A Game-Engine-Based Platform for Modeling and Computing Artificial Transportation Systems

Qinghai Miao, Fenghua Zhu, Yisheng Lv, Changjian Cheng, Cheng Chen, and Xiaogang Qiu

***Abstract*—A game-engine-based modeling and computing platform for artificial transportation systems (ATSs) is introduced. As an important feature, the artificial-population module (APM) is described in both its macroscopic and microcosmic aspects. In this module, each person is designed similarly to the actors in games. The traffic-simulation module (TSM) is another important module, which takes advantage of Delta3D to construct a 3-D simulation environment. All mobile actors are also managed by this module with the help of the dynamic-actor-layer (DAL) mechanism that is offered by Delta3D. The platform is designed as agent-oriented, modularized, and distributed. Both modules, together with components that are responsible for message processing, rules, network, and interactions, are organized by the game manager (GM) in a flexible architecture. With the help of the network component, the platform can be constructed to implement a distributed simulation. Finally, four experiments are introduced to show functions and features of the platform.**

***Index Terms*—Artificial transportation systems (ATS), distributed simulation, game engine.**

I. Introduction

FROM THE viewpoints of sociology and anthropology, which are based on the artificial society, computational intelligence, and parallel computing, which are aimed at analyzing the transportation as a subsystem in the whole society [1], [2], the artificial transportation system (ATS) distinguished itself from traditional transportation-simulation systems. In our opinion, four key problems need to be solved to implement a complete ATS, i.e., modeling, computing, experimenting, and parallel management [3]. Here, modeling is an agent-based design of each simulation-related objects, including vehicles, pedestrians, traffic infrastructures, buildings, vegetation, and even the weather. Computing is processing the actions of all objects and interactions between objects in each time step. Experimenting stands for simulations with the variance of parameters, such as the change of the average speed when it is raining. Finally, when the generated traffic flow becomes similar to the real-world traffic flow, the evaluation results of the ATS can be used to optimize and improve the real-world traffic control, which is called parallel management. Apparently, modeling and computing are the basis of experimenting and parallel management. For these two fundamental problems, the ATS requires better solutions than traditional traffic-simulation systems. First, the transportation system is an open complex system, which should be studied using methods in complexity science. We took the principle of "emergence" to model the ATS from the bottom up [4]. That is to say, what needs to be modeled is each person of the whole population. The traffic flow naturally emerged from the population and was influenced by traveling persons. Then, one problem is how to properly manage the thousands of objects, with attributes including 3-D meshes that the object looks like. Second, as previously mentioned, an ATS study needs to cover large areas, which are usually from a town to several cities, and involves several social subsystems aside from transportation [5]. There are too many elements for a stand-alone computer to finish a step of computing in a proper time range. Therefore, we must adopt a parallel computing technology to put all the different models into a distributed framework. Each computer in the distributed framework is only a charge of a subarea, which ensures the computing performance when the population is relatively large.

To address these problems, we could use a traffic-simulation software as most researchers usually do [6] or develop a simulation system by ourselves [7]. Some traffic-simulation software has the ability of distributed computing and real-time 3-D demonstrations. Moreover, some activity-based traffic-simulation softwares, to some extent, match the ideas of the ATS, such as the TRansportation ANalysis and SIMulation System (TRANSIMS) and the best-practice model from New York Metropolitan Transportation Council. Aside from these two aspects, we want the ATS to have the ability to interact with the real world. There may be two kinds of these interactions. First, the ATS should have an interface to read in real-world information such as control signals. Second, to use the ATS as a training system, there should be client terminals from which users can "drive" in the virtual environment that is similar to playing games. Furthermore, one of our long-range goals is to make the proposed platform a new type of online game that is dedicated to transportation, which is a system that is similar SecondLife [8]. Thus, it is hard to use a traditional traffic-simulation software to construct such gamelike simulation

Manuscript received February 25, 2009; revised August 20, 2009 and April 21, 2010; accepted December 23, 2010. Date of publication January 28, 2011; date of current version June 6, 2011. This work was supported in part by the Chinese National Basic Research Program under Project 2006CB705500; by the National Natural Science Foundation of China under Project 61004090, Project 60904057, Project 60921061, Project 60974095, and Project 90920305; and by the Shandong Province Taishan Chair Professor Fund under Project 011006005. The Associate Editor for this paper was R. J. F. Rossetti.

Q. Miao is with the College of Computing and Communication Engineering, Graduate University of the Chinese Academy of Sciences, Beijing 100049, China (e-mail: miaoqh@gucas.ac.cn).

F. Zhu, Y. Lv, C. Cheng, and C. Chen are with the Key Laboratory of Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China.

X. Qiu is with the College of Mechatronic Engineering and Automation, National University of Defense Technology, Changsha 410073, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2010.2103400

platforms; on the other hand, a game engine seems to be a feasible choice.

In this paper, we introduced a game-engine-based modeling and computing platform for ATSs. The platform depends (not entirely but highly) on the Delta3D game engine [9]. We organized this paper as the following six sections. Section II introduces the backgrounds and the related works that are using a game engine as simulation platforms. Sections III and IV address the modeling problem. The artificial-population model is very important, and we have separately described it in Section III. Section IV introduces the agent-based modeling and programming for the traffic flow, which utilizes the actor-proxy mechanism of Delta3D. Section V introduces the overall architecture that depends on Delta3D; emphasis is placed on the integration of different modules under the game-manager (GM) architecture and on the way to build a distributed computing system in a client/server style. Section VI introduces three test experiments; although primary, they show that the platform is feasible and reasonable. Section VII draws conclusions and discusses some important "to do" work in the next developing cycle.

II. Backgrounds and Related Works

A. Game Engine and Simulation

Behind a lot of well-known games, e.g., Final Fantasy, people seldom know how the games were created, and most people have no idea of the difference between the game engine and the game. Similar to the engine in a car, the game engine is the heart of a game. Traditionally, a game engine is a software system that is designed for the creation and the development of video games. The core functionality that is typically provided by a game engine includes a rendering engine ("renderer") for 2-D or 3-D graphics, a physics engine for collision detection and response, and a scene graph for the management of both static and animation models, sound, scripting, artificial intelligence, networking, streaming, memory management, threading, etc. [10]. Nowadays, the game engine has become a general software platform; the application of game engines has broadened in scope. A trend is using game engines for serious games, i.e., visualization, training, medical, and military simulation applications [11]. There are a lot of such applications. For example, Ship Simulator [12] is a revolutionary simulation game that puts players at the helm of some of the most varied and detailed ships to be found at sea. Players will need to perform a multitude of tasks within a stunning 3-D environment, taking control of a wide array of ship types—from massive cargo ships and ferries to speedboats, yachts, and water taxis. Virtual Battlespace [13] is a fully interactive 3-D training system that provides a premium synthetic environment that is suitable for a wide range of military (or similar) training and experimentation purposes. In fact, a serious game comes from the battle simulation, and applications for military purposes occur far more than other areas.

