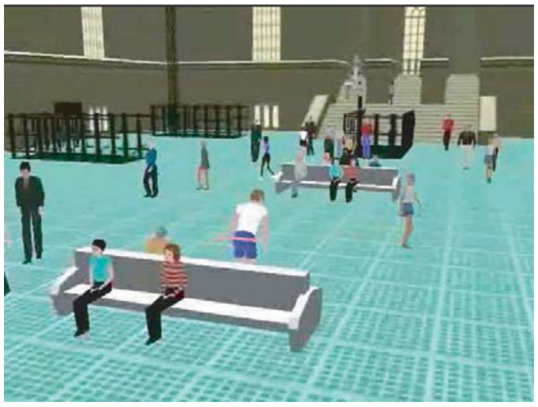
Figure 8.1: Following an individual commuter.



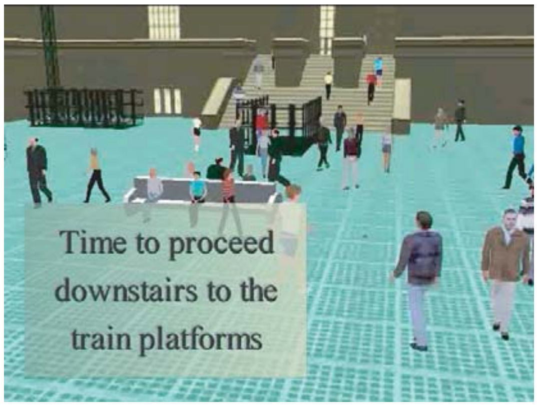
(g)

Feeling a bit tired...
it's still early, so why not find a seat?

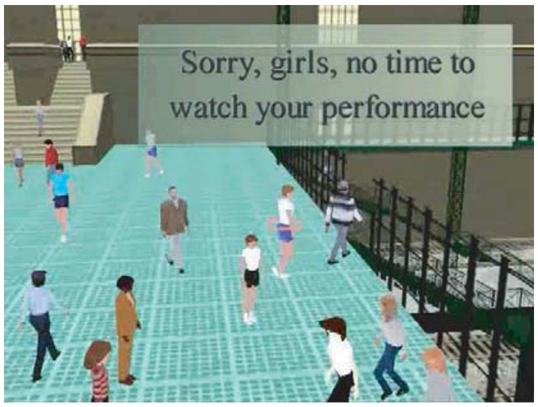
(h)



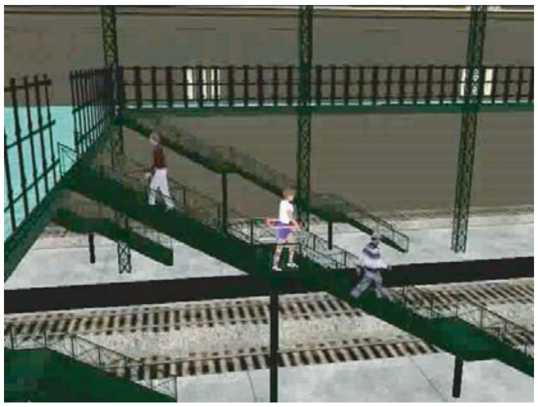
(i)



(j)



(k)



(l)

Figure 8.2: Following an individual commuter (continued).

down. Later, the clock chimes the hour and it is time for our subject to get up and proceed to his train platform. He makes his way through a somewhat congested area by following, turning, and stopping as necessary in order to avoid bumping into other pedestrians. He passes by some dancers that are attracting interest from many other pedestrians, but our subject has no time to watch the performance and descends the stairs to his train platform. Snapshots of this animation are given in Figure 8.1 and Figure 8.2.

Pedestrian Activity in the Train Station

The second simulation, which includes over 600 autonomous pedestrians, demonstrates a variety of pedestrian activities that are typical for a train station. We can interactively vary our viewpoint through the station, directing the virtual camera on the main waiting room, concourse, and arcade areas in order to observe the rich variety of pedestrian activities that are simultaneously taking place in different parts of the station. Some additional activities that were not mentioned above in the first animation include pedestrians choosing portals and navigating through them, congregating in the upper concourse to watch a dance performance for amusement, and proceeding to the train platforms using the rather narrow staircases. Snapshots of this animation are shown in Figure 8.3.

8.1.2 Performance

We have run various simulation tests on a 2.8GHz Intel Xeon system with 1GB of main memory. The total length of each test is 20 minutes in virtual world time. Time costs of these tests are measured and plotted in Figure 8.4. The

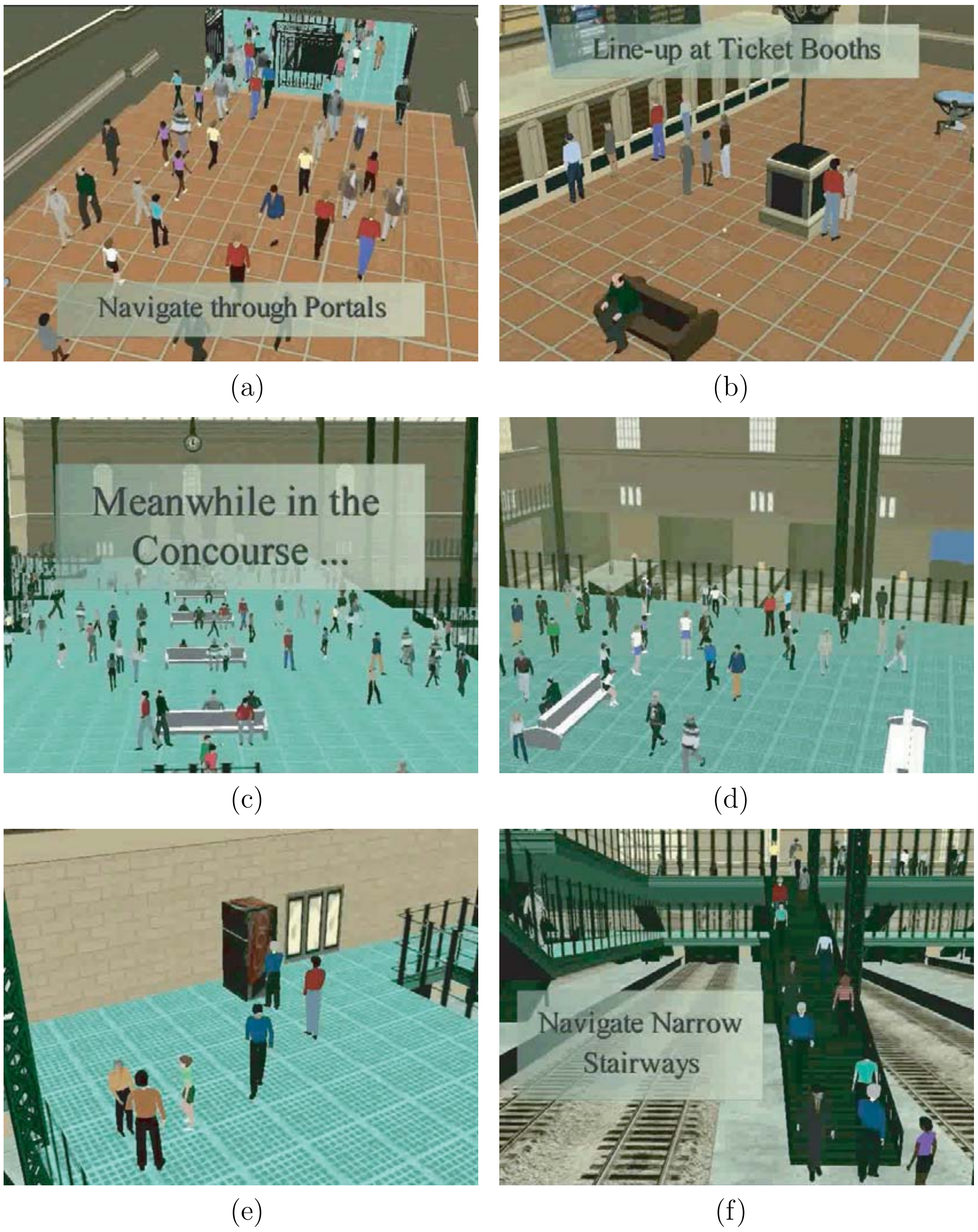


Figure 8.3: Pedestrian activity in a train station.

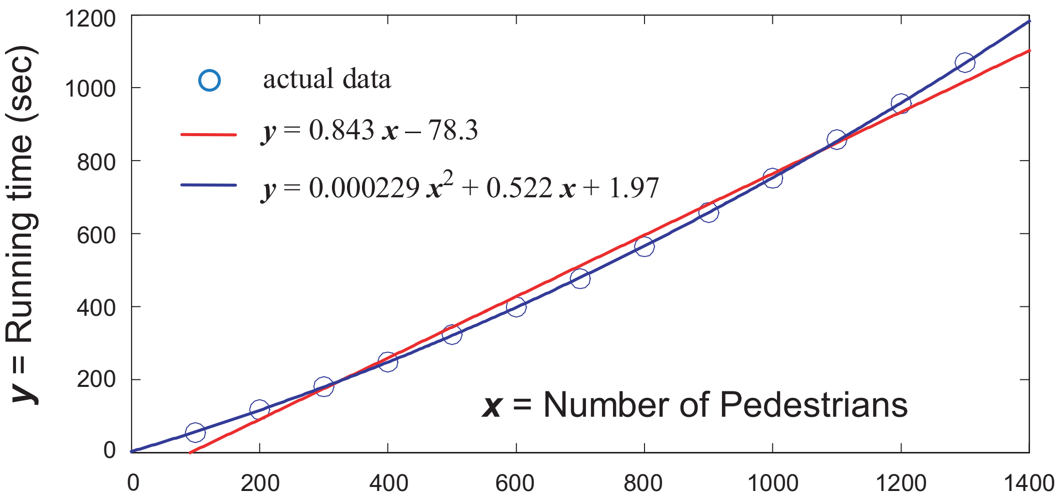


Figure 8.4: Pure simulation time *vs.* number of pedestrians.

simulation times reported here include only the requirements of our algorithms—cognitive control, behavioral control, perceptual query, and motor control for each pedestrian plus environment model update. If we look at Figure 5.1, then the only thing we excluded from the time cost measurement is the DI-Guy rectangle at the very bottom, which is not controllable by us.

In the figure, the plotted curve indicates that computational load increases almost linearly with the number of pedestrians in the simulation. It shows that real-time simulation can be achieved for as many as 1400 autonomous pedestrians (i.e., 20 virtual world minutes takes 20 minutes to simulate at 30 frames/second). Although the relation is best fit by a quadratic function, the linear term dominates by a factor of 2200. The small quadratic term is likely due to the fact that the number of proximal pedestrians increases as the total number of pedestrians increases, but with a much smaller factor.

Figure 8.5 breaks down the computational load for various parts of the simula-

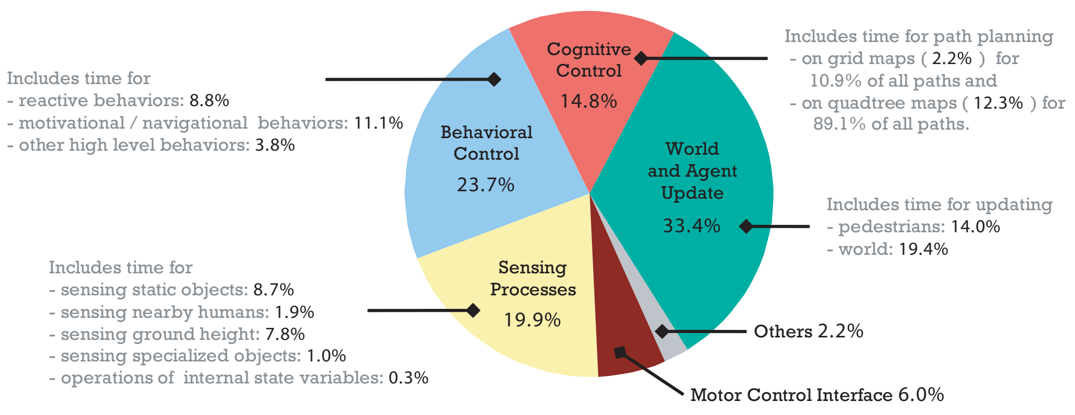


Figure 8.5: Computation time of the various components of the system.

tion based on multiple experiments with different numbers of pedestrians ranging from 100 to 1000 in the Penn Station model on the aforementioned PC. The computation time inside *DI-Guy* and the rendering time is disregarded here.

Number of Pedestrians	**0**	**100**	**200**	**300**	**400**	**500**
Simulation Only	n/a	64.4	32.2	23.0	16.9	12.3
Rendering Only	21.0	12.5	9.2	7.6	6.0	5.4
Simulation + Rendering	21.0	10.5	7.2	5.7	4.4	3.8

Table 8.1: Frame rate in different test settings.

Table 8.1 gives the frame rates that our system achieves on the aforementioned PC with an *NVIDEA GeForce 6800 GT AGP8X 256MB* graphics system. The frame rates listed here (in frames/sec) are measured from tests with various number of *DI-Guy* characters in the Penn Station environment. Unlike the previous measurements, we include computation of *DI-Guy* and rendering selectively in our tests listed in this table. In "simulation only" tests, full simulation is done (including *DI-Guy*) without rendering. In "rendering only" tests, all simulation algorithms are disabled and static scene with stationary characters is rendered.

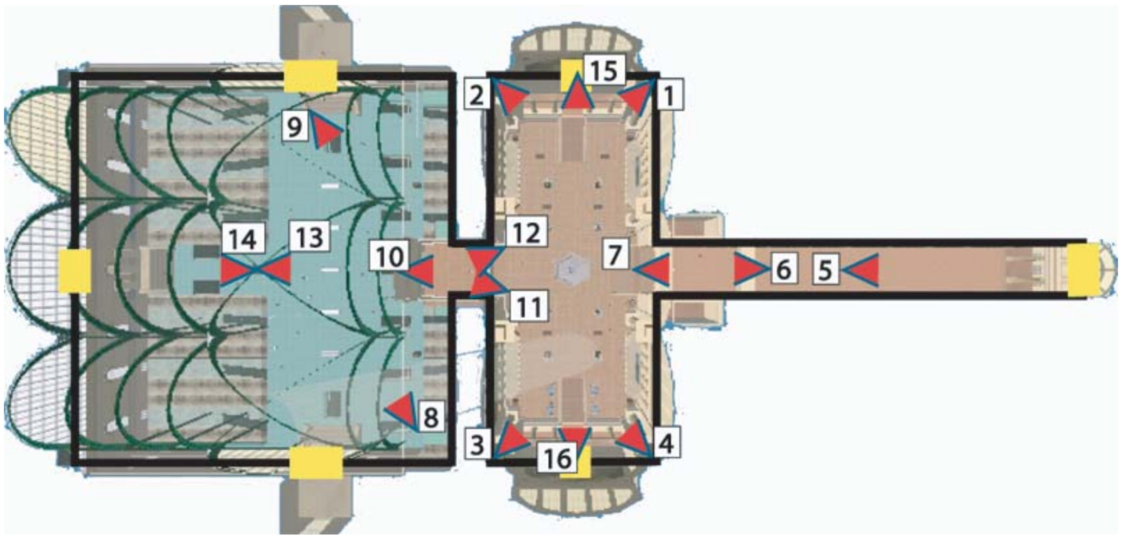


Figure 8.6: Virtual surveillance camera layout in Penn Station. Plan view of the virtual Penn Station environment (the yellow rectangles indicate station portals) illustrating an example visual sensor network comprising 16 simulated active (pan-tilt-zoom) video surveillance cameras. Image provided courtesy of F. Qureshi.

