3-tier Architecture for Pedestrian Agent in Crowd Simulation[1]

GAO Peng, XU Ruihua

School of traffic and Transportation Engineering, TongJi University
CaoAn Road 4800#, Campus JiaDing, Shanghai 201804, China
Email: blue.silver@hotmail.com

Abstract: After extensive investigation and in-depth study of crowd dynamics with Chinese characteristics, especially in the stations of Urban Mass Transit, we proposed a new architecture for agent oriented pedestrian simulation. It's designed to simulate the movement of thousands of individual pedestrians through large, geometrically complex 2D space.

Our PSS (Pedestrian Simulation System for Urban Mass Transit Station) based on the architecture has been developed. The results of the case studies show that the architecture is feasible and effective in simulating intelligent human behaviors. Characteristic by openness and scalability, it can be easily applied to other situations, such as Olympics and expo, etc.

Keywords: Modeling crowd dynamics, Pedestrian agent, Agent oriented simulation, Crowd Simulation, Urban Mass Transit Station

1 Introduction

Construction of railway transit station requires a significant amount of human, material and financial resources. Besides, it's very expensive or impossible to make any alteration after the blueprints turn into realities. The issue of security, convenience and coziness for passengers, and the facility utilization, operation costs for operators largely depend on the size, space structure and equipments layout of railway station. Moreover, how to deal with the emergencies and how to evacuate passengers timely in the case of fire, explosion and poison gas is a formidable problem.

Computer simulation is a feasible and effective way to test and evaluate optional designs and emergency strategies of railway station. However, existing pedestrian simulation systems usually are:

1. Based on CA (Cellular Automata) theory, so