In the field of traffic simulation, there are also reports mainly focused on training. In [14], a game-based learning environment is described to help novice police officers to investigate the traffic accidents in Dubai. In [15], the authors report on the construction and the evaluation of a game-based driving simulator using a real car as a joystick. The feasibility of the simulator as a learning tool has been experimentally evaluated. In [16], a complex model is described for vehicles that are involved in traffic accidents has been developed, including multibody components and different collision models.

There are a lot of game engines that are available. These game engines can be classified into three types according to their claimed intellectual property. The first ones are commercial game engines, such as the famous DOOM, Unreal, and LithTech, the latest versions of which are the most powerful. However, before using them, a very expensive license must be bought. The second ones are engines with a low cost, such as Torque and Truevision3D. They have good performance for a lot of applications with licenses that cost less than $100. The third type covers a series of open-source game engines. Some of them are the older versions of commercial game engines, such as Quake 3, which is open source under a certain license. Some of them have been developed by small or academic groups, for example, Nebula [17], Delta3D, IrrLicht [18], Crystal Space [19], etc. The performance of this type of game engine depends on specified applications and programmers' efforts for optimization. However, the knowledge base and the technical support are easily available from the official Web site, as well as open-source communities. What is more attractive is that they are usually under the Lesser General Public License (LGPL) or Massachusetts Institute of Technology license, which give people the maximum freedom to use them. That is to say, we can develop our applications based on these game engines, distribute them with our own intellectual property, and even publish them for commercial purposes.

Although game-engine technology has become more user friendly, there is still a learning curve to master all parts of a full-function game engine. Because we have been learning and using Delta3D for about two years, we choose Delta3D to construct our modeling and computing platform. Delta3D is an open-source engine, which can be used for games, simulations, or other graphical applications. Its modularized design integrates well-known open-source projects such as Open Scene Graph, Open Dynamics Engine (ODE), Character Animation Library 3D, and Open Audio Library, as well as projects such as Trolltech's Qt, Crazy Eddie's Graphical User Interface (GUI), Xerces-C, InterSense Tracker Drivers, HawkNL, and the Game Networking Engine (GNE). Rather than burying the underlying modules, Delta3D integrates them together in an easy-to-use application programming interface, which allows direct access to the important underlying behavior when needed. Furthermore, Delta3D has an extensive architectural suite that is integrated throughout the engine. This suite includes a frameworks such as the application base classes for getting started, the dynamic actor layer (DAL) for actor proxies and properties, the signal/slot support for direct-method linking, and the GM for the game management and high-level messaging for the actor communication. In fact, several other game engines are able to take Delta3D's role, e.g., IrrLicht is a good choice.

B. Related Traffic-Simulation Software

A computing platform is important to the traffic simulation. Researchers always pursue platforms with more-feasible models, a faster speed, and a larger capacity with parallel computing. There are several traffic-simulation platforms that have good performance in aspects that are previously listed, e.g., Paramics, Split, Cycle and Offset Optimization Technique (SCOOT), TRANSIMS, Vissim, and so on. Here, we will discuss TRANSIMS [20] because the proposed platform in this paper has some similar features. For example, they are both component integrated systems, both have a disaggregate model instead of a four-step model, both have microsimulation, etc.

However, a bottleneck constrains such activity-based models to narrow applications, i.e., population census data. For example, TRANSIMS adopted three types of census data: Standard Tape File 3A (STF-3A), public use microdata sample, and Master Area Block Level Equivalency (MABLE) file [21]. All the information, such as age, gender, household, incomes, vehicles, together with activity chains, depends on those data. However, in some countries or maybe in most of the world, we have no such data. For example, in China, the population is too large to collect the information of only a part of it. Additionally, the proportion of the transient population is relatively high in such a developing country, which makes it more difficult to do a census. Thus, a novel population model is needed as substitute.

In addition, there are differences that make the proposed platform unique. First, the proposed platform is based on an open-source game engine with the LGPL, without much funding. The long goal is to build a traffic-specified serious game that runs similar to a massive multiplayer online role-playing game (MMORPG). Second, the disaggregate model for the travel demand is the integration of the macroscopic and microscopic designs. It tracks physiological indexes of each person using sociology and Maslow's theory, whereas the disaggregate model in TRANSIMS tracks individuals and households based on census data. Third, TRANSIMS has a 3-D display module, but the proposed platform has a built-in 3-D rendering because it is the basic function of the game engine. Finally, the proposed platform has an interface for users to interact with the simulation, and in the future, many players can act as residents in the virtual world. Together with the artificial population that is introduced in Section III, we can construct a virtual city, which can be used to evaluate control algorithms or as a training system.

III. Artificial-Population Module

The artificial-population module (APM) is the most important module of the ATS. Here, two viewpoints of the ATS should be emphasized. First, the society is a giant complex system that includes many subsystems such as industry, commercial, culture, climate, etc. Transportation is only one such subsystem and interacts with the others [22]. Second, the human is the primary element of the society; travel is one of the human behaviors. All traveling people generate traffic flow [4]. In a traditional transportation simulation, the initial input data are usually origin–destination (OD) matrices. However, in the ATS, the initial input is the data of the population. We designed the APM to reach these requirements. That is to say, the APM takes the place of the role of the OD matrix as input data. In the APM, each single person is designed as an actor. The APM offers a mechanism to assign attributes of an actor, as well as how the attributes change over time. This mechanism can be described in two different aspects, i.e., macroscopic and microcosmic, as follows.

TABLE I
Classification of the Chinese Urban Strata

Stratum	Description	% Percent [a]
Political and business elites	Cadres from high level government, Manager, foreign executives, etc	5%-10%
Private enterprise owners	Small-scale capital holders, small property owners, individual households, small shareholders	5%-10%
Business services and civil servants	Party and government organs and enterprises, institutions, companies and other low-level clerk	20%
Industrial workers	Involved in manual, semi-manual production workers	20%-30%
Transport workers	Transport, public transport, taxi drivers as well as urban courier, etc	10%
Students	Primary and high school students	10%
Housewives and retirees	Full-time housewives, domestic service personnel and retirees	10%
Temporary residents	Short-term tourism, business trip, visiting relatives	5%
Urban unemployed	Temporarily unemployed, semi-unemployed persons	5%

[a]The percent of each stratum among whole population is approximately estimated in this paper. One can change the value accordingly.