And in "simulation + rendering" everything is enabled. As shown in the table, rendering times dominate pedestrian simulation times and, consequently, lower the frame rate during full simulation tests. This is probably due to the geometric complexity of both the Penn Station model and numerous pedestrians. We believe this rendering bottleneck can be ameliorated by using existing techniques such as culling, impostors, levels of detail, and so on. However, this is beyond the scope of this thesis.

8.1.3 Application to Computer Vision & Sensor Networks

Our autonomous pedestrian simulator can be of use to researchers in the domains of computer vision and visual sensor networks in an approach called *virtual vision*

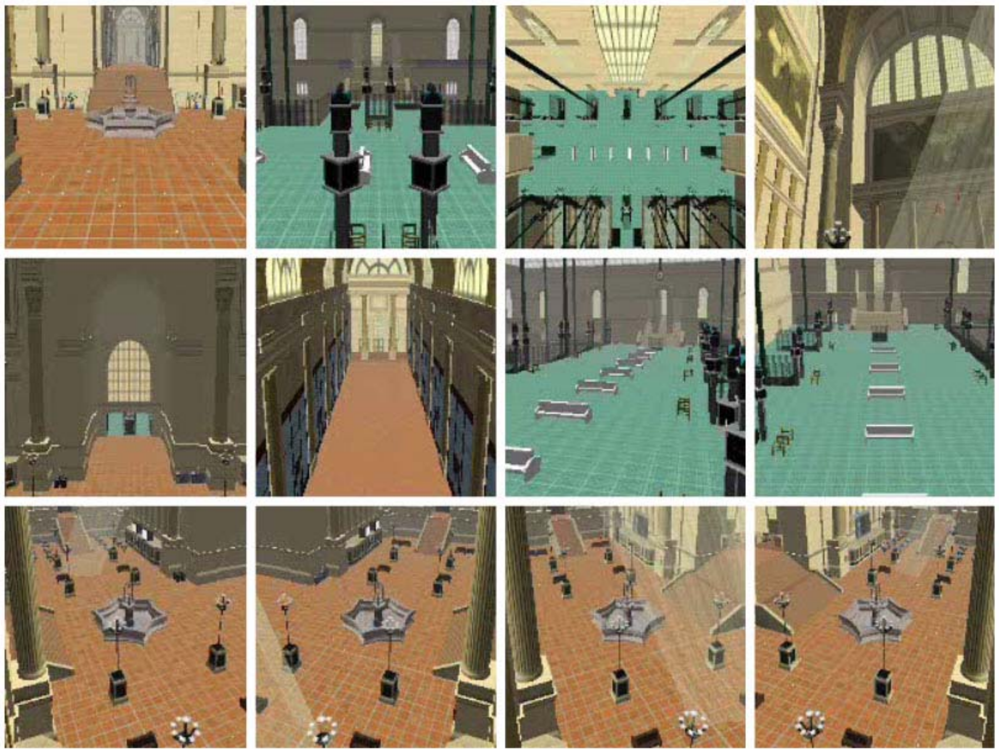


Figure 8.7: Virtual vision. Synthetic video feeds from multiple virtual surveillance cameras situated in the (empty) Penn Station environment.

[Terzopoulos et al. 1994; Qureshi and Terzopoulos 2005b; Qureshi and Terzopoulos 2005a]. In particular, our simulator is currently serving as the core of a software testbed for the design and essaying of multi-camera automated sensor networks for visual surveillance, which may eventually be capable of detecting suspicious activity and behavior. To this end, virtual cameras are situated at various locations in our virtual Penn Station environment populated by autonomous pedestrians (Figure 8.6). The virtual cameras generate synthetic video feeds that emulate those generated by real surveillance cameras monitoring public spaces (Figure 8.7). Figure 8.8 illustrates the virtual sensor network developed by Qureshi and Terzopoulos [2005a], which comprises multiple active (pan-tilt-zoom) cameras, automatically tracking a female pedestrian wearing the red top as she proceeds through the station.

In the context of the virtual vision approach, our simulator offers several advantages, among them the following:

- Deploying a large-scale visual sensor network in the real world is a major undertaking whose cost can easily be prohibitive for most researchers interested in designing and experimenting with sensor networks. Despite its sophistication, our simulator runs on high-end commodity PCs, obviating the need to acquire and grapple with special-purpose hardware and software.

- Unlike the real world, researchers can readily deploy a virtual sensor network in our simulated world and easily reconfigure the multi-camera layout in the virtual environment.

- The use of realistic virtual environments in computer vision and sensor network research offers significantly greater flexibility during the design and evaluation cycle, thus expediting the engineering process.

- The virtual world provides readily accessible ground-truth data for the purposes of visual sensor network algorithm validation.

- The hard real-time constraints of the real world can be relaxed in the simulated world; i.e., simulation time can be prolonged relative to real, "wall-clock time", in principle permitting arbitrary amounts of computational processing to be carried out during each unit of simulated time.

- Privacy laws generally restrict the monitoring of people in public spaces for

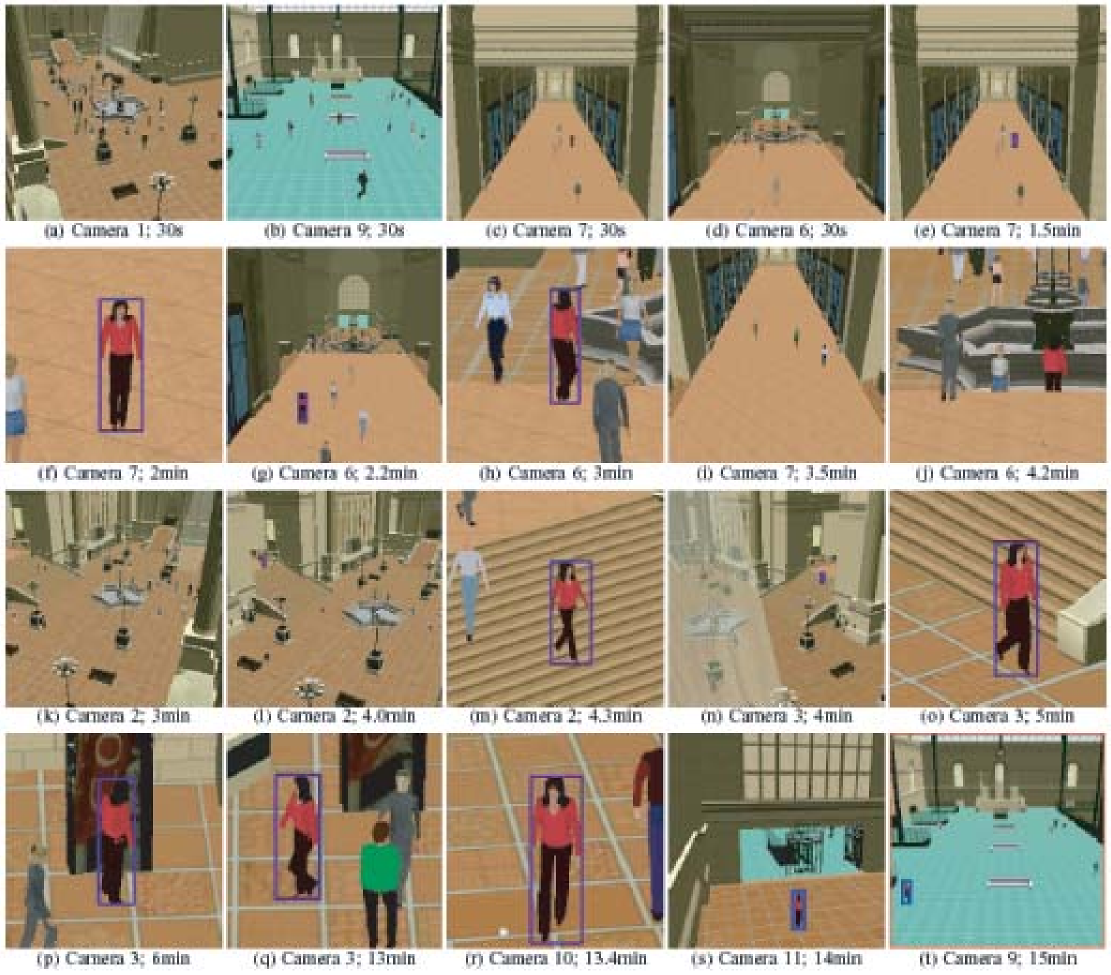


Figure 8.8: A pedestrian is tracked by autonomous virtual cameras. Image provided courtesy of F. Qureshi.

experimental purposes. We do not violate the privacy of any real people in our virtual world.

- Obviously, in the virtual world researchers can potentially experiment with complex and/or dangerous scenarios that cannot be attempted safely or repeatedly in real life. For example, one can potentially simulate the reactions of crowds in crisis situations, such as the presence of an armed terrorist, and see if one can design suites of computer vision algorithms that, through an automated analysis of the virtual video streams, can detect such situations and issue appropriate alerts.

A thorough description of ongoing virtual vision research is beyond the scope of this thesis. For further details, we refer the reader to the above cited and forthcoming publications of Qureshi and Terzopoulos.

8.2 Virtual Archaeology

Archaeology is another domain that can benefit from the application of autonomous virtual pedestrians, particularly for the purposes of visualizing human activity in computer reconstructions of archeological sites and, potentially, for testing of archeological theories of site usage in ancient times.

8.2.1 The Petra Archeological Site and the Great Temple

Petra (from the Greek word for “rock”) lies in a great rift valley east of Wadi Araba in Jordan about 80 kilometers south of the Dead Sea. As the principal city of ancient Nabataea at its heyday, Petra has a history that can be traced

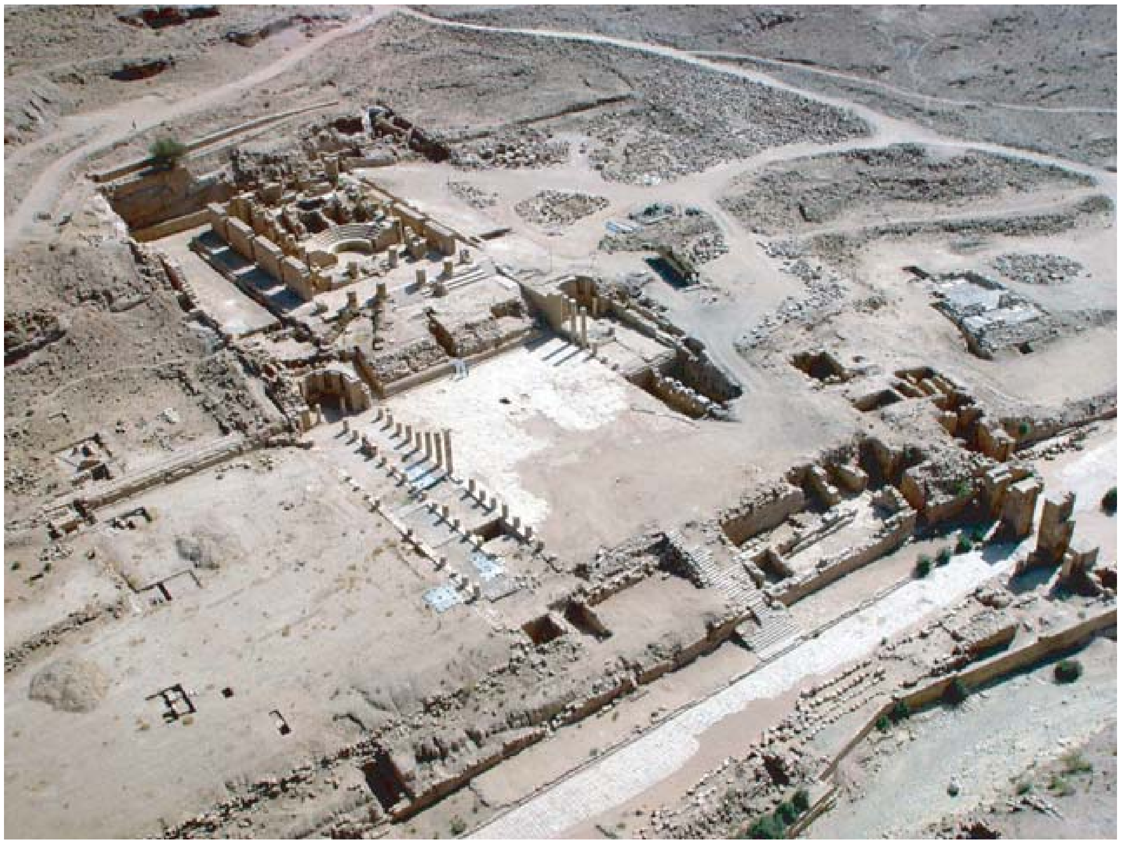


Figure 8.9: Site of Petra Great Temple.

back over 3,000 years. The Great Temple of Petra represents one of the major archaeological and architectural components of central Petra. It is now recognized by archaeologists to have been one of the main buildings of the ancient city of Petra and it is thought to have been dedicated to the most important cult deity of the city. Unfortunately, earthquakes in ancient times demolished the temple and buried it under debris.

The temple site has been explored since the 1890s. A major excavation recently carried out by an archeological team from Brown University has unearthed many amazing structures and sculptures [Joukowsky 1998]. With the new find-

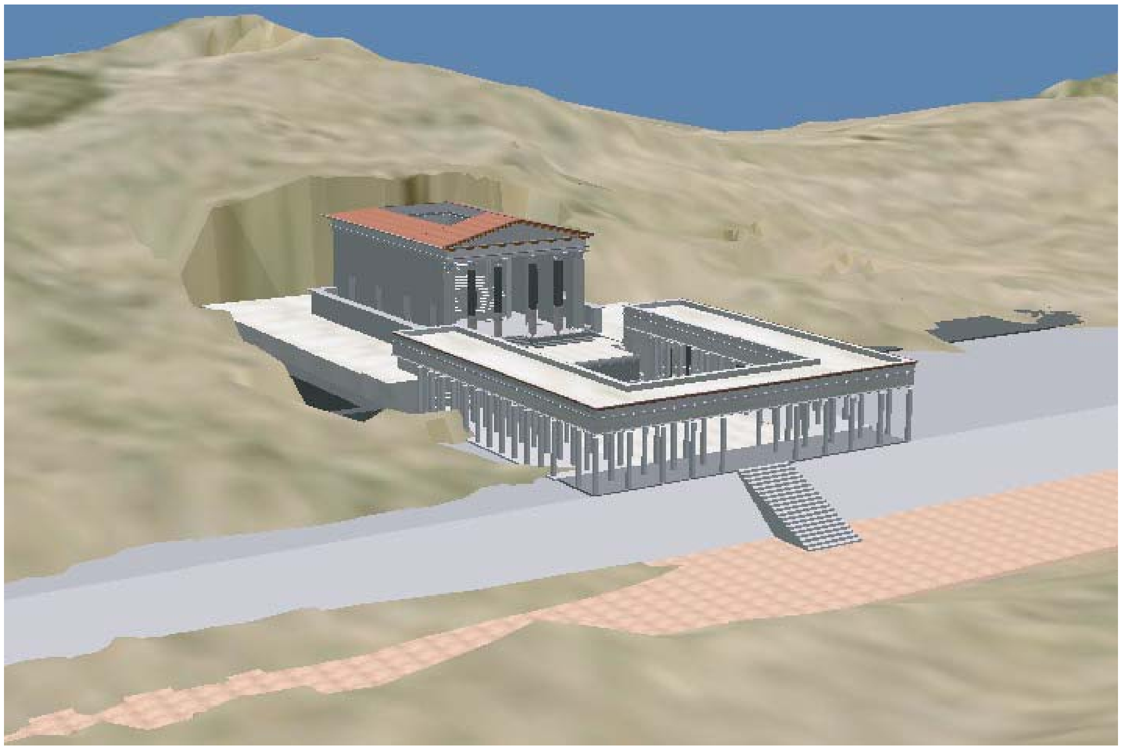


Figure 8.10: Reconstructed 3D model of Petra Great Temple.

ings, archaeologists, computer scientists, and artists have been collaborating to reconstruct the original appearance of the temple in a virtual 3D model [Vote et al. 2002]. Figure 8.9 shows a photo of the temple site under excavation and Figure 8.10 shows a CG image rendered from the reconstructed 3D model.