A. Macroscopic Design

The purpose of the macroscopic design is to answer questions about where a person lives, where he will go, and how he will get there. Collecting such information person by person is impossible; thus, we adopt theories and statistic data from sociology and anthropology. In sociology, populations are divided into different strata [24]. People from different stratum have different properties such as income, education, consumption capacity, etc. For example, a chief executive officer of a big company may live in a high-order community. He drives to the office in a central business district (CBD) with his own car. A worker of a steel company lives in a common community and goes to the factory in a suburb by bus. There are strata statistic data that we can be used in the ATS. For example, the Chinese Academy of Social Sciences released a report on the Chinese strata analysis in 2001 [23]. In this report, there are ten different strata in the modern China society, and each stratum was described in detail. However, the data cannot be directly used for two reasons. First, the agricultural population should not be included in an urban traffic simulation. Although it accounts for two thirds of all of the Chinese population, their residents are not in the cities. Second, people under 16 years old, who are not included in the report, should be considered in the ATS. These young people, who are mainly students, obviously contribute to the traffic flow. After such revisions, we got a transportation-oriented classification of the Chinese urban strata, which is shown in Table I.

As shown in Table I, the urban population was classified into nine strata. When a person was generated at the beginning of simulation, a random number was used to assign him a stratum according to the percentage for which each stratum accounts. By assigning a stratum to each person, we can determine his travel origins, destinations, and travel modes in the next steps.

According to theories of urban sociology [25], we divided the city space into nine regions: 1) CBD; 2) outlying commercial district; 3) government and education district; 4) industrial park; 5) upper class residential area; 6) middle-class residential area; 7) lower class residential area; 8) sports and leisure area; and 9) transport corridor area. Each region may include one or more areas in the city. A person from a different stratum has a different probability of staying in a certain place. For example, a manager of a big company lives in an upper class community and works at a CBD, but a primary student from a family of industrial workers lives in a middle-class community and stays in school most of the daytime. In such a way, travel origins and destinations can be determined using random numbers.

We can determine the travel mode from Table I as well. People from a different stratum have different travel capacities and different travel habits. For example, a senior official may go to work via a special vehicle; a worker may go to work via bus or bicycle, whereas a student may go to school on foot. A person has access to all travel modes according to different probabilities.

B. Microcosmic Design

In Section III-A, we set several attributes of a person, e.g., potential travel origins and destinations. Here, we will address the questions about when and how these attributes change over time. After analyzing varieties of social behaviors, the American social-psychologist Newcomb conformed that the physiological status and the social impact are causes of human behaviors [26]. According to the principles of Maslow's hierarchy of needs, the five levels of needs are physiological needs (basic needs for body function), safety, love and belonging, esteem, and self-actualization [27]. There was a regular order to the needs that were added when people had enough basic necessities. The microcosmic design of the APM is based on Maslow's principles.

First, we classified a person's common behaviors into four states, i.e., physiological, working, chore, and relax states. Each state contains several specific actions. There may be a lot of actions under each state, but we only consider important ones for simplicity. For example, in the physiological state, there are actions such as eating and sleeping; in the chore state, there are actions such as shopping and financing; and in the relax state, there are actions such as going to theater or playing basketball. All the states are connected by a special state, i.e., travel, as shown in Fig. 1. In such a way, each actor is programmed as a finite-state machine.

Second, we designed a mechanism to cause the state translation. This mechanism is somewhat similar to the design of actors in video games. For each state of a person that has previously been introduced, we define a demand index, the value of which varies as time passed by. For example, a person

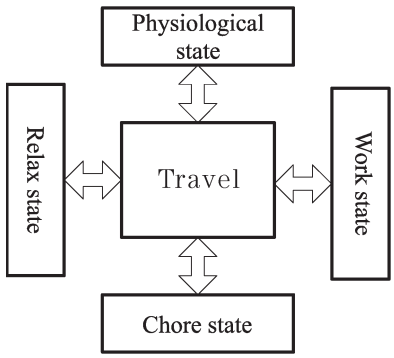


Fig. 1. Travel is a special state that connects all state translations.

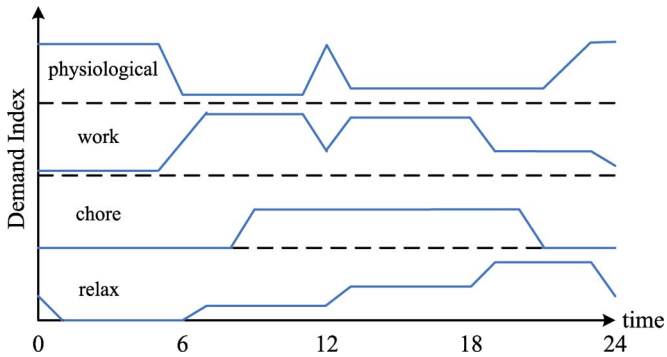


Fig. 2. Daily change of the four demand indexes.

of stratum 3 may have a set of index shown in Fig. 2. A person from a different stratum has different index curves, particularly the curves for relax and chore states. We set the minimum value of each state as 0 and the maximum value as 360. The reason is that it is easy to calculate when the simulation step is 1 s.

All the indexes are added together as a total value, i.e., IndexTotal, and the probability for a person to translate to the next state i is

$$\mathrm{Prob}_i = \frac{\mathrm{Index}_i}{\mathrm{IndexTotal}}.$$

A uniformly distributed random number, which is ranging from 0 to IndexTotal, is generated to decide which state is selected. Only when a certain state is selected three successive times, it is set as the next state. Then, the person enters into the travel state, joins the traffic flow, and travels on the road until the person's arrival.

C. Working Process of the APM

The designs that are introduced in Section III-A and B together form the APM. The APM, as an engine, runs all people that are involved in the ATS and generates the traffic flow. The people in the traffic flow only account for a small percentage of the entire population. When running, each person in the APM goes through five steps, as shown in Fig. 3:

Step 1. The APM generates a uniformly distributed random number. Then, according to the probabilities in Table I, a person gets a social stratum. Based on the stratum,

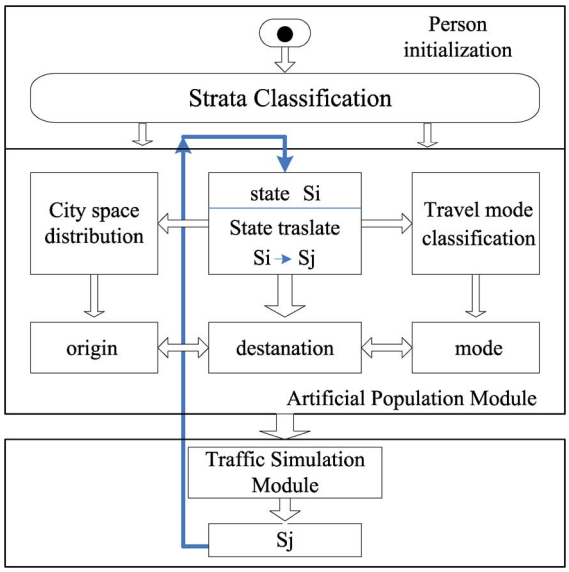


Fig. 3. Working mechanisms of the APM.

the person obtains a set of potential travel origins and destinations, as well as the travel mode.

Step 2. The APM initializes the variable set of each person according to the simulation time and assigns that person a beginning state.

Step 3. Variables change over time, and the APM arbitrates a state translation according to the decision mechanism.

Step 4. Once a translation was triggered, the person changes to the travel state. The traffic status determines how long this procedure lasts.

Step 5. The person arrives at the destination; the corresponding variable was reset by taking specific actions. The state returns to step 3, and a new cycle begins.

IV. Traffic-Simulation Module

When a person translates to the travel state, the traffic-simulation module (TSM) will handle all his actions and properties. Objects in the TSM, such as pedestrians, varieties of vehicles, roads, buildings, vegetation, etc., are all regarded as actors. All these actors are divided into two types, i.e., static and mobile. Static actors cover all the traffic infrastructures, whereas mobile actors include pedestrians and vehicles, which can change positions over time. Mobile agents are designed using the DAL of Delta3D, and the static actors are designed and configured using a 2-D editor and then loaded into Simulation, Training, and Game Editor (STAGE), which is an editor tool of Delta3D. All 3-D models in STAGE are organized as a scene graph, which can be used by the game engine.

A. DAL

First of all, the concept of the actor in games is equal to what we named as the agent in the simulation. Delta3D built an architecture that is completely generic, and it let us build our own actors. The DAL provides a flexible nonintrusive mechanism for generically exposing the properties of game actors in C++ [28].

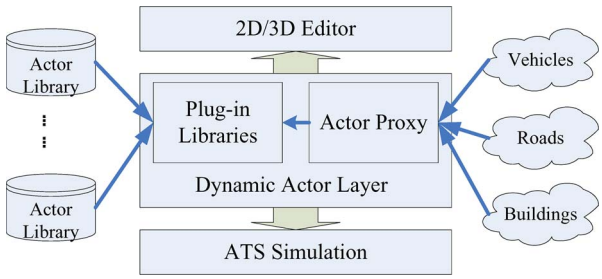


Fig. 4. DAL plays an important role in the whole platform. Through the DAL, all the agents such as vehicles, roads, and buildings can be processed in a uniform way in a dynamic-link-library file.

The two primary components of this design are ActorProxies and ActorProperties. The proxy component is a wrapper for the underlying game actor and holds a collection of the individual properties. The property component exposes the data for a single game-actor property via getter and setter function objects. The proxy knows about its properties, and the properties know how to access their data. These two components are used to generically expose all underlying data without ever modifying the original game-actor code. Using these data-driven components instead of the underlying game actor promotes data encapsulation and code reusability.

Another important component is ActorLibraries. Actor libraries are distributable components that serve to package groups of related actors and actor proxies. Actor libraries are dynamic libraries of C++ code that are loaded into the TSM or STAGE. The overview of the DAL and the editor architecture is shown in Fig. 4.

B. Mobile-Actor Design

A person in the TSM is designed as an actor, which is represented either as a pedestrian or a vehicle when traveling in road networks. A person will move from place to place in the traffic environment and will sometimes need to transfer from one host computer to another through the network. Thus, the mobility is the first feature of these agents.

When moving in the same host computer, the walking or driving actions of the actors are controlled by a cellular-automaton method. When one actor is moving out of the area that is simulated by the current host computer, a connection to the computer that is simulating the adjacent area will be set up through the network. We will introduce the distributed network later in detail.

Aside from the mobility, the mobile actors have a set of attributes. First, each actor has a unique ID for identification just as each car has a unique license. Second, each actor has position coordinates that are noted as (x, y, z) in the 3-D city space. Third, each actor has dimensions in length, width, and height. Fourth, to display the traffic simulation with 3-D rendering, each actor has a 3-D mesh model that is corresponding to its travel mode.

In addition to actions that are introduced in the APM, an actor also has some traffic-related actions. For example, selecting a route and getting a signal-light information, as well as flowing

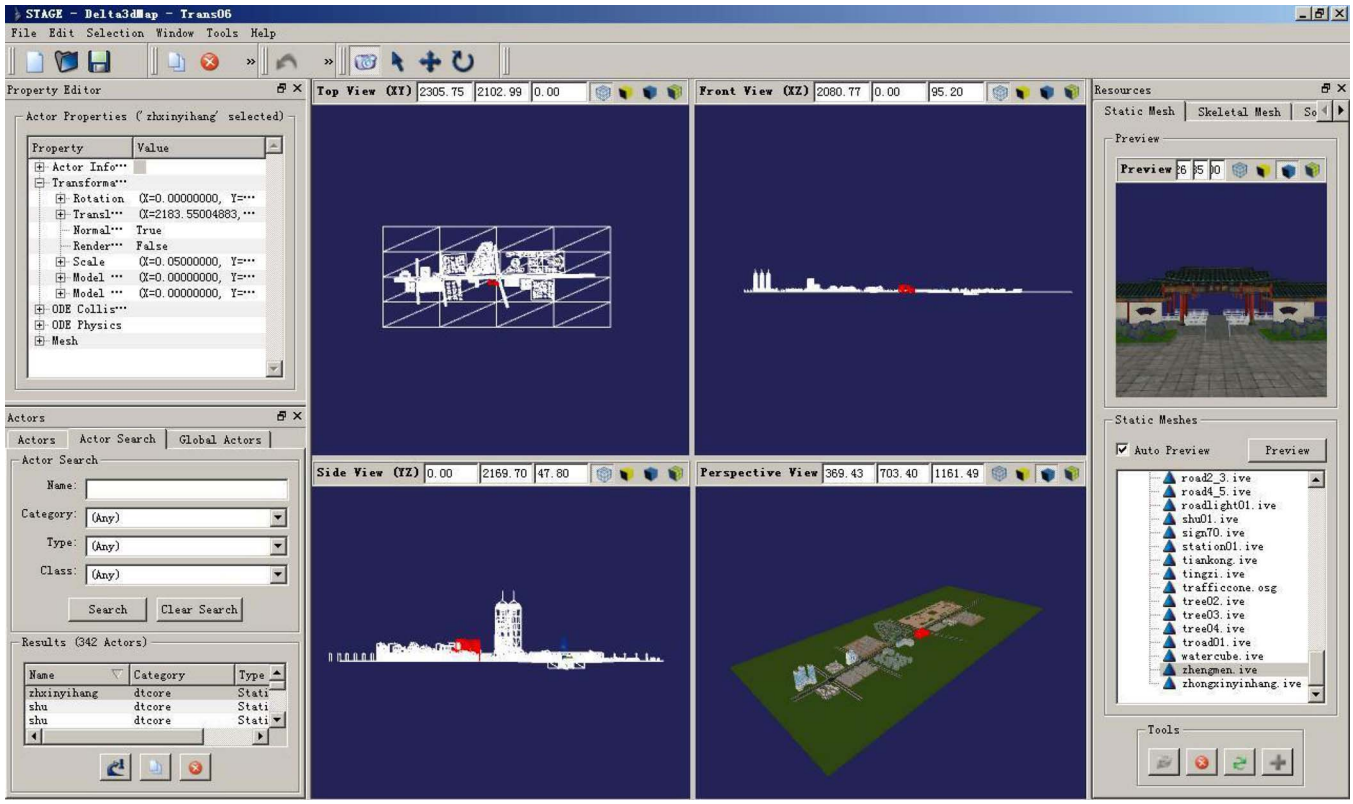


Fig. 5. Map that is constructed in the 3-D editor.