Located on the southern citadel hill of Petra, the 7560 square meters Temple Precinct is comprised of a Propylaeum (monumental entryway), a Lower Temenos (with two colonnades on the east and west sides), and monumental east and west Stairways, which lead to the Upper Temenos. Further ahead is the Great Temple itself. This impressive structure measures 28 meters in width and some 42 meters in length. Inside this structure, there is a small amphitheater with circular tiers of stair seats surrounding a restricted stage area. Figure 8.11 shows the layout of

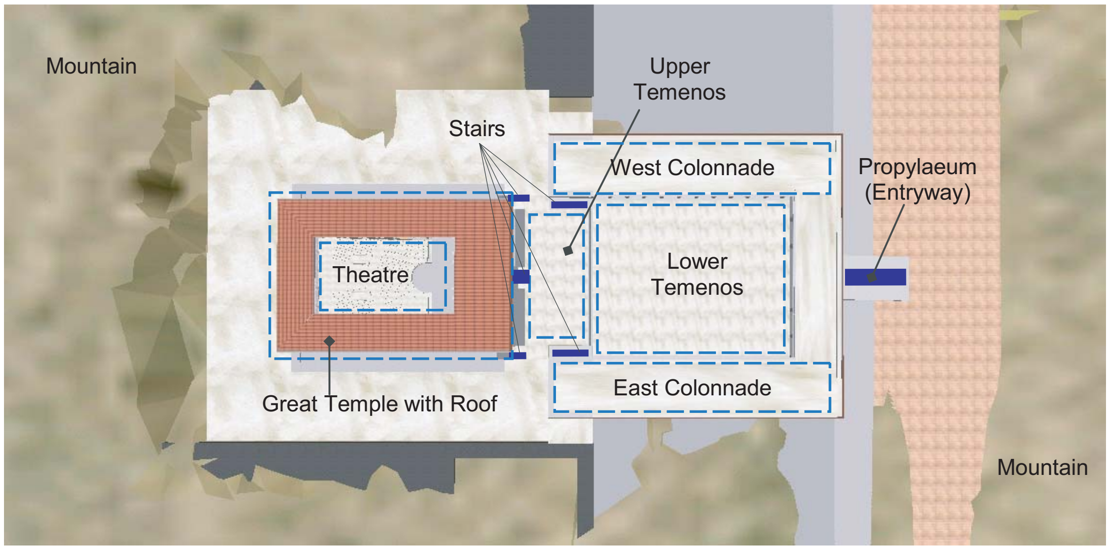


Figure 8.11: Layout of the temple precinct.

the various parts of the Temple Precinct.

As the Temple has sustained severe damage and most of its parts have not yet been found, many aspects of the Temple, from the layout of missing structures to details of the sculptures to the purpose and use of the Temple and the amphitheater inside, remain unclear. Archaeologists must use scholarly speculation while studying the site. Assimilating the reconstructed temple model within our autonomous pedestrian simulation system, we are able to visualize possible interactions between the temple and its human occupants, thus potentially serving as a tool to archaeologists in their speculations about various functional aspects of the Temple.

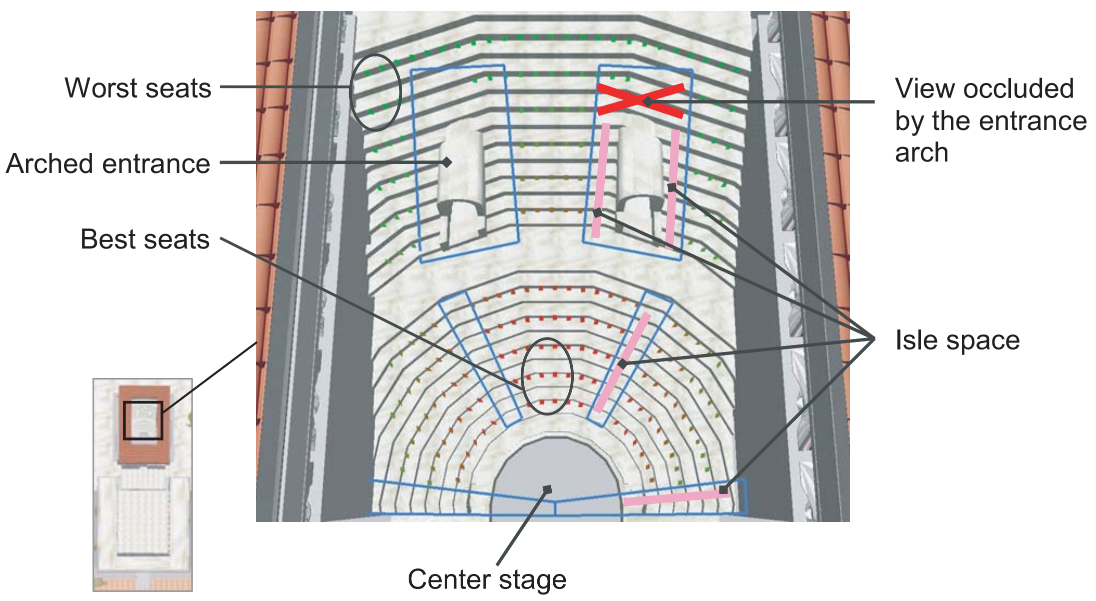


Figure 8.12: Layout of the amphitheater. Seats are color-coded from red (best seats) to green (worst seats). Spacing between neighboring seats is defined as $0.6m$ in left-right direction and $1.0m$ in front-back direction. The setting shown includes 201 seats.

8.2.2 Temple Precinct Environmental Model

We divide the entire space of the Temple Precinct, which can be bounded by a 180m (l) × 60m (w) × 30m (h) box, into 20 regions, including the entrance area, the lower and upper squares, the temple, the amphitheater, and the stairs connecting different regions, including those beneath the amphitheater that lead to its auditorium. Objects such as columns and walls are automatically loaded during map initialization. This large model consists of over 410,000 triangles and occupies about 22 MB of memory for its geometry and textures. To prepare the entire model for use by our autonomous pedestrians, our environment data structures consume an additional 65 MB of memory.

A key issue of interest from the archaeological perspective is to determine how

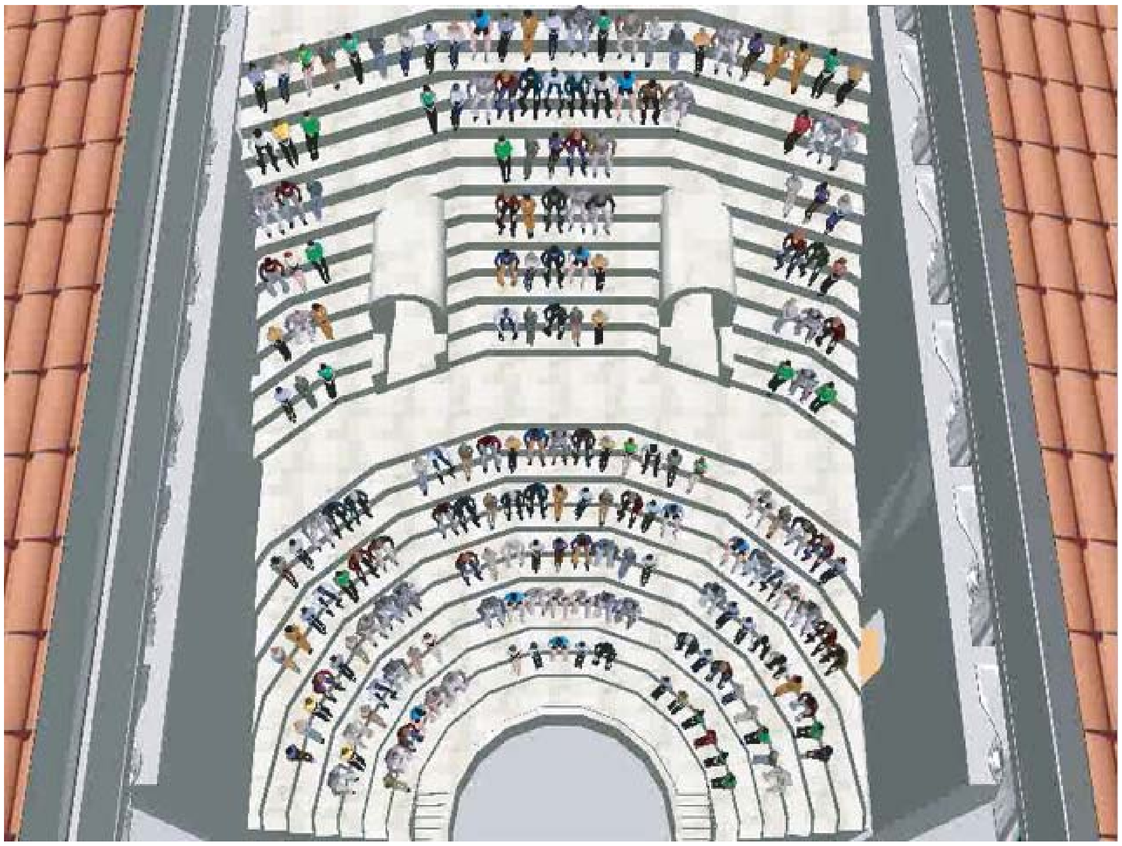


Figure 8.13: Amphitheater filled with audience (201 people seated).

many people can sit in the amphitheater and how efficiently they enter and exit it. To this end, we developed a new specialized environment object to model the amphitheater inside the temple. According to several user-specified parameters loaded upon initialization, the theater object defines the locations of the stage and the auditorium including the arrangement of the seats. During simulation, the environment object keeps track of the size of the audience and where each member of the audience sits.

In our simulations, we set the amphitheater parameters such that isle space is reserved in order for people to reach their seats, even when the auditorium is almost full. Regions from which the stage area is largely occluded are excluded as possible seating areas. Figure 8.12 illustrates the amphitheater layout in our simulation tests.

8.2.3 Simulation Example and Results

An animation example that is typical of our simulation tests within the Petra Great Temple environment unfolds as follows: In the beginning, the simulation demonstrates hundreds of pedestrians entering the Temple Complex through the Propylaeum—its grand entryway. On the Lower Temenos, the stream of pedestrians separates to approach the two stairways on the east and west sides. Proceeding to the Upper Temenos, the pedestrians enter the Great Temple via the three small staircases. Once inside the Temple, pedestrians approach the amphitheater entrance stairs located on the east and west sides beneath the amphitheater. Through two arched gates each pedestrian autonomously enters the auditorium, selects a seating area, determines a way to get to a selected seat and sits down. After everyone has entered and taken a seat, a "tour guide" comes to the center stage to enact the delivery of a short lecture about the Great Temple. At its conclusion, the audience rises and starts to evacuate the amphitheater through the two narrow arched gates, which are hardly wide enough to accommodate two pedestrians side by side. Exiting the same way they entered, the pedestrians leave the Temple through the Propylaeum. Successive sample frames from the animation are shown in Figures 8.14–8.16.

Note that in order to function properly in the described scenario within the amphitheater environment, our autonomous pedestrians require several naviga-

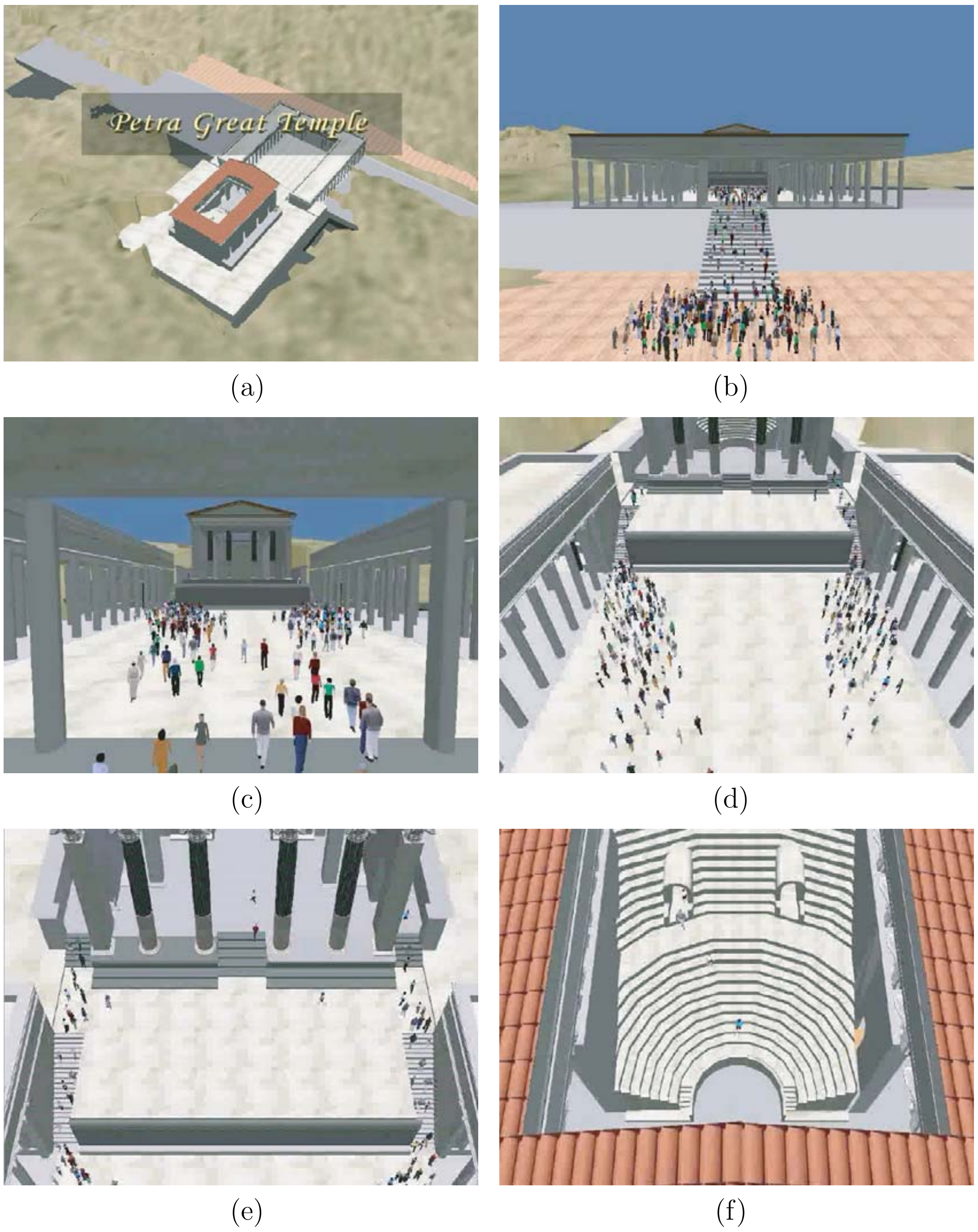
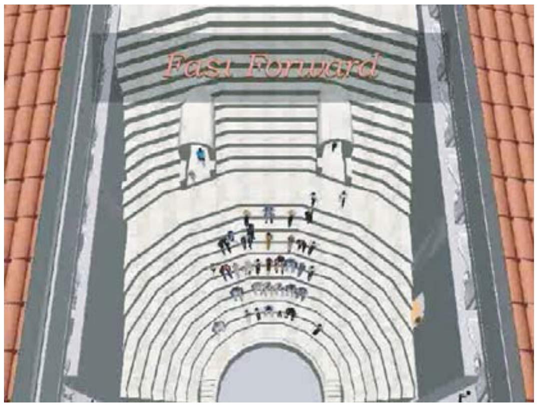
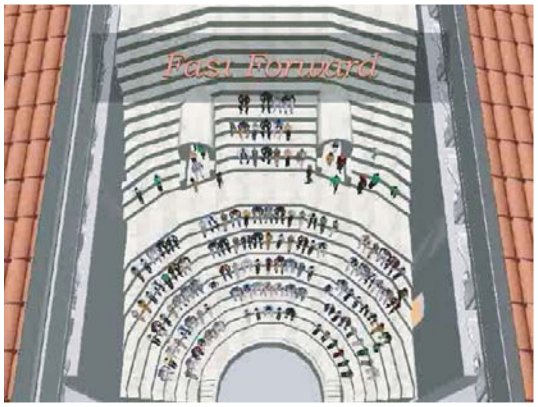