cars and changing of lanes, are all necessary for the microscopic simulation.

All the attributes and actions are capsulated in one or more ActorLibraries. Through ActorProxies and ActorProperties components, the TSM manages all actors at run time. When a new type of actors need to be added in the system, we just add it in the ActorLibrary or separately make a new library.

C. Static-Actor Design

Static actors, which include not only traffic infrastructures but also buildings, vegetation, etc., compose the traffic environment. Two steps are needed to construct such a 3-D virtual-reality environment.

First, construct and configure road networks in 2-D editors. It is a simple but laborious work when the simulation environments need to cover large areas in great detail. Thus, we developed a 2-D editor to accelerate the process. In the 2-D editor, one can draw road networks with precise coordinates, revise the number of lanes, set connections to other roads, and configure the signal-light phases, as well as place buildings and vegetation according to the real world. The constructed map is saved as extensible-markup-language (XML) files, which are used for both 3-D environment construction and simulations.

The ATS is from the point view of complex systems, which insist on studying a system from the bottom up. Therefore, there must be a communication-and-interaction mechanism between actors and environments. We designed a low-level interaction method for mobile actors. For each road link, there is a 2-D data array, which is indexed by lane numbers and road length. The position of a vehicle is marked by the element of these arrays, i.e., 1 for engaged and 0 for empty. Thus, when a vehicle drives along the link, it interacts with other vehicles on this road by inquiring neighbor elements of the array.

Second, construct 3-D environments based on the 2-D map. STAGE organizes all static actors in a hierarchical format called the map to construct 3-D scenes. In STAGE, read in the XML files that are produced by the 2-D editor. The footprints of road networks and places are shown in four views, i.e., top, front, left, and perspective, respectively. With help of STAGE, we can create new actors or delete unnecessary actors. We can assign and change 3-D meshes and accordingly translate, rotate, and scale them. We can also set up the light, the weather, and some other effects. When all static actors have been correctly placed, the 3-D environment was completed and saved as the Delta3D map, which can be directly loaded by applications. Fig. 5 shows a 3-D map being constructed in STAGE.

V. Component-Based Design and System Integration

The ATS modeling and computing platform takes advantages of Delta3D's features, such as GM, DAL, 3-D rendering, networks, physics, audio, GUI, etc. With help of the DAL, all objectives can be treated as the uniform actor format. That is to say, not only vehicles but also traffic lights, roads, buildings, trees, etc., are actors. Based on the GM architecture, the platform is modularized. Modules with different functions, such as artificial-population modeling, message processing,

user interface, network communication, etc., can be separately designed and can be added or removed from the platform when needed. Furthermore, HawkNL and the GNE enable the ATS platform to compute in the distributed mode. Alternatively, we can substitute them with the message passing interface and Open Multiprocessing to run the ATS platform on high-performance computing cluster machines.

In addition to the two important modules that were previously introduced, we designed a rule module for traffic regulations, a control module for signal-light control, a network module for the distributed computing that is based on a client/server network, an input module for user–machine interaction, a statistic module for the traffic-flow data record and display, and a hardware module for the real-world-in-loop implementation. In addition, an ATS also requires conveniently adding or removing a specific module when needed. Therefore, it is a tremendous engineering effort with numerous complexities to logically integrate all these modules into one system. The component mechanism that is offered by the GM of Delta3D alleviates this problem. The GM helps decouple applications into independent components and integrate them as one entity. Here, we give an overview of the main GM features, i.e., the GM itself, the messaging passing architecture, and the game components, together with the mechanism to construct a modularized ATS system that is based on the GM.

A. Messaging Passing Architecture

Aside from the 2-D array as the interaction space that is previously introduced, we can also utilize the messaging architecture of the GM to send and process messages among objects (actors and components) at a high level.

The GM exposes methods for sending and processing messages and knows what actors are interested in a certain message. For example, a person needs the public-transport information to select the shortest route when traveling. In our design, the public-transport information is a message that is periodically sent out by the statistic module, and actors register for this message at the GM. The message was not directly sent to actors but was sent to the GM instead. It is the GM that accordingly processes this message and sends it to the registered actors. By enforcing communication through a central interface, there is more flexibility in extending the behavior of the message passing architecture without breaking the existing code.

In addition to the messages that are related to the traffic simulation, there are special messages for the system control. For example, the tick message is the most important system message to trigger each simulation step. In the case of a networked simulation, there is a RemoteTick message that is emitted by the server, which is received by all clients as a synchronous heart beat. The GM might forward messages to the server or perform message validation. Note that the GM and the message passing architecture do not enforce either server- or client-centric games. As many or as few of the actors in the system can be simulated on the server. In a client-centric simulation, the server would just be a rule enforcer and a data collector. In a server-centric case, the clients would only exist to display the scene and accept the user input.

B. Game Components

An ATS system usually involves thousands of actors. To improve the computing performance, each individual actor is designed as simply as possible. Thus, actors are not powerful enough to handle all the ways in which messages could be used. How does an actor directly interact with the global simulation information or intercept the message stream? The answer is the components. A component is a special type of object that works with the GM. Components receive all messages and can therefore work at a much higher level than an actor. Components perform all sorts of behavior, such as network connectivity, message logging, and rule validation. Similar to actors, we can dynamically add components at either runtime or compile time. The component is a direct result of the component-based design discussion in the game-actor layer.

Components are the high-level processors in this architecture. Any time the system-level sophisticated behavior is required, a component is most likely constructed. In our implementation, we designed several important components as follows.

1) Message Processor Component. The message processor is a GM component that handles messages that are common to all games and simulations. For example, messages that an actor enters and leaves the TSM are handled by this component. Most, if not all, of the message handling methods can be overridden, thus providing a great entry point for custom-game or simulation applications to hook into the message stream if the custom behavior is required for such messages.
2) Statistic Component. The statistic component is used to collect traffic-related statistic data during the simulation. These statistic data include the traffic flow, the average speed, and the density on a certain road. Other data such as the stop times and the average queue length can also be calculated. These statistic data are important for the verification of the models, as well as the analysis evaluation of the traffic-control methods.
3) Network Component. Since the GM plays such a central role in the actor management and organization, it serves as a perfect entry point for networking capabilities. The networking capabilities are designed to follow the component-based design strategy that is mentioned at the beginning. Due to its complex nature, the networking component is actually comprised of two separate components, one of which for the server side and one for the client side. With such a design, we can configure the network that is both a peer-to-peer model and a client–server model at the same time. For actors, it is a peer-to-peer network that agents can travel from one computer to another; for traffic-statistic data, it is a client–server network from which the server collects the distributed data and displays the dynamic data charts. The networking component is responsible for any and all network connectivity and flow.
4) Input Component. As a game-engine-based platform, it is easy for users to interact with the simulation. The input component responses to the keyboard and mouse inputs. Aside from control actions such as start and stop, this component can be used for driver training. By keyboard

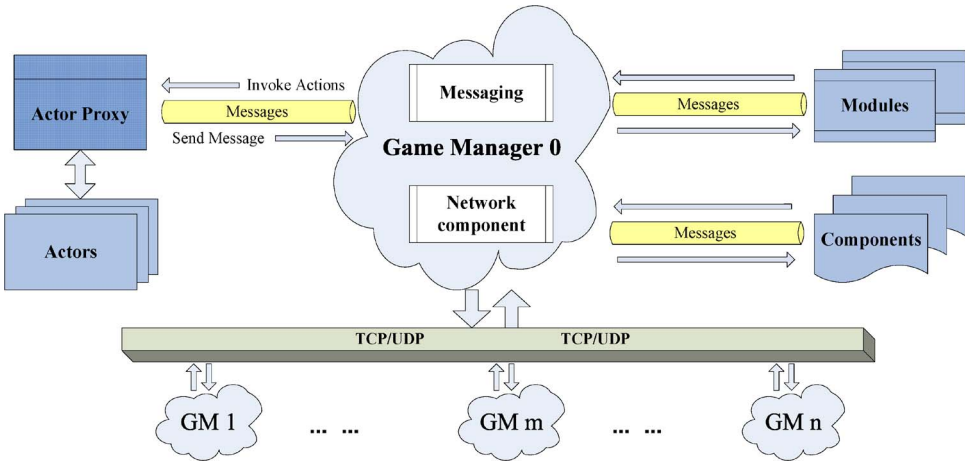


Fig. 6. Distributed architecture that is based on the GM.

and mouse or by joystick, users can drive a car in the 3-D simulation environment that is similar to playing a game. Morever, a learning management system can be integrated as the evaluation.

5) Real-World Component. This component is the interface that is used to run an ATS with the hardware in the loop. In our design, we connect ten signal controllers to our platform. The signal controllers, with control methods programmed in, interchange information according to China's National Transportation Communications for ITS Protocol.

Aside from the components that are previously introduced, there are also an XML component for reading and parsing XML documents and a heads-up display component for real-time data display. Additional components can be easily added when needed.

C. System Integration

The GM is the core of the architecture, the glue that holds everything in the system together, and the system entry point of the simulation. It is responsible for managing all actors, ensuring messaging and interobject communication, and directing the component behavior. It knows which components exist, what actors are interested in which messages, and what actors exist in the simulation. It is then responsible for making sure that messages get to the components and interested actors, as well as for handling low-level events from the Delta3D system, such as frame and preframe events.

Fig. 6 shows a high-level relationship between the GM, the actors, and the game components that were discussed here. As previously introduced, actors and components within the same GM (or the same computer) communicate by local messages, whereas actors and components in different GMs (or computers) communicate by remote messages. Remote messages are sent and received through the network component, which constructs a distributed system.

VI. Examples of Experiments

It is necessary to test and improve the proposed platform through experiments and applications. Here, we introduced four cases to test the performances of the platform in different aspects.

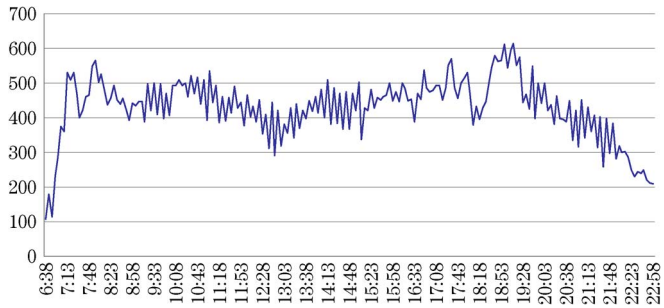


Fig. 7. Real traffic flow of Zhongguancun East Road.

A. Evaluation of the APM

In this experiment, we constructed a traffic environment of the Zhongguancun area in Beijing. The simulation was taken on the proposed platform, which is based on a road network only that extracts the main roads. The parameters are as follows. The road network includes 18 intersections and 42 road links. There are 71 places, including shopping malls, colleges, hospitals, theaters, factories, and so on. For the APM, we set a population with 50 000 persons. All persons are divided into nine strata with the proportion that is listed in Table I and live a life that is described in Section III. What we are interested in is to make a generated traffic flow approach a real traffic flow. We recorded the daily traffic flow of Zhongguancun East Road with a video camera from March 1–7, 2009. Then, we got the average flow shown in Fig. 7. We also got a simulation traffic flow of the same position by the proposed platform, which is shown in Fig. 8.

By comparing the two curves, we can see that the simulation-generated data are very different from the real data. There are two ways to reduce the differences. First, the population that is used in the experiment is too small. Because we cannot get the accurate population size of the Zhongguancun area, a series of experiments is needed to test the changes in the traffic flow with the increased population size. Second, for the simulation area is a part of the big city, we should consider the passing-through

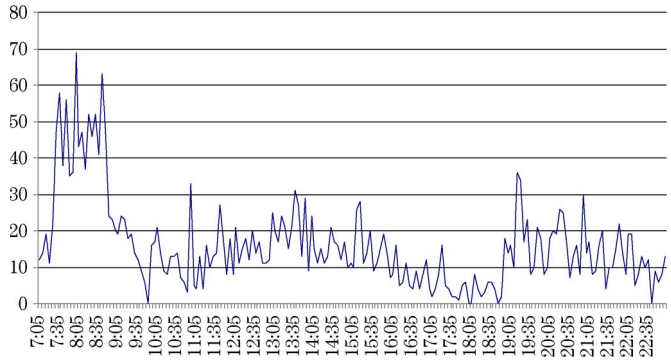


Fig. 8. Simulation-generated traffic flow of Zhongguancun East Road.