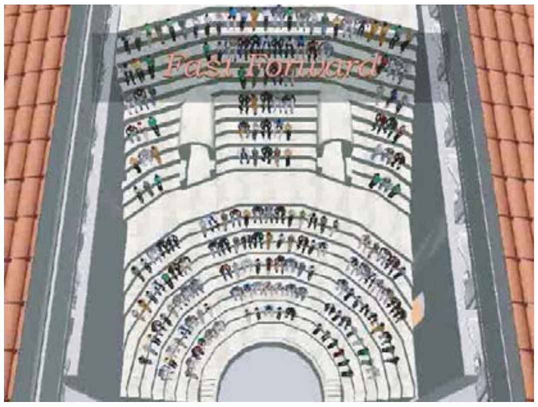
Figure 8.14: Petra Great Temple animation snapshots.



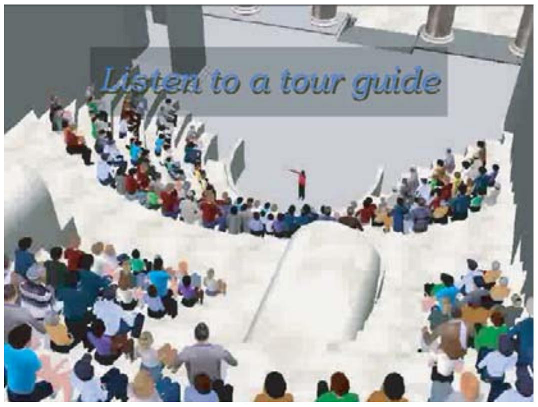
(g)



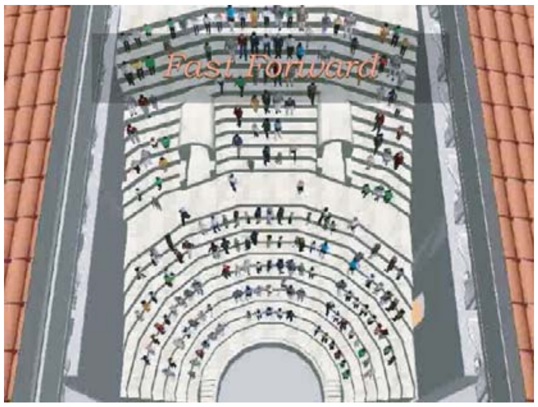
(h)



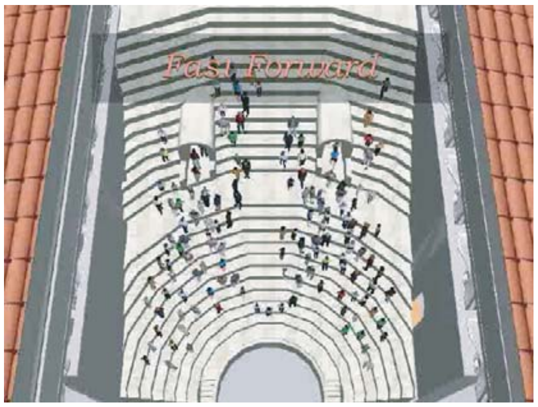
(i)



(j)

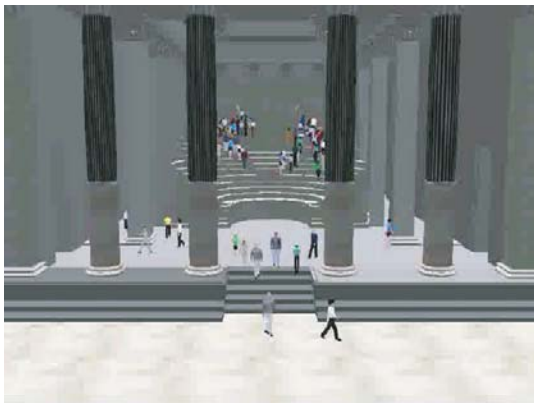


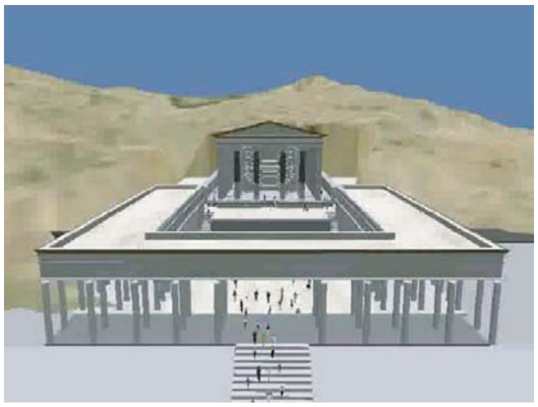
(k)



(l)

Figure 8.15: Petra Great Temple animation snapshots (continued).


(m)


(n)

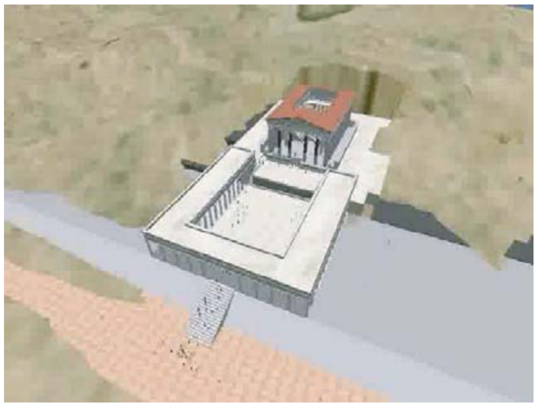

(o)

Figure 8.16: Petra Great Temple animation snapshots (continued).

tional and motivational behavior routines in addition to those that we developed in Chapter 4. In particular,

1. a seat selection behavior to pick a seat in the auditorium;
2. a navigation behavior to reach the selected seat to sit down; and
3. an exit behavior to leave the auditorium.

The first routine is similar to those described in Appendix B. In the second routine, a pedestrian will regard other seated pedestrians as static obstacles rather than mobile objects, and will appropriately apply detailed-path-planning and static obstacle avoidance routines to reach the selected seat. Finally, in the third routine, a pedestrian will simply follow pedestrians ahead of him/her in order to exit the auditorium.

Our simulation experiments reveal that, given the structure of the amphitheater and with an inter-personal distance (i.e., the distance between the centers of two neighboring people) set to 0.6m for left-right and 1.0m for front-back, the amphitheater can accommodate approximately 201 people comfortably (Figure 8.13). It requires approximately 7–8 minutes for the audience to enter and fill the amphitheater and approximately 5 minutes to exit and evacuate the amphitheater. Note that the two arched stairways leading from underneath the amphitheater to its auditorium are the only avenues for people to enter and exit; thus, as expected, they become bottlenecks to pedestrian traffic.

Chapter 9

Conclusion

In a departure from the substantial literature on so-called "crowd simulation", we have developed a decentralized, comprehensive model of pedestrians as autonomous individuals capable of a broad variety of activities in large-scale synthetic urban spaces. Our artificial life approach spans the modeling spectrum from pedestrian appearance, motion, perception, behavior, to cognition.

In addition, we presented a methodology for modeling large-scale urban environments that facilitates the animation of numerous autonomous virtual pedestrians. Our environment model, which involves a set of hierarchical data structures (a topological map, perception maps, path maps and specialized objects), supports the efficient interaction between pedestrians and their complex environment, including perceptual sensing, situation interpretation, and path planning at different scales.

We developed a simulator that incorporates the autonomous pedestrian model and the environmental model. Our simulator enables us to deploy a multitude of self-animated virtual pedestrians within a large indoor environment, a VR reconstruction of the original Pennsylvania Train Station in New York City. We also briefly described the application of our Penn Station simulator and its autonomous virtual pedestrians in the domain of computer vision for the design and evaluation of visual sensor networks. Finally, in the domain of virtual archaeology, we described how our autonomous pedestrian model serves in the visualization of urban social life in reconstructed archaeological sites, specifically the Great Temple precinct of ancient Petra. Our simulation results in each of these application scenarios speak to the robustness of our system and its ability to produce prodigious quantities of intricate animation of pedestrians carrying out various individual and group activities suitable to their environment.

9.1 Limitations and Future Work

Although motion artifacts are at times conspicuous in our animation results, primarily due to the limitations of the underlying *DI-Guy* software, the architecture of our simulation system facilitates the potential replacement of this low-level software package by a better character rendering and motion synthesis package should one become available.

Despite the sophisticated pedestrian models that we have developed, the gap between the abilities of our virtual pedestrians and those of real people remains substantial. To further close the gap, additional behavioral abilities and reasoning facilities must be added to the current model. Additional future work on our models and our simulation system can be done along the following directions:

Cooperating Pedestrians: In our work, we have focused on modeling pedestrians as highly competent *individuals* through a comprehensive artificial life modeling approach that integrates motor, perceptual, behavioral, and cognitive components. However, we have barely modeled the additional interactions inherent to small groups of cooperating pedestrians (e.g., friends or family units), which can be important to the realism of pedestrian animation within urban environments. To this end, models of human relationships and non-verbal communication can be introduced. Relationships between characters will affect their behaviors. For example, two pedestrians walking together will try to remain close to each other. They may have different choices in certain situations when walking alone, but they will coordinate their choices when walking together. Any conflict can be resolved either by leadership in an asymmetric relationship (father and son) or by non-verbal communication (such as gestures) in a symmetric one (such as close friends).

Perception: Although the perceptual processes employed by our pedestrians are efficient and they serve the behavior routines well, most of them employ shortcut algorithms that make use of the world database directly. On the one hand, such perceptual mechanisms guarantee the correctness of the extracted information. On the other hand, however, it does not attempt to closely model the actual perceptual processes employed by real humans. A more sophisticated approach along these lines is referred to as synthetic vision by Noser et al. [1995]. Terzopoulos and Rabie [1996; 1999] introduce a more biomimetic approach, which models active, foveated visual sensors ("eyes") and active computer vision processing for sensorimotor control.

Perceptual Inference: Furthermore, as we have pointed out, meaningful interpretation of perceived situations is of greater value to decision making for autonomous characters than raw sensed data. Currently, however, such interpretations are computed in an *ad hoc* manner within the world model of our system. This choice is due to the high complexity of the problem and concerns about efficiency. In view of ever increasing computational power and given advances in computer vision and artificial intelligence, it may soon be feasible for each individual to infer high level information (e.g., "long wait line", "crowded portal", etc.) from low level sensed data (e.g., "many people") in a biologically plausible manner.

Head (and Eye) Movement: Given the quality of the behaviors that our pedestrians demonstrate, their lack of head movements has begun to become increasingly conspicuous. Their upper bodies seem rigid relative to their natural navigation. Therefore, it is our intention to develop a satisfactory set of subconscious, reactive and deliberative head and eye motion behaviors for our virtual pedestrian model. As eyes are the organs for visual perception, these new head and eye movements should be correlated with perceptual attention and will in turn introduce more intricacy into the behavioral control. A well-designed coupling between perceptual attention and head/eye movement will then become necessary.

Manipulation: For the same reason mentioned above, one should also try to imbue our pedestrians with useful manipulation skills, such as the necessary upper body motions for making purchases at ticket booth or vending machines. Also, it hasn't escaped our notice that our train station simulations would be more realistic if some virtual pedestrians toted luggage.

Alternatives for Reactive Behaviors: The optimal sequencing of our six reactive behavior routines enables our pedestrians to navigate safely in a dynamic world. Despite the satisfactory performance, this may not be the ideal approach. Alternatively, it might be possible to integrate these reactive behavior routines in parallel rather than sequentially, or one can devise a different set of key routines. Furthermore, instead of matching part of the current obstacle situation with various "situation patterns" and then integrating the various responses to those patterns to get the final response, the situation may be considered in its entirety by a pedestrian in order to determine the appropriate reaction, as has been attempted in the machine learning community for steering mobile robots [LeCun et al. 2005].

Externalization and Agreement: Sociologist Erving Goffman points out:

> *"The workability of lane and passing rules is based upon two processes important in the organization of public life: externalization and scanning."*
>
> *(E. Goffman. 1971. Relations in public: Microstudies of the public order, Page 11.)*

Here, externalization is the "process whereby an individual pointedly uses overall body gesture to make otherwise unavailable facts about his situation gleanable" [Goffman 1971]. Scanning is a reciprocal process in which an individual, as he moves along, visually gauges the intentions of others which are in the front of a close circle around him. These two processes have been modeled to an extent in the sensory processes and the reactive behavioral level for our pedestrians such that they can avoid collisions with other pedestrians. Future work that would enable our pedestrians to resolve such conflicts more smoothly would engage a further mechanism—an implicit agreement established by mutual signals between conflicting pedestrians (through what Goffman called "body check" and "checked-body-check"). Such agreements are often employed in real life when two pedestrians tie in an avoidance situation. Sooner or later (usually after several attempts) they will agree on who takes what resource, mostly through non-verbal (sometimes verbal) communication. Such mechanisms should be implemented in our pedestrians.

User Interaction: Our system enables the user (animator) to change the way a simulation runs through configuration files, which are used to initialize the simulator. In these files, users can change numerous parameters, including environment settings (such as region decomposition) and pedestrian settings (such as total number, their types, personal traits, initializations, etc.). During real time simulation, a user can also control a pedestrian interactively by issuing motor control commands (locomotion speed and direction) via the keyboard, in a manner similar to that commonly used in computer games. Just like the motor control commands issued by the higher-level autonomous behavior controllers of the autonomous pedestrian model, the user's interactive motor commands will go through the set of reactive behavior routines for a safety check. The interface enables users to control a pedestrian to walk around without worrying about obstacle avoidance. We also provide an interactive interface for users to input high level directions to a pedestrian, such as asking him to go to a specific place, to get a ticket, or to find a seat and take a rest. Future work would give users the option of initializing a simulation via an interactive interface without using any configuration files. A more ambitious goal is a user to control several pedestrians at once.

Improving the Animation Frame Rate: We pointed out that the visualization frame rate of our fully rendered online simulation is dominated by rendering. We have already made use of levels of detail for both character geometry and motion as provided by the low-level *DI-Guy* API. However, the large number of pedestrians and the large Penn Station model impose a heavy rendering burden. By using advanced rendering techniques and tricks, such as culling and impostors, this problem may be mitigated in the future.

In closing, our animation results show that, like real humans, our autonomous virtual pedestrians demonstrate rational behaviors reflective of their internal goals and compatible with their external world. Although the gap between real and synthetic humans remains substantial, our progress makes us confident that the end result of related research in the fields of computer graphics and animation, artificial life/intelligence, robotics, simulation, as well as cognitive science, biology, physiology, psychology, ethology and sociology, will continue to narrow the gap until it eventually closes.

Appendix A

Ordering the Reactive Behavior Routines

In Chapter 4, we presented the six key reactive behavior routines and briefly described how they are activated sequentially in a best permutation ordering that we found via an exhaustive search (cf. [Reynolds 1993]), evaluating the performance of all 720 possibilities. Here in this appendix, we will present more details of this exhaustive search, explaining the criteria and discussing the result.

A.1 Fitness – The Performance Measure

For each permutation ordering of the six routines, an identical set of simulations given in Table A.1 is run and the results are summarized to a single value called "fitness". As a measure of the performance of a permutation, fitness is defined in Table A.2.

We put two factors, liveness and safety, in the definition of agent fitness, because

Environment Setting	**Number of Agents**	**Simulation Length (in Virtual Time)**
A Simple Synthetic Environment *	100	three 20-min simulations with different initial configurations
Penn Station	210	three 20-min simulations with different initial configurations
Penn Station	500	one 20-min simulation

Table A.1: The set of simulations for permutation search.
*: This environment is shown in Figure A.1.

we believe both of them are crucial to the realism of a pedestrian model. However, these two conflict with each other. Although we can always guarantee one of them by sacrificing the other, the most desirable is to have both of them in harmony—collisions shall surely be avoided and meanwhile pedestrians shall moving in their pleasant pace. The fitness value is such a measure that reveals the tradeoff between the two factors. It has the range from 0 to 1 and the closer to 1 the better.

A.2 Result Analysis

In Figure A.2, we show the plot of fitness values of all the 720 possible permutations. The y-axis in this plot shows the fitness score of each permutation ordering along the x-axis. The plot seems to be a chaos at the first glance. But when examining it carefully, we notice that in the middle of the diagram near $x = 240$, there is a sharp boost in fitness value. In fact, $x = 240$ corresponds to the last permutation ordering starting with Routine B ($B - F - E - D - C - A$) and $x = 241$ is the first starting with Routine C ($C - A - B - D - E - F$). The sudden increase happens exactly from $x = 240$ to $x = 241$ and fitness values stay high for

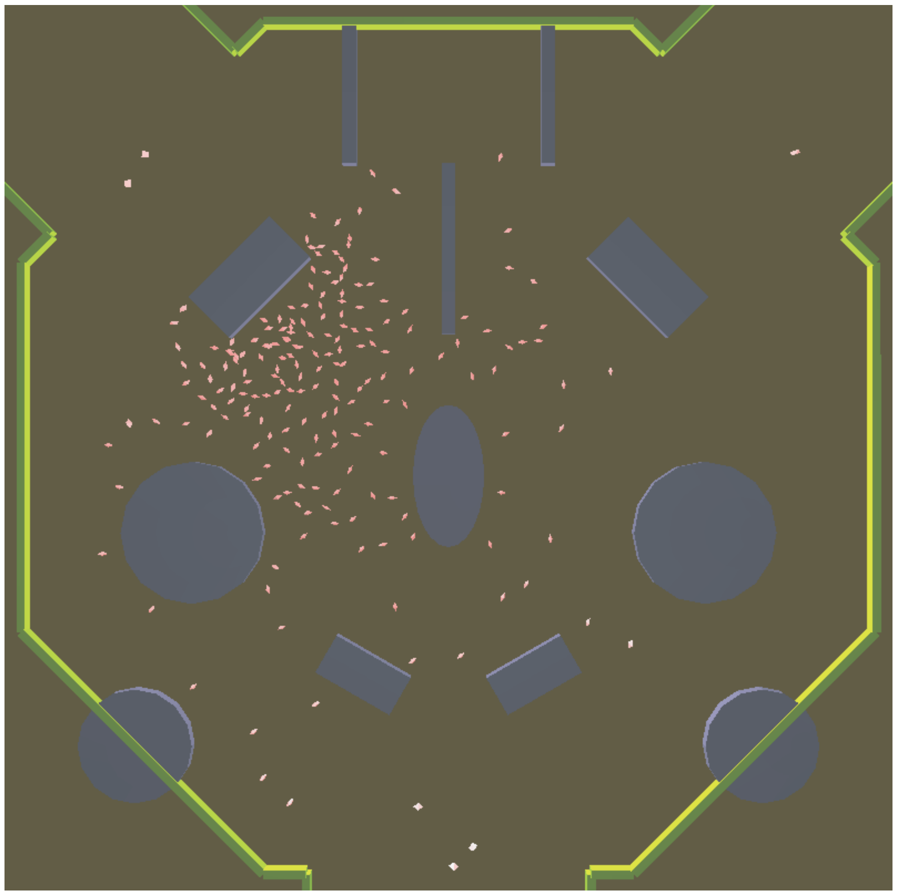


Figure A.1: A simple synthetic environment. The environment can be bounded by a $110 \times 110m^2$ box. In the picture, dark blue objects are obstacles, thin long green objects are walls and red dots are agents.

a while after $x = 241$. Now let us look at F_b and F_c, the two sets of fitness values of permutations starting with Routine B and C, respectively, shown by two red circle in the figure: $F_b = \{y(x)|x \in [121, 240]\}$ and $F_c = \{y(x)|x \in [241, 360]\}$.

It is obvious from the plot that most values in F_b are much smaller than those in F_c. (Actually the mean of F_b is smaller than the minimal value of F_c.) Therefore, we can conclude that permutations $C - *****$ are generally better than $B - *****$. By similar observation, together with further data analysis such as sorting

System fitness F	$=$	$\frac{1}{N} \times \sum_{i=1}^{N} F_i \quad$ (N is the number of agents)
Agent fitness F_i	$=$	$(S_i \times L_i)^4$
Safety factor S_i	$=$	$\begin{cases} (1 - \frac{C_i}{50})^2 & \text{if} \quad C_i < 50 \\ 0 & \text{otherwise} \end{cases}$
Collision frames C_i	$=$	the average number of frames in every 10000 frames that pedestrian i involves in collision with either stationary obstacles or other agents
Liveness factor L_i	$=$	$\begin{cases} 1 & \text{if} \quad R_i > 0.5 \\ (2 \times R_i)^8 & \text{otherwise} \end{cases}$
Speed ratio R_i	$=$	$\frac{\text{average speed of pedestrian } i \text{ in simulation}}{\text{his preferred speed}}$

Table A.2: Definition of fitness.

permutations by partial ordering and comparing shape of different parts of the plot, etc., we found the following fact with a permutation P:

- Those starting with A and B usually give poor performance.
- The later D appears in P, the better the performance will be.
- It's better for C to appear before A, but no need for them to be adjacent.

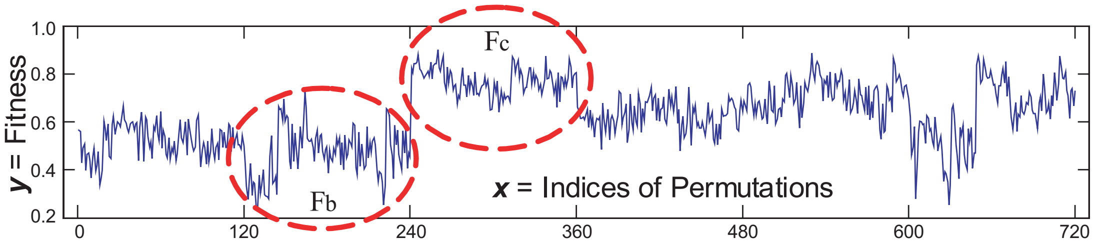


Figure A.2: Plot of fitness values of all permutations.

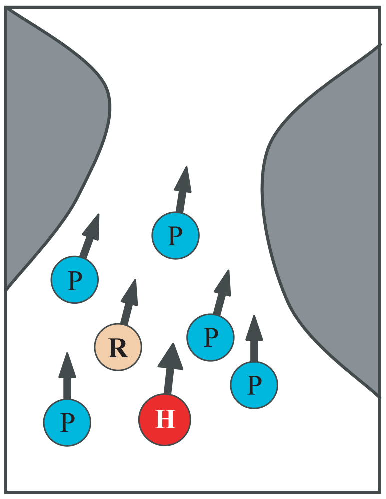


Figure A.3: Explanation of the order of $C - A$: pedestrian H can avoid obstacles by simply following the crowd and letting others (those labeled P) deal with them. So can pedestrian R.

- It's better for A to appear before B, but no need for them to be adjacent.
- If F appears earlier than C, D and E, it has the exact same performance as if F is omitted. (This is obviously true as routine F is designed to correct directions picked by routines C, D or E. If they are not executed, F will have no effect.)

Actually, almost all of the high performance permutations have A before B and after C and have D at the end. It is difficult to fully explain the result of the permutation search. We humbly provide our speculative explanation in the following and hope it will be inspiring to readers.

1. The order of $C - A$. Imagine a crowd, among which there is pedestrian H, moves at a certain direction. Chances are that H, if not on the edge of the crowd, will not bump into a stationary obstacle as long as he stays within the crowd, since the people on the boundary of the crowd may have already dealt with the obstacle, if any, and have steered the crowd to a safe direction (see Figure A.3. So the only thing H need to do is to stay within the crowd by using routine C. The order of $C - A$ allows a pedestrian to take advantage of efforts made by others—"let the others do the obstacle avoidance job and by following the crowd I probably do not have to do it at all".

2. The order of $A - B$. Put B after A is sensible as whenever we want to check a turn, the turn itself should better be already determined. Otherwise, the check is a waste of effort. In the six routines, only A will change the turning angle so big that the turn may need more than one step to finish. Therefore, it is better for A to appear before B.

3. The later appearance of D. D is to avoid the pedestrians coming toward me (either from side or front) that may cause potential collision on my future trajectory. Generally speaking, it considers pedestrians a bit far away and therefore its situations are usually not as urgent as those in other routines. So all of the avoiding options in D are small changes in either moving speed or turning angle or both, which means motor control command issued by the previous routine is likely to be preserved. If D appears early in a permutation, other routines may likely overwrite D's motor control command with their bigger changes. However, if it gets executed at the end, its result will surely remain untouched while motor control command issued by the previous routine is preserved as well (or only slightly altered).

4. "Flexibility" of E and F. If routines A, B, C, and D are perfectly designed, almost all danger situations will be dealt with at their early stages and it is most likely that situations described in E and F will hardly happen. So ideally, E and F do not provide as much help as the other four. On the other hand, even if E and F might help a lot, both of the avoiding options for them involve "slowing down to a stop" should a threat be detected and subsequent reactive behavior routines will hardly have any effect on a pedestrian if he has already stopped. Therefore, given that a threatening situation described by E or F confronts the pedestrian, he will almost always have the same reaction–slow down to a stop–regardless of the order of the reactive behavior routines, which makes E and F fit anywhere. However, in some cases early routines may redirect the pedestrian such that the original threat does not block the way any more but new threats appear. Due to the existence of such occasions, the positions of E and F do affect the performance a bit, but the effect is not strong enough for us to figure out the rule. So they appear "flexible" to us.

Permutation	**333 Agents**	**666 Agents**	**1000 Agents**
CABFED	4	22	84
FCABED	3	25	85
CEABFD	3	23	94
CAFBED	4	24	99
ECABFD	1	31	102

Table A.3: Result performance. Average number of collisions happened in simulations with different number of pedestrians using various best permutations of reactive behaviors.

Before this exhaustive search, we had originally designed, out of our intuition, the order of reactive behavior routines to be simply $A - B - C - D - E - F$, which corresponds to $x = 1$ in the plot in Figure A.2. This choice turned out to have the performance no better than the average. Table A.3 lists some of the best permutations we found together with the number of collisions that happened in several Penn Station simulation experiments (with each being 20 minutes long in virtual time, or 36000 frames) with different number of pedestrians. (In our implementation, we do not impose hard constraint to prevent collisions.) Most of the collisions are human-human and less than 3% are human-obstacle. Collisions usually last no more than 1 second and for the human-obstacle type collisions, obstacle-crossing (e.g., moving "across" a solid wall) never happens.

Despite the satisfactory performance, our approach at the reactive behavior level may not be the only good answer. Alternatively, it might be possible to integrate these reactive behavior routines somehow "in parallel" instead of "sequentially" or one can even come up with another set of key routines.

Appendix B

Additional Motivational Behavior Routines

In this appendix, we describe several representative higher level behavior routines in detail. As mentioned in Chapter 4, these routines depend greatly on other routines that are beneath them in our bottom-up behavioral hierarchy, including navigational behaviors and reactive behaviors, as well as a collection of action level motor skills, such as walking, running, turning while moving, turning at the spot, standing, sitting, etc. In addition, they also rely on specialized environment objects for abstract level interpretation of situation in order to make decisions. Next we detail the routines.

B.1 Surround Artists and Watch Performance

Suppose a pedestrian is attracted by a performance nearby, he will use the routine shown in Table B.1 to approach the performance and watch it until he leaves. In the

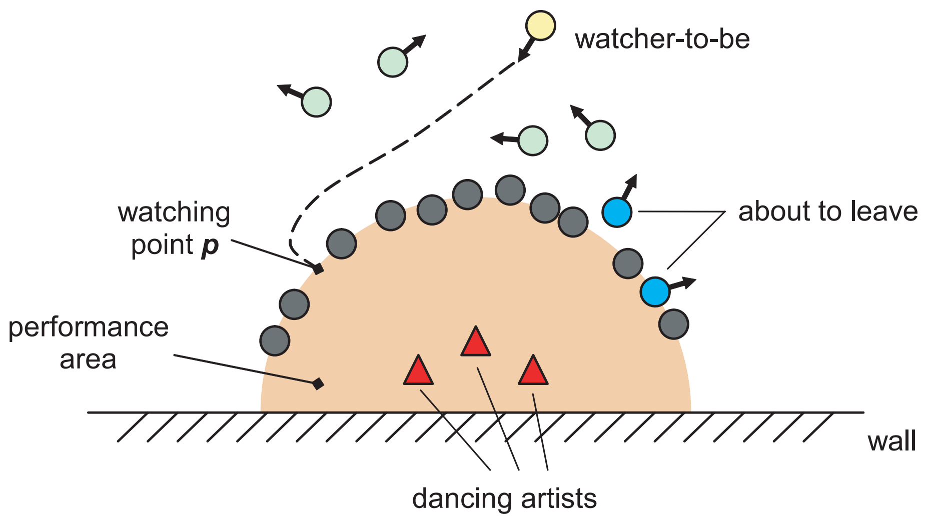


Figure B.1: Approach and watch performance. The yellow pedestrian is interested in the performance. He finds an available spot and approaches it. Among the current watchers, there are two (in blue) on the right who are about to leave. And outside the area, pedestrians (in green) are constantly passing by.

routine, A is the performance area defined as a (part of) circular area surrounding the performing artists, illustrated in Figure B.1.