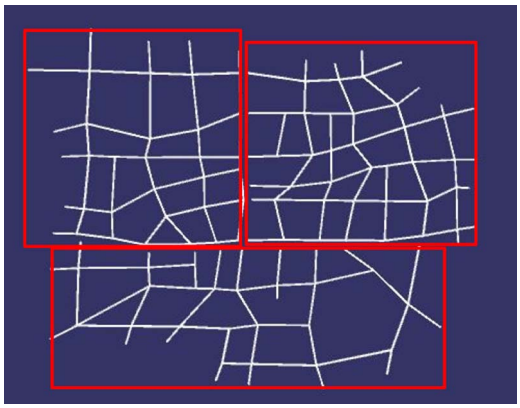


Fig. 9. Main road network of Jinan, which is divided into three parts.

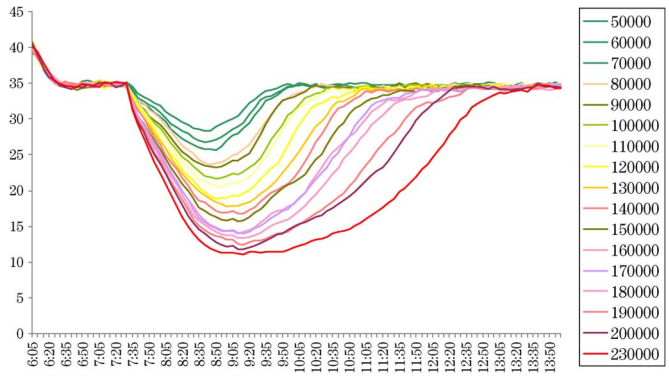


Fig. 10. Average speed drops with increased population.

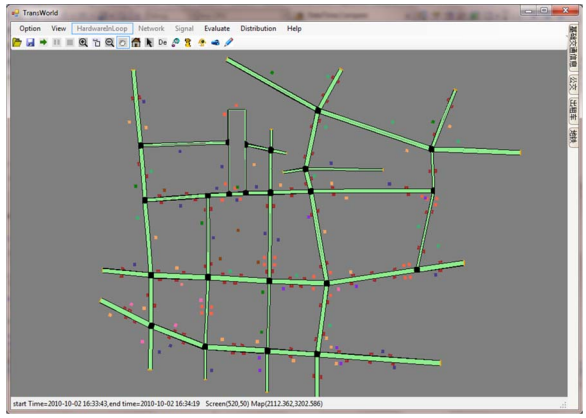


Fig. 11. Simulation road network around Guangzhou Tianhe Sports Center.

vehicles. However, in spite of the differences, the simulation-generated traffic flow has the feature of double peak in the morning and the dusk, which is similar to the real traffic flow. Of course, much effort is needed to improve the APM to get similar traffic data to the data in the real world.

B. Tests for System Performance

This experiment aimed to test the system performance, particularly for distributed simulation of large areas. The experiment constructed a road network that is covering main city roads of Jinan, China. The simulation area is 17 km in length and 15 km in width. The road network includes 81 intersections and 646 road links. There are 320 places, including all nine types. We divided the road network into three parts, as shown in Fig. 9. The system ran on three computers that are connected in a local-area-network environment; each computer was in charge of one part of the area. Vehicles (persons) travel from computer to computer if the road link is cut off and connected through the network. We ran simulations with population sizes from 70 000 to 230 000 and recorded the average speed of vehicles in the road network from 6:00 A.M. to 14:00 P.M. Fig. 10 shows that, as the population increases, the average speed of vehicles in the road network drops. The generated traffic data meet our common sense.

It took 2.5 h to simulate a whole day with a population size of 230 000, without the real-time 3-D display. The computers' platform is based on Intel Core 2 Duo E4400. It will be running much slower with the 3-D real-time display turned on, but it is still much faster as compared with a single computer.

C. Support for the Traffic Scheduling of the 2010 Asian Games

The 16th Asian Games were held in Guangzhou in November 2010. To improve the traffic environment, we were invited as a partner of the Committee of the Guangzhou Asian Games. Based on the proposed platform, we evaluated different traffic scheduling schemes around the areas of the Tianhe Sports Center. Fig. 11 shows the road network of the simulation.

The network involves 19 crosses and 79 places. The aim is to select the best scheduling scheme that helps to most quickly disperse the audiences in different conditions. The parameters are as follows: the number of people in the audience at the Tianhe Sports Center, which we take to be 30 000, 50 000, and 70 000 at most; and the percentage of different vehicles, including bus rapid transits, subways, taxis, bicycles, and by foot. Through a series of simulations, we acquired the evacuation time of different combined conditions, which were adopted by the Traffic Council of the 2010 Asia Games as a reference to optimize the traffic control. The details of this work will be introduced in another paper. Fig. 12 shows the scene of the audiences that are exiting the Tianhe Sports Center in one of the simulations.

D. Using the Platform as Driver Training System

This experiment is more like a serious game. To keep the simulation at a normal frame rate (about 30 ft/s), we took a small

Fig. 12. Audiences that are exiting the Tianhe Sports Center in the simulation.

Fig. 13. Using the platform as a training system, the car that is marked with a circle is controlled by the user.

area of Jinan, covering five intersections, with a population size of 10 000. Users take part in the simulation through the input component that was introduced in Section V. One can drive the car owned by him using the four keys, i.e., "↑, ↓, ←, and →," which is similar to what we usually do in a race game. In Fig. 13, the "beetle" car that is marked with a circle is controlled by a user. This user-controlled car is different from other actor vehicles. First, by the physics-engine ODE that is integrated in Delta3D, the car has the ability of collision diction. Second, the car is driven by users instead of a person from the APM. Such an experiment can be used to train drivers before they drive a real car on a real road.

VII. Conclusion and Future Work

This paper has introduced a modeling and computing platform based on the Delta3D game engine. The APM has been designed to take place of the OD matrix. The APM depends on the social strata and Maslow's principle and acts as the traffic demands generation. The TSM has been designed to process all the microbehaviors of vehicles or pedestrians. A pair of 2-D and 3-D editors has been used to convert or design virtual environments, including road networks and activity places. There are also components for Internet, communication, statistics, interface, etc. All the modules and components are organized by the GM of Delta3D in a flexible architecture. Each module can be added and substituted as needed.

Results of experiments show that this modeling and computing platform has the basic functions as a simulation platform. However, there is a lot of work to do on this platform. For example, a more detailed and larger scale APM is needed to replace this one, and consequently, a larger network or a cluster is necessary to do ATS experiments for big cities such as Beijing. In addition, we are working on making the platform an online system that is similar to a MMORPG. This is interesting but more challenging, for the revision of the engine and a more-powerful network component must be added.