In Step 2, a pedestrian will use detailed path-planning to find an available watching point and plan a path to it simultaneously. He first puts the performance area A as a target on a grid path map and then add every watcher surrounding the area as a static circular obstacle, which can effectively prevent path search to access the target area through its standing point. This can lead path search toward the nearest available interval space around the area, if one exists. In cases when no space is available, the pedestrian can either give up or enlarge the target performance area by the size of pedestrian bounding box and try finding a path again. This probabilistic strategy leads to a sparse second layer of watchers.

In order to quickly identify all the current watchers, a pedestrian needs the help

1. If A is still far away, use navigation behaviors to approach A.
2. Once A is close enough, find an available watching point p around A.
3. Use detailed arrival behavior to approach and reach p.
4. Turn to face the performance.
5. Watch the performance for a while.
6. Turn to face the outside.
7. Leave the performance.

Table B.1: Surround artists and watch performance

from a specialized object—the performance area object. This object keeps track of the watchers that surrounding the area. Whenever a new watcher joins, it will be registered into the watcher list. Once it leaves, it will be removed. These operations are triggered by the watchers themselves. If a watcher-to-be is approaching and is close to its watching position (say one meter away), he will request a registration. This also in a way resolves conflict of two watchers-to-be competing a spot big enough for only one. The first one that issues the registration request will get the spot. It is fair enough as the first issuer is usually also the closer one. When he leaves the performance area, a watcher will issue a remove request once he is far away enough (say more than one meter away). Specifically in the routine, the registration request is usually issued late in Step 3 and the removing request late in Step 7.

B.2 Make a Purchase

When a pedestrian need to get something T, say a ticket or a drink, through a purchase, the routine "make a purchase" shown in Table B.2 will be employed.

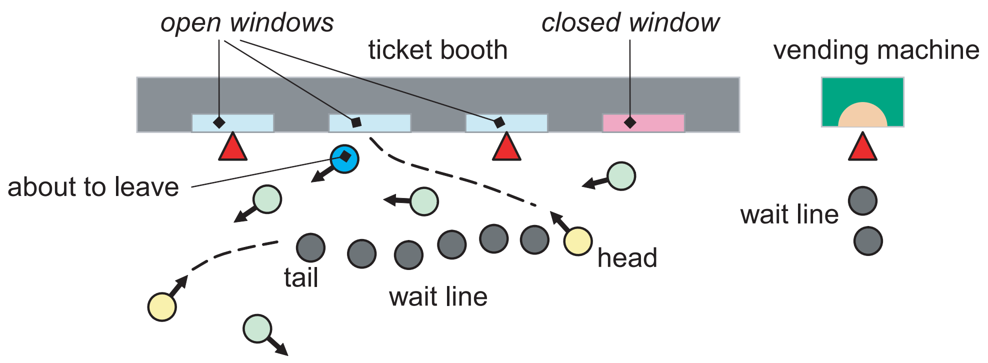


Figure B.2: Get on the line and make a purchase. A yellow pedestrian on the left is going to wait on the line for tickets. The other yellow pedestrian in the front of this line, is about to take the transaction spot just available as the blue pedestrian leaves. On the right side, a wait line also forms for purchasing drink from a vending machine. Red triangles are pedestrians who are currently making purchases. With other pedestrians (in green) constantly passing by, the situation is difficult to analyze without the help of specialized objects.

```
1.    Find out all places within the current region that sell $T$.
2.    Pick the best one $B$ among them in terms of proximity and expected
      wait time.
3.    If $B$ is far
4.       use navigation behavior to approach it.
5.    else if there is a spot available in B for transaction
6.       approach and take it.
7.    else
7.1      Go and stay behind the last person on the waiting line.
7.2      Wait either patiently or impatiently.
7.3      If the line moves forward, follow it to move forward.
7.4      If I become the first on the line and a spot for transaction is available
7.5         then approach and take the spot.
8.    Make a purchase at the transaction spot.
9.    Leave the transaction spot.
```

Table B.2: Make a purchase

In Step 2, a routine similar to passageway selection (see Table 4.2) will be used for the pedestrian to make a choice among several available purchasing places. In Step 7.2, one of several waiting motions of different styles in the motion repertoire will be picked in a probabilistic way to express the (im)patience of the pedestrian.

Like the routine in last section, "make a purchase" requires two types of specialized objects to help it analyze the situation. The *wait-line* object which keeps track of the waiting pedestrians on the line will tell a pedestrian how many people are on the line, who is the first and who is the last. The *purchase-point* object can point out whether a transaction spot is available for taking. And similarly, pedestrians need to issue requests to register themselves into and remove themselves from those specialized objects. Given that there are other pedestrians constantly passing by (as shown in Figure B.2), the help of these specialized objects are crucial to situation analysis and decision making.

B.3 Take a Rest

The last example (see Figure B.3) is the routine that enables the pedestrian to take a rest as shown in Table B.3. The basic structure is pretty much the same as the previous two. The selecting behavior here uses the resting comfort in addition to proximity as the criteria. And as usual, there is a specialized object—*seat-space*, which tracks all the available spaces on a resting facility—that helps the execution of this behavior routine.

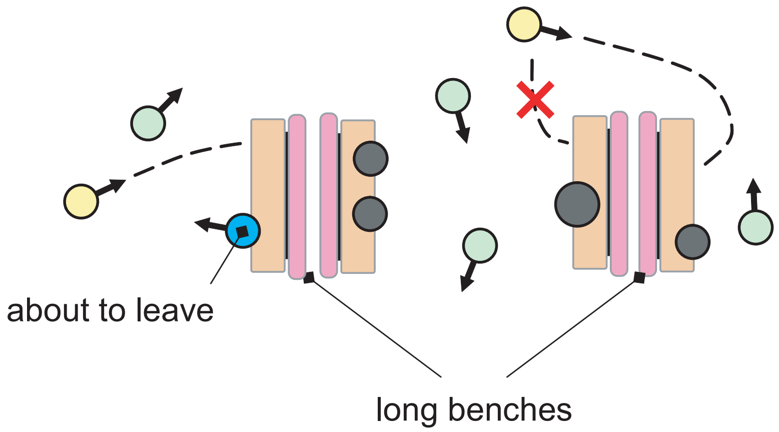


Figure B.3: Pick a comfortable seat and take a rest. There are four long benches in this figure. Characters in dark gray are currently sitting on the benches. A blue character is about to leave the left-most bench, on which a yellow character is going to sit. On the upper right side, another yellow pedestrian also want to take a rest. With two choices close to him, he picks the more comfortable one (in the sense of more space) and proceed to it. Other pedestrians that pass by are shown in green.

1. Find all seats around within the current region.
2. Pick the best available one B among them in terms of proximity and expected resting comfort.
3. If B is far, use navigation behavior to approach it.
4. Once B is close, plan a detailed-path and use detailed arrival behavior to approach and reach it.
5. When in front of the seat, turn to face the correct direction and sit down.
6. Sit for a while.
7. Stand up and leave the seat.

Table B.3: Pick a comfortable seat and take a rest

B.4 Summary

To summarize, the routines described in this appendix represent a big category of meaningful behaviors suitable for pedestrians. Such behaviors involve interpersonal conflicts among pedestrians on available resources [Mataric 1994]. While everyone is trying to maximize its own benefit through various selecting behaviors in resolving these conflicts, they shall at the same time obey rules that are explicitly or implicitly defined for the sake of others in the society. It is exactly the balancing of these two sides that makes their ultimate behaviors appear to be natural and rational.

Appendix C

Motion Analysis and Synthesis

The human animation package *DI-Guy* we have been using provides a variety of colored and textured human models. It frees us from the tedious work of creating different character models. But at the same time, unfortunately, the software restrains us to use only the motion repertoire and blending algorithms that come with the package, allowing only limited flexibility in customization. As the software was originally designed for users to make animation out of scripts, it suffers from limited amount of motion data and non-optimal quality of synthesis algorithms when used to produce realtime animation. In the this appendix, we provide a motion analysis and synthesis framework we designed as an alternative to our current motor level implementation for interested readers.

To get high fidelity behavioral animation, we choose to use a data-driven approach to synthesize human motions, especially pedestrian motions. To prepare a motion database, our system first analyzes a given motion library and transforms the data into a composite network structure for the sake of search efficiency. In the synthesis phase, heuristic graph traverse is used to find, in this network structure, motion sequences with certain properties. After a set of modification these sequences are pieced together to create the target motion. Now we describe our method in detail.

C.1 Find Constraints

As we focus on modeling human pedestrians, locomotion is the type of motion that interests us most. The most important constraint for locomotion is foot-plant. In order to save time and effort, we incorporate a simple but effective method to automatically detect foot-plant constraints in motions as follows:

1. Both feet are labeled as unconstrained for every frame.

2. Compute the vertical position p_y and velocity magnitude $|v|$ of two reference points (one attached on heel and the other on toe) for each foot at each frame. As we assume all motion data are recorded on a flat ground, if p_y and $|v|$ are below their respective threshold T_{p_y} $(= 10cm)$ and T_v $(= 1.25cm/frame)$ for reference point P of foot F at frame i, then we will label F as to be planted at P for frame i.

3. Compute a_{com}, the vertical acceleration of the body **COM** (center of mass), for those frames that are still unlabeled (i.e., neither foot is planted so far). Theoretically, if a_{com} is bigger than gravity constant g (assuming g takes the minus sign), there must be some upward supporting force underneath the body, which infers the existence of planted points. In practice, we choose

$0.33g$ as a clamp value (see $G_{support}$ in Table C.2) and run Step 2 again with new thresholds ($T_{p_y} = 10cm$, $T_v = 2.5cm/frame$) on those frames whose a_{com} are bigger than $0.33g$. The choice of $0.33g$ makes sure that this process will not mistake an unsupported case as supported, although it may fail to detect a right one.

4. Label all the unconstrained frames as "neither", which means neither foot is planted. Thus every frame has some constraint.

5. To maintain certain level of continuity, abrupt constraint changes are removed: for a constraint that lasts a tiny period of time, say less than F_c frames (currently set to 2), we replace the first half of the frames with their predecessor's constraints and the second half their successor's. This process is iterated until no change is needed.

In the above method, finite difference is used for velocity/acceleration computation. In order to compute the body **COM** position, we attach a set of reference points (or "point cloud") to different skeletal parts of our character model. These points are properly weighted and distributed all over the body (see figure C.1) according to an anthropometry study by Naval Biodynamics Laboratory [Naval Biodynamics Laboratory 1988]. We utilize the cloud of points to effectively approximate the mass distribution of a character body at different postures. In this way, body **COM** position is simply the weighted spatial average of these points.

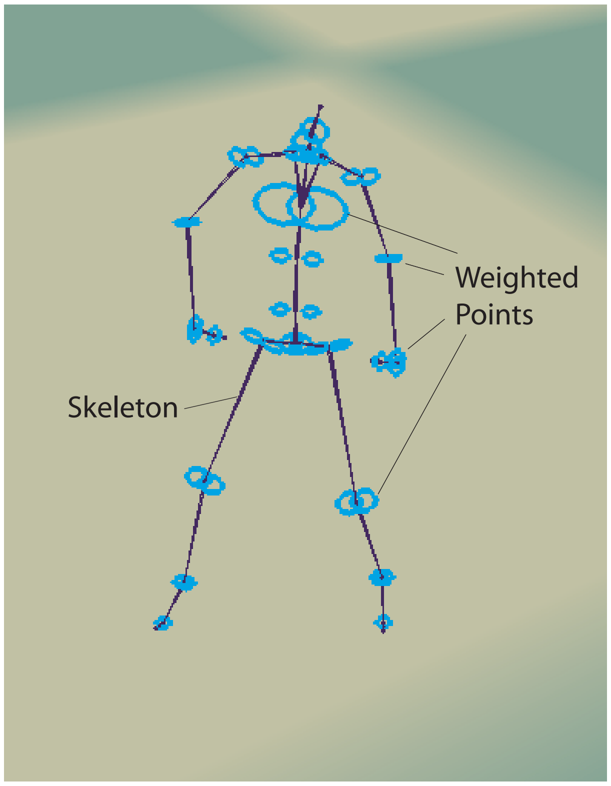


Figure C.1: Weighted point cloud on a skeleton. Center of each light blue circle is the position of a point and the size of each circle indicates the weight of that point.

C.2 Compute Similarity Values

In order to determine how similar two motion clips are, we compute the aligned difference between postures of the two motions. Here we use the point cloud model again. As mentioned before, during any motion the cloud of points will move with their attached body parts, which approximately gives the body weight distribution for different postures. With this, we define our similarity function D as the sum of weighted distance between aligned point clouds of frame segments from two motion clips:

$$D(A,i,B,j) = \sum_{t=0}^{L_f-1} \sum_{k=0}^{N_p-1} w_k \times D_p(A, i+t, B, j+t, k) \tag{C.1}$$

where

1. A and B are motion clips;

2. i and j are frame numbers;

3. L_f is a predefined constant indicating the length of frame segment used for similarity computation;

4. $D_p(A,i,B,j,k)$ is the distance between the two "kth points" in frame $A(i)$ and $B(j)$ respectively;

5. N_p is the number of points in the predefined point cloud;

6. w_k is the predefined weight associated to the "kth point" in the point cloud; and

7. (A,i,B,j,L_f) must be compatible, whose meaning will be explained later; Otherwise $D(A,i,B,j) = \inf$.

It is clear from the definition of the similarity function that smaller values indicate the two motions are more similar to each other. As smooth blends require more information than can be obtained at individual frames, we use a segment of L_f continuous frames to effectively include the important kinematic information (such as velocity, acceleration and higher order derivatives) of body parts into account. To remove the difference of global transformation (rotation around the vertical axis and translation), we align up two sequences of point clouds based on

Pairs of constraints	(L, L)	(R, R)	(B, B)	(N, N)	(L, B) or (B, L)	(R, B) or (B, R)	Others
Compatible	Yes	Yes	Yes	Yes	Yes	Yes	No
New constraint	L	R	B	N	L	R	N/A

Table C.1: Compatible pairs of constraints. Foot constraints abbreviation: L – left, R – right, B – both, N – neither.

1) the body **COM** (center of mass) positions, and 2) the root joint facing directions of the two first frames. This alignment process will be used again later in both transition construction and motion synthesis.

In equation C.1, we require (A, i, B, j, L_f) to be compatible. This means that each correspondent frame pair of segments $A(i)$, $A(i+1)$, ..., $A(i+L_f-1)$ and $B(j)$, $B(j+1)$, ..., $B(j+L_f-1)$ must have compatible constraints, which we define in Table C.1. This requirement avoids constraint ambiguity in transition sequences to be generated and thus improves transition continuity.

C.3 Pick Transition Points

Using the similarity metric described above, we can compute a similarity matrix for two motion clips A and B with each entry M_{ij} as the value of $D(A, i, B, j)$. Intuitively, in this matrix, a value below a given threshold indicates a high quality transition point and a local minimum implies a best transition point among the neighbors. However, different kinds of motions have different kinematic properties and different transition fidelity requirements, and therefore need different threshold values. We believe that the best way to find the threshold is to look back into the original motion clips. By comparing two neighboring frame segments (say frame $1 \sim 10$ and frame $2 \sim 11$) from the same motion, we can get a quantitative concept of the inherent similarity between different postures of this motion, which we call the "similarity reference" value R. As this value may still vary along a motion clip, we put the comparison down to the frame level and it can be computed by plugging in the same motion for A and B to equation C.1. Thus R can be written as

$$R(A,i) = D(A,i,A,i+F_{offset}) \tag{C.2}$$

where F_{offset} is a predefined constant indicating offset frames between segments for computation of similarity reference value. Now that every motion frame (except those at the end of a motion clip) is assigned a similarity reference value, we can compare every original similarity value $D(A,i,B,j)$ with its two correspondent reference values:

$$D_{adj}(A,i,B,j) = \frac{D(A,i,B,j)}{min(R(A,i),R(B,j))} \tag{C.3}$$

where $min(x,y)$ is the minimum function. The result of this comparison D_{adj} is a uniform measurement that reveals how similar the two frame segments are with respect to their original motions. Now we convert every matrix of D into a new matrix of D_{adj} and pick local minimal values below a constant threshold T_q as transition points. This threshold T_q is a user-defined constant which controls the overall quality of all transition clips. In Table C.2, we list all the constants we

Constant Name	**Type**	**Current Value**	**Function Description**	**See Section**
T_{p_y}	threshold	$10cm$	maximal vertical distance (due to noise) allowed between ground and a planted foot	C.1
T_v	threshold	1.25 $cm/frame$	maximal speed allowed for a planted foot	C.1
$G_{support}$	threshold	$0.33g$, (g takes minus sign)	minimal acceleration of body **COM** indicating existence of supporting force	C.1
F_c	threshold	2 frames	minimal period of time that a constraint shall last	C.1
L_f	constant	10 frames	the length of frame segment used for similarity computation	C.2
N_p	constant	45 points	the number of points in the predefined point cloud	C.2
w_k $(k = 1, 2, ..., N_p)$	constant	various according to k	weight associated to kth point in the point cloud	C.2
F_{offset}	constant	1 frame	the offset frames between segments for computation of similarity reference value	C.3
T_q	threshold	2.5	threshold for quality control of transition clips	C.3

Table C.2: Constant list for motion analysis and synthesis

mentioned in this chapter and their values in our current system, which are picked based on experiments.

C.4 Construct Transitions

For each transition point found in Section C.3, transition motion will be constructed via a blending process. Suppose $D(A, i, B, j)$ is the similarity value for a transition point. First of all, frame segments $A(i) \sim A(i + L_f - 1)$ and

$B(j) \sim B(j + L_f - 1)$ need to be transformed so that the first frames are aligned together in the sense that 1) body **COM** (center of mass) shall be at the same position and 2) root joint orientation (around vertical axis only) shall be the same. Then we linearly interpolate between the two aligned frame segments to compute each frame p $(0 \leq p < L_f)$ of the new transition C:

$$C(p) = \alpha(\frac{p}{L_f}) \times A(i+p) + (1 - \alpha(\frac{p}{L_f})) \times B(j+p) \tag{C.4}$$

where $\alpha(x)$ is a blending weight function defined as:

$$\alpha(x) = \begin{cases} 1, & x < 0 \\ 2x^3 - 3x^2 + 1, & 0 \leq x \leq 1 \\ 0, & x > 1 \end{cases} \tag{C.5}$$

As this weight function has C^1 continuity everywhere and satisfies $\alpha(0) = 1$, $\alpha(1) = 0$, and $\alpha'(0) = \alpha'(1) = 0$, motion continuity is maintained at both boundaries as well as along the transition motion. Constraint continuity is implicitly guaranteed since those pairs of segments with incompatible constraints have been already discarded during similarity value computation, as described early. In Table C.1, we enumerate the choice of the new constraint out of a compatible pair for new transition frames.

C.5 Extract Footstep Information

Up to this stage, we have transformed a library of unrelated motion clips into a directed network, as shown in figure C.2. Paths in this network system reveal the possible sequences of motions that can be synthesized. However, it does not contain any explicit information for searching, such as those that describe motion properties. In addition, the network is composed of a great number of frames which form a space that is too big for online searching. To solve these problems, we construct an abstract network above the current one and embed information in the higher level network to facilitate online searching.

To make the abstraction, we segment the current network into sequences of frames, each of which is an "atomic action". Currently as we are focusing mostly on pedestrian's locomotion, it is natural and intuitive to take a "footstep" as an atomic action. Here footstep is a general concept. Using foot-plant constraint information of each frame, we define footstep as follows:

> A footstep is a sequence of frames whose constraint sequence is of the form $(a, a, \ldots, a, b, b, \ldots, b, c)$, where a, b and c are constraints satisfying
>
> 1. a can be "left", "right" and "both";
> 2. b sequence is an optional "neither" sequence; and
> 3. c can be "left", "right" and "both" but must be different from a if no b appears in between.

According to this definition, constraint sequences (left, left, left, both) and (right, right, right, neither, neither, neither, right) are legal footsteps but sequences (left, left, neither) and (neither, neither, right) are not. Intuitively speaking, footsteps are always bounded by frames that start a new foot-plant constraint (except the "neither" constraint). Following this intuition, it is easy to pick out all the

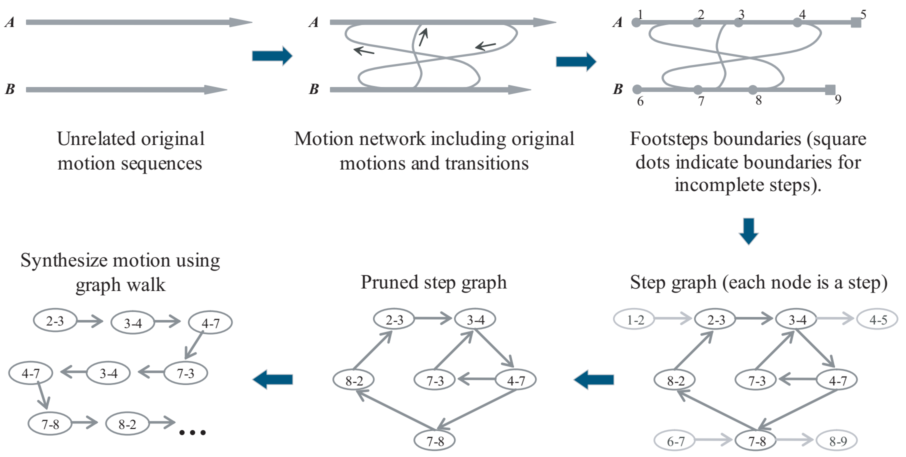

A
B
Unrelated original motion sequences
Motion network including original motions and transitions
1
2
3
4
5
6
7
8
9
Footsteps boundaries (square dots indicate boundaries for incomplete steps).
Step graph (each node is a step)
1-2
2-3
3-4
4-5
8-2
7-3
4-7
6-7
7-8
8-9
Pruned step graph
Synthesize motion using graph walk
...

Figure C.2: From motion data analysis to synthesis.

boundary frames from the current motion network and then each frame sequence between two neighboring boundary frames (inclusive) is a footstep. Note that boundary frames are always used by two neighboring footsteps, which makes it natural for them to act as the reference frame for aligning continuous sequences together in the synthesis stage. Incomplete footsteps may appear at the end of motion sequences and they will be discarded in the pruning stage, which will be discussed soon. After the above segmentation, we can construct a new graph (called *step graph*) with each node encapsulating a footstep and each edge correspondent to the adjacency between footsteps, as shown in figure C.2. If we assume each footstep consists of about 10 frames, then clearly, the size of the step graph is one magnitude smaller than the original motion network. In addition, we analyze each footstep and store in its step graph node motion information such as moving speed, direction change, jumping distance and so on. Such helpful information will further improve the performance of search during online motion synthesis.

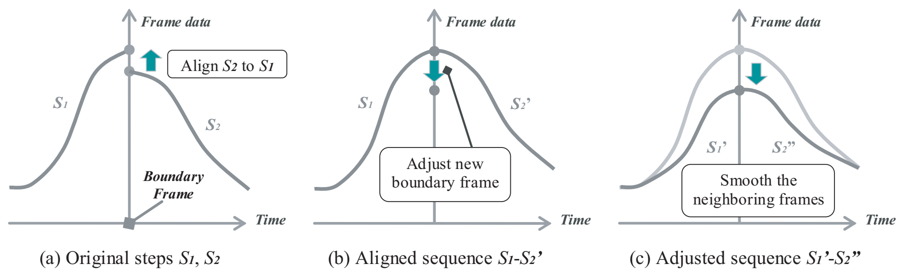

Frame data
Align S_2 to S_1
S_1
S_2
Boundary Frame
Time
(a) Original steps S_1, S_2
Frame data
S_1
S_2'
Adjust new boundary frame
Time
(b) Aligned sequence S_1-S_2'
Frame data
S_1'
S_2''
Smooth the neighboring frames
Time
(c) Adjusted sequence S_1'-S_2''

Figure C.3: Alignment and adjustment.

C.6 Prune the Step Graph

The step graph is likely to have nodes that are not on any cycle (for instance, those incomplete footstep nodes mentioned before). While we want to synthesize motion continuously and indefinitely, these nodes will bring motion synthesis to an end. Other nodes may be part of one or more cycles but nonetheless only be able to reach a small fraction of all nodes in the graph. They will cause motion synthesis confined to a small part of the database. To address the problem, we find in the step graph the largest strongly connected component and eliminate all the remaining part of the step graph. This solution guarantees non-stop motion synthesis at the cost of motion data loss.

C.7 Synthesize Motion

Up to now, we have converted a collection of unrelated motion clips into a two-level composition of directed graph together with descriptive information for each atomic action. To synthesize motions, we do the following two steps.

1. Use a goal-directed traverse in the step graph level to find a path with certain properties. As we are doing online synthesis, the traverse is heuristic in the sense that breadth first search is used and only a limited number of nodes are visited before making the final decision. This assures that search time is bounded by a strict upper limit. To decide whether a path in step graph is correspondent to a motion sequence that satisfies certain properties, such as "going straight forward", "getting closer to a target" or "turning at point p to face east", the character's current global transformation and posture configuration are used as initial condition. During the traverse, motion information stored in each step graph node being visited is retrieved and combined with values from previously-visited nodes for computation of objective functions. If a candidate satisfies the goal, the traverse stops and returns the candidate. Otherwise, a set of best candidates found so far is kept until the time limit comes. At that moment, a random one from the set will be picked as the answer.

2. The path found in the step graph has several nodes, each of which correspondent to a sequence of frames in the motion graph level. By connecting these sequences together with proper alignment and adjustment respectively at action and frame level, a new motion is ready. The action level alignment takes the last frame (which is also a footstep boundary frame) of the previous atomic action as the reference and transforms the current action rigidly as a whole to merge its first frame with that reference frame. Adjustments happen at frame level and they usually alter character postures (in the case of foot-plant constraint enforcement, for instance), causing changes inside a frame. In the cases that footstep boundary frames are modified, we need a special adjustment to propagate, with decreasing weights, the modification part toward both directions: forward to all frames after the boundary and backward to all before it, as show in part (c) of figure C.3. For online animation, such adjustment requires us to look ahead at least two footsteps. Once the new motion sequence is ready, the algorithm goes back to step 1 again to look for solutions with new initial condition.

C.8 Implementation and Discussion

We have implemented the above algorithms in a stand-alone *Maya* plug-in program using *Visual C++*. This program can read in a collection of motion sequences, analyze them, convert the collection to a two-level composite network structure, and generate in real-time continuous motions with desired properties (see Figure C.4 and C.5). Currently, we restrict our objective motion properties to be spatial ones such as "desired target point", "desired target orientation", etc. The system can be extended to accommodate other properties such as those in the temporal or posture domains. As we mentioned before, for the time being, we have not linked this motion synthesis mechanism with our higher level pedestrian behavioral and cognitive model, due to the interface limitation of *DI-Guy*, which provides us the textured human models. However, the algorithms we provided in this appendix can be used as a motion synthesis alternative once a better, more flexible human animation software is available.

Motion synthesis using captured real data described here is not exactly an artificial life approach. The ideal way is to use motion controllers in physical

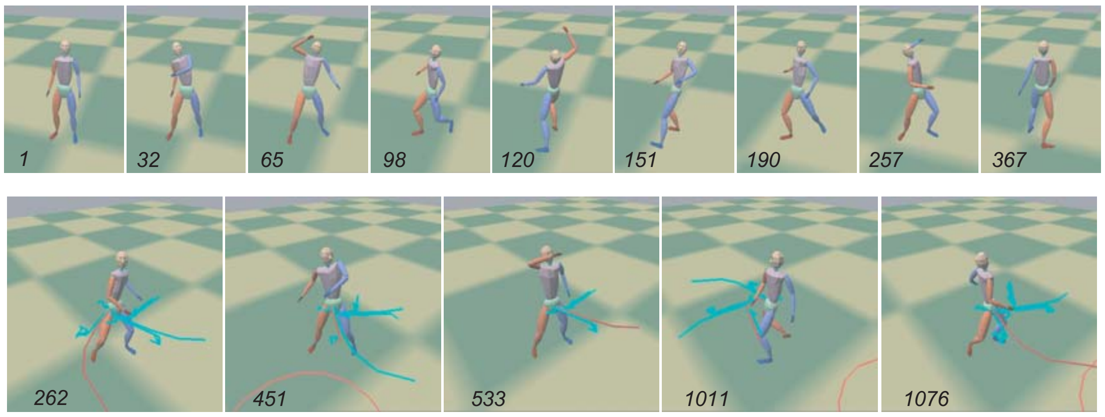


Figure C.4: Snapshots of online motion synthesis, part 1. Images in the first row are snapshots of an animation of the original motion capture data (12 seconds in length) in which the character is trying to drive away some bees. Images in the second row show an animation with motions synthesized in realtime. Here the character is asked to reach several targets marked by red circles (partly shown in the bottom of the 2nd, 4th, and 5th image). As the available motion data are limited, the character needs to plan several steps ahead and searches for best answers. The blue curves coming out from the pelvis of the character are part of the search tree at every step and the red curve is the branch he picked. Each snapshot has its frame number shown on its lower-left corner.

simulation to drive different parts of the body to accomplish various motor skills, as presented in [Faloutsos et al. 2001b]. However, not only are the controllers difficult to design and thus only a limited amount of motion data can be produced so far, but also the frame rate of such simulation is quite low even for a single character. For now, obviously it is not suitable for realtime simulation with hundreds even thousands of pedestrians.

Recent advance in hybrid techniques (see Chapter 2, Section 2.1) opens up new possible ways of motion synthesis with flexibility in highly-interactive situations. And such techniques may well be suitable for animating autonomous creatures in future.