Acknowledgment

The authors would like to thank Prof. F.-Y. Wang for his intensive instructions, the Laboratory of Complex Adaptive Systems for Transportation, Prof. S. Tang, and the hard work of H. Zhao and Y. Ou. Without their support, the authors could not have implemented the introduced platform.

References

[1] F. Y. Wang, "Toward a revolution in transportation operations: AI for complex systems," *IEEE Intell. Syst.*, vol. 23, no. 6, pp. 8–13, Nov./Dec. 2008.
[2] F. Y. Wang and S. J. Lansing, "From artificial life to artificial societies: New methods in studying social complex systems," *J. Complex Syst. Complexity Sci.*, vol. 1, no. 1, pp. 33–41, 2004.
[3] F. Y. Wang and S. M. Tang, "Artificial societies for integrated and sustainable development of metropolitan systems," *IEEE Intell. Syst.*, vol. 19, no. 4, pp. 82–87, Jul./Aug. 2004.
[4] M. M. Waldrop, *Complexity: The Emerging Science at the Edge of Order and Chaos*. New York: Simon and Schuster, 1992.
[5] F. Y. Wang, "Computational experiments for behavior analysis and decision evaluation of complex systems," *J. Syst. Simul.*, vol. 16, no. 5, pp. 893–898, 2004.
[6] A. Kotsialos, M. Papageorgiou, C. Diakaki, Y. Pavlis, and F. Middelham, "Traffic flow modeling of large-scale motorway networks using the macroscopic modeling tool METANET," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 4, pp. 282–292, Dec. 2002.
[7] J. Maroto, E. Delso, J. Félez, and J. M. Cabanellas, "Real-time traffic simulation with a microscopic model," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, pp. 513–527, Dec. 2006.
[8] [Online]. Available: http://secondlife.com/
[9] [Online]. Available: http://www.delta3d.org
[10] J. Simpson, Game Engine Anatomy. [Online]. Available: http://www.extremetech.com/article2/0,2845,594,00.asp
[11] M. O. H. Duin, "Serious game development by distributed teams: A case study based on the eU project prime," in *Proc. 3rd Int. Conf. WEBIST*, vol. 3, *Society, e-Business and e-Government, e-Learning*, Barcelona, Spain, 2007.
[12] [Online]. Available: http://www.shipsim.com/home.php
[13] [Online]. Available: http://virtualbattlespace.vbs2.com/
[14] A. Binsubaih, S. Maddock, and D. Romano, "A serious game for traffic accident investigators," *Interactive Technol. Smart Educ.*, vol. 3, no. 4, pp. 329–346, 2006.
[15] P. Backlund, H. Engstrom, M. Johannesson, and M. Lebram, "Games and traffic safety—An experimental study in a game-based simulation environment," in *Proc. 11th Int. Conf. Inf. Vis.*, Jul. 4–6, 2007, pp. 908–916.
[16] J. Félez, J. Maroto, G. Romero, and J. M. Cabanellas, "A full driving simulator of urban traffic including traffic accidents," *Simulation*, vol. 83, no. 5, pp. 415–431, May 2007.
[17] [Online]. Available: http://nebula.emulatronia.com/
[18] [Online]. Available: http://irrlicht.sourceforge.net/index.html
[19] [Online]. Available: http://www.crystalspace3d.org/main/Main_Page
[20] [Online]. Available: http://transims-opensource.net/
[21] Virginia Tech, TRANSIMS Fundamentals. [Online]. Available: http://www.transims-opensource.net
[22] F. Y. Wang *et al.*, "A complex systems approach for studying integrated development of transportation, logistics, and ecosystems," *J. Complex Syst. Complexity Sci.*, vol. 1, no. 2, pp. 60–69, 2004.

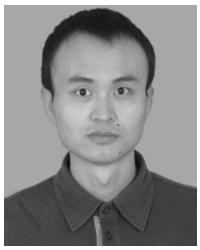
[23] X. Y. Lu, *Report on Social Strata of Modern China*. Beijing, China: Social Sci. Academic, 2002.
[24] A. Giddens, *Sociology,* 4th ed. Beijing, China: Beijing Univ. Press, 2006.
[25] Y. F. Zheng, *Urban Sociology*. Beijing, China: China City, 2002.
[26] D. Coon, *Introduction to Psychology: Gateways to Mind and Behavior,* 9th ed. Beijing, China: China Light Ind., 2004.
[27] A. H. Maslow, *Toward a Psychology of Being,* 3rd ed. Hoboken, NJ: Wiley, 1998.
[28] BMH Associates, Inc., Game Manager, Delta3D Game and Simulation Engine SOFTWARE DESIGN DOCUMENT, 2005. [Online]. Available: http://www.delta3d.org

Qinghai Miao received the Ph.D. degree in automatic-control engineering from the Chinese Academy of Sciences, Beijing, China, in 2007.

Currently, he is a Lecturer with the College of Computing and Communication Engineering, Graduate University of the Chinese Academy of Sciences. His research interests include software agents and multiagent systems, parallel computing and high-performance computing, and massively distributed simulation of artificial transportation systems.

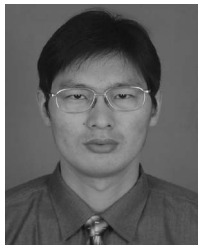
Fenghua Zhu received the Ph.D. degree in control theory and control engineering from the Chinese Academy of Sciences, Beijing, China, in 2008.

He is currently an Associate Researcher with the Key Laboratory of Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences. His research interests include artificial transportation systems and parallel-transportation management systems.



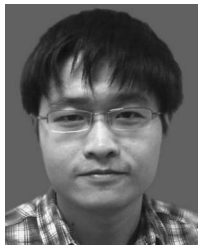
Yisheng Lv received the Ph.D. degree in control theory and control engineering from the Chinese Academy of Sciences, Beijing, China, in 2010.

He is currently an Assistant Researcher with the Key Laboratory of Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences. His research interests include intelligent transportation systems and emergency transportation management.



Changjian Cheng received the Ph.D. degree in chemical engineering from the Chinese Academy of Sciences, Beijing, China, in 2008.

Currently, he is an Associate Professor with the Institute of Automation, Chinese Academy of Sciences. His research interests include multiagent systems, parallel-management theory and applications, and high-efficiency computing and simulation of transportation systems.



Cheng Chen is currently working toward the Ph.D. degree with the Key Laboratory of Complex Systems and Intelligence Science, Chinese Academy of Sciences, Beijing, China.

His research interests include multiagent systems and distributed artificial intelligence and its applications.

Xiaogang Qiu received the Ph.D. degree in system simulation from the National University of Defense Technology, Changsha, China.

Currently, he is a Professor with the College of Mechatronic Engineering and Automation, National University of Defense Technology. His research interests include simulation, multiagent systems, knowledge management, and parallel control.