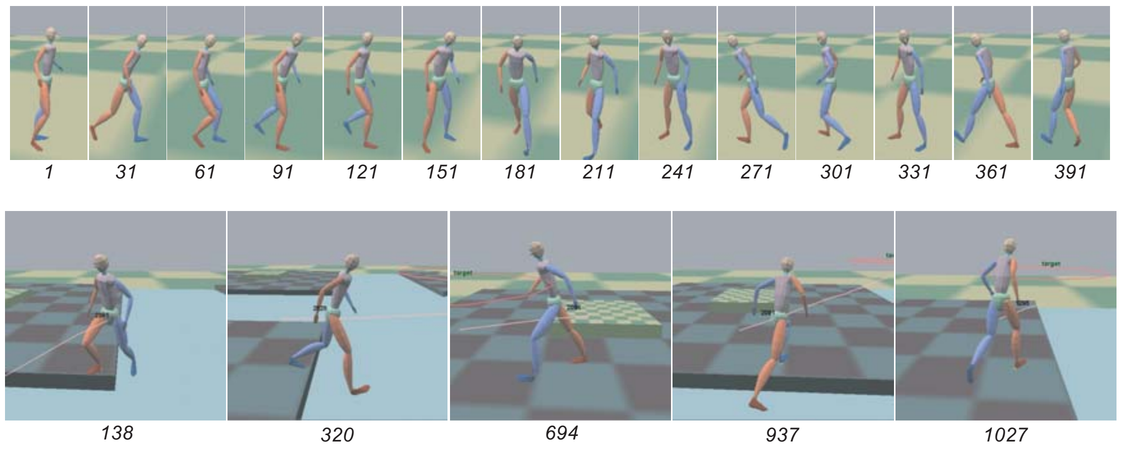


Figure C.5: Snapshots of online motion synthesis, part 2. The first row shows snapshots of the original motion capture data (14 seconds in length). In this data set, the character walks nervously on a flat ground, frequently stops and looks around, seemingly looking for something. The second row of snapshots shows an animation in which the character is asked to reach several targets on an uneven terrain. Again motions are synthesized in realtime using only the available data. In each image, the number near the character's pelvis is the number of possible choices of the next several steps ahead. The pink line coming out from the character's lower body is the route he chose. Note that although original motion is on even ground, our synthesis process can smoothly adjust (see the second step in Section C.7) the motion for uneven terrains and still maintain the natural look. Again, each snapshot has its frame number shown beneath the image.

Bibliography

ALIAS SYSTEMS CORP., 2005. Maya. http://www.alias.com.

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 483–490.

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2005. Pushing people around. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 59–66.

ASHIDA, K., LEE, S., ALLBECK, J., SUN, H., BADLER, N., AND METAXAS, D. 2001. Pedestrians: Creating agent behaviors through statistical analysis of observation data. In *Proc. Computer Animation, Seoul, Korea. IEEE Computer Society.*, 84–92.

AUTODESK 3DS MAX, 2005. Character studio. http://www.discreet.com.

BADLER, N., PHILLIPS, C., AND WEBBER, B. 1993. *Simulating Humans: Computer Graphics, Animation, and Control.* Oxford University Press.

BADLER, N., ALLBECK, J., ZHAO, L., AND BYUN, M. 2002. Representing and parameterizing agent behaviors. In *CA '02: Proceedings of the Computer Animation*, IEEE Computer Society, Washington, DC, USA, 133.

BIOGRAPHIC TECHNOLOGIES, INC., 2005. Ai. implant for animation. http://www.biographictech.com.

BLUE, V., AND ADLER, J. 1998. Emergent fundamental pedestrian flows from cellular automata microsimulation. *Transportation Research Record 1644*, 29–36.

Blue, V., and Adler, J. 2000. Cellular automata model of emergent collective bi-directional pedestrian dynamics. In *Artificial Life VII: Proc. of the Seventh International Conference on Artifical Life*, 437–445.

Boston Dynamics, Inc., 2004. Human simulation. http://www.bdi.com.

Botea, A., Müller, M., and Schaeffer, J. 2004. Near optimal hierarchical path-finding. *Journal of Game Development 1(1)*, 7–28.

Brogan, D. C., and Hodgins, J. K. 1997. Group behaviors for systems with significant dynamics. *Auton. Robots 4*, 1, 137–153.

Brooks, R. A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation RA-2(1)*, 14–23.

Brooks, R. A. 1991. Intelligence without reason. Tech. rep., Massachusetts Institute of Technology, Cambridge, MA, USA.

Brooks, R. A. 1995. Intelligence without representation. In *Computation & intelligence: collected readings*, American Association for Artificial Intelligence, Menlo Park, CA, USA, 343–362.

Bruderlin, A., and Williams, L. 1995. Motion signal processing. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 97–104.

Buchsbaum, D., and Blumberg, B. 2005. Imitation as a first step to social learning in synthetic characters: a graph-based approach. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 9–18.

Chi, D., Costa, M., Zhao, L., and Badler, N. 2000. The emote model for effort and shape. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 173–182.

DeSouza, G. N., and Kak, A. C. 2002. Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence. 24*, 2, 237–267.

Eibl-Eibesfeldt, I. 1975. *Ethology: The Biology of Behavior. 2nd Edition.* Holt, Rinehart & Winston., New York.

Elfes, A. 1987. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation 3(3)* (June), 249–265.

Faloutsos, P., van de Panne, M., and Terzopoulos, D. 2001. The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills. *Computers and Graphics 25(6)* (December), 933–953.

Faloutsos, P., van de Panne, M., and Terzopoulos, D. 2001. Composable controllers for physics-based character animation. In *Proceedings of SIGGRAPH 2001*, 251–260.

Farenc, N., Boulic, R., and Thalmann, D. 1999. An informed environment dedicated to the simulation of virtual humans in urban context. *Computer Graphics Forum 18(3)* (September), 309–318.

Funge, J., Tu, X., and Terzopoulos, D. 1999. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. In *Proceedings of SIGGRAPH 99*, 29–38.

Gipps, G., and Marksjo, B. 1985. A micro-simulation model for pedestriain flows. *Mathemathics and Computers in Simulation 27*, 95–105.

Gleicher, M. 1998. Retargetting motion to new characters. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 33–42.

Gleicher, M. 2001. Motion path editing. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, ACM Press, New York, NY, USA, 195–202.

Goffman, E. 1971. *Relations in public: microstudies of the public order*. Basic Books, Inc., New York.

Guivant, J., Nebot, E., Nieto, J., and Masson, F. 2004. Navigation and Mapping in Large Unstructured Environments. *The International Journal of Robotics Research 23*, 4-5, 449–472.

Helbing, D., and Molnar, P. 1995. Social force model for pedestrian dynamics. *Physical Review 51(5)*, 4282–4286.

Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. 1995. Animating human athletics. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 71–78.

Joukowsky, M. S. 1998. *Petra Great Temple Volume I: Brown Univeristy Excavations 1993-1997.* E. A. Johnson Company, Providence, Rhode Island.

Koechling, J., Crane, A., and Raibert, M. 1998. Applications of realistic human entities using di-guy. In *Proc. of Spring Simulation Interoperability Workshop. Orlando, Fl.*

Kovar, L., Gleicher, M., and Pighin, F. 2002. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 473–482.

Kovar, L., Schreiner, J., and Gleicher, M. 2002. Footskate cleanup for motion capture editing. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 97–104.

Kuipers, B., and Byun, Y. T. 1991. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems 8*, 47–63.

Kwon, T., and Shin, S. Y. 2005. Motion modeling for on-line locomotion synthesis. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 29–38.

Lamarche, F., and Donikian, S. 2004. Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Comput. Graph. Forum 23*, 3, 509–518.

LeCun, Y., Muller, U., Ben, J., Cosatto, E., and Flepp, B. 2005. Off-road obstacle avoidance through end-to-end learning. In *Advances in Neural Information Processing Systems (NIPS 2005)*, MIT Press.

Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., and Pollard, N. S. 2002. Interactive control of avatars animated with human motion data. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 491–500.

Leonard, J., and Durrant-Whyte, H. 1991. Mobile robot localization by tracking geometric beacons. *IEEE Transaction on Robotics and Automation 7*, 376–382.

Lorenz, K. 1981. *The Foundations of Ethology.* Springer-Verlag New York, Inc., New York.

Loscos, C., Marchal, D., and Meyer, A. 2003. Intuitive crowd behaviour in dense urban environments using local laws. In *Theory and Practice of Computer Graphics*, IEEE Computer Society, 122–129.

Lovas, G. G. 1993. Modeling and simulation of pedestrian traffic flow. In *Modeling and Simulation: Proceedings of 1993 European Simulation Multiconference.*

Loyall, A. B., Reilly, W. S. N., Bates, J., and Weyhrauch, P. 2004. System for authoring highly interactive, personality-rich interactive characters. In *SCA '04: Proc. of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, 59–68.

Maes, P., Darrell, T., Blumberg, B., and Pentland, A. 1995. The alive system: full-body interaction with autonomous agents. In *CA '95: Proceedings of the Computer Animation*, IEEE Computer Society, Washington, DC, USA, 11.

Maes, P., Ed. 1991. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back.* The MIT Press, Cambridge, MA.

Maletic, V. 1987. *Body, Space, Expression: The Development of Rudolf Labans Movement and Dance Concepts.* Mouton de Gruyte.

Massive Software, Inc., 2005. 3d animation system for crowd-related visual effects. http://www.massivesoftware.com.

Mataric, M. J. 1994. *Interaction and Intelligent Behavior.* Ph.D. Dissertation. Department of EECS, MIT, Cambridge, MA.

Moravec, H. P. 1988. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 61–74.

MultiGen-Paradigm, Inc., 2005. Openflight file format. http://www.multigen.com.

Musse, S., and Thalmann, D. 2001. Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics 7(2)*, 152–164.

Natural Motion, Ltd, 2003. Active character technology (a.c.t.). http://www.naturalmotion.com.

Naval Biodynamics Laboratory. 1988. *Anthropometry and Mass Distribution for Human Analogues, Volume I. Military Male Aviators.* Project Report, March.

Noser, H., Renault, O., Thalmann, D., and Magnenat-Thalmann, N. 1995. Navigation for digital actors based on synthetic vision, memory and learning. *Computers and Graphics 19(1).*

Okazaki, S. 1979. A study of pedestrian movement in architectural space, part 1: Pedestrian movement by the application on of magnetic models. *Transactions of Architectural Institute of Japan 283*, 111–119.

Perlin, K., and Goldberg, A. 1996. Improv: a system for scripting interactive actors in virtual worlds. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 205–216.

Pollard, N. S., and Zordan, V. B. 2005. Physically based grasping control from example. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 311–318.

Prasso, L., Buhler, J., and Gibbs, J. 1998. The pdi crowd system for antz. In *SIGGRAPH '98: ACM SIGGRAPH 98 Conference abstracts and applications*, ACM Press, New York, NY, USA, 313.

Pullen, K., and Bregler, C. 2002. Motion capture assisted animation: texturing and synthesis. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 501–508.

Qureshi, F., and Terzopoulos, D. 2005. Surveillance camera scheduling: A virtual vision approach. In *Proc. Third ACM International Workshop on Video Surveillance and Sensor Networks (VSSN 05)*, ??–??

Qureshi, F., and Terzopoulos, D. 2005. Towards intelligent camera networks: A virtual vision approach. In *Proc. Second Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 177–184.

Rabin, S. 2000. A* speed optimizations. *Game Programming Gems*, 272–287.

Reynolds, C. W. 1987. Flocks, herds, and schools: A distributed behavioral model. In *Proceedings of SIGGRAPH 87*, vol. 21(4), 25–34.

Reynolds, C. W. 1993. An evolved, vision-based behavioral model of coordinated group motion. In *Proceedings of the second international conference on From animals to animats 2 : simulation of adaptive behavior*, MIT Press, Cambridge, MA, USA, 384–392.

Reynolds, C. W. 1999. Steering behaviors for autonomous characters. In *Proc. Game Developers Conference 1999*, 763–782.

Rose, C., Guenter, B., Bodenheimer, B., and Cohen, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 147–154.

Ross, E. A. 1908. *Social Control.* The Macmillan Company, New York.

Samet, H. 1989. *Spatial Data Structures.* Addison-Wesley.

Schreckenberg, M., and Sharma, S. 2001. *Pedestrian and Evacuation Dynamics.* Springer-Verlag.

Shao, W., and Terzopoulos, D. 2005. Autonomous pedestrians. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 19–28.

Shao, W., and Terzopoulos, D. 2005. Environmental modeling for autonomous virtual pedestrians. *2005 SAE Symposium on Digital Human Modeling for Design and Engineering.*.

Shapiro, A., Pighin, F., and Faloutsos, P. 2003. Hybrid control for interactive character animation. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, 455.

Stout, B. 2000. The basics of A* for path planning. *Game Programming Gems*, 254–263.

Stoytchev, A., and Arkin, R. 2001. Combining deliberation, reactivity, and motivation in the context of a behavior-based robot architecture. In *Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 290–295.

Sturman, D. J. 1994. A brief history of motion capture for computer character animation. In *SIGGRAPH 94, Character Motion Systems, Course notes.*

Sung, M., Gleicher, M., and Chenney, S. 2004. Scalable behaviors for crowd simulation. *Comput. Graph. Forum 23*, 3, 519–528.

Sung, M., Kovar, L., and Gleicher, M. 2005. Fast and accurate goal-directed motion synthesis for crowds. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 291–300.

Tecchia, F., Loscos, C., and Chrysanthou, Y. 2002. Visualizing crowds in real-time. *Computer Graphics forum 21(4)* (December), 753–765.

Teknomo, K. 2002. *Microscopic Pedestrian Flow Characteristics: Development of an Image Processing Data Collection and Simulation Model.* Ph.D. Dissertation. Department of Human Social Information Sciences, Graduate School of Information Sciences, Tohoku University, Japan, March.

Terzopoulos, D., Tu, X., and Grzeszczuk, R. 1994. Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life 1*, 4, 327–351.

Terzopoulos, D., Rabie, T., and Grzeszczuk, R. 1996. Perception and learning in artificial animals. In *Artificial Life V: Proc. Fifth International Conference on the Synthesis and Simulation of Living Systems*, 313–320.

Terzopoulos, D. 1999. Artificial life for computer graphics. *Commun. ACM 42*, 8, 32–42.

Thompson, P., and Marchant, E. 1995. A computer model for the evacuation of large building populations. *Fire Safety Journal 24*, 131–148.

Thrun, S., and Buecken, A. 1996. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence.*

Thrun, S. 2002. Robotic mapping: A survey. Tech. rep., School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

Tinbergen, N. 1951. *The Study of Instinct.* Clarendon Press, Oxford, England.

Tomlinson, B., Downie, M., Berlin, M., Gray, J., Lyons, D., Cochran, J., and Blumberg, B. 2002. Leashing the alphawolves: mixing user direction with autonomous emotion in a pack of semi-autonomous virtual characters. In *SCA '02: Proc. of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, 7–14.

Tu, X., and Terzopoulos, D. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH 94*, 43–50.

Ulicny, B., de Heras Ciechomski, P., and Thalmann, D. 2004. Crowdbrush: interactive authoring of real-time crowd scenes. In *SCA '04: Proc. of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, 243–252.

Vote, E., Acevedo, D., Laidlaw, D. H., and Joukowsky, M. 2002. Discovering Petra: Archaeological analysis in VR. *IEEE Computer Graphics and Applications* (September/October), 38–50.

Watts, J. 1987. Computer models for evacuation analysis. *Fire Safety Journal 12*, 237–245.

Witkin, A., and Popovic, Z. 1995. Motion warping. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 105–108.

Wu, J., and Popovic;, Z. 2003. Realistic modeling of bird flight animations. *ACM Trans. Graph. 22*, 3, 888–895.

Zordan, V. B., and Hodgins, J. K. 2002. Motion capture-driven simulations that hit and react. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 89–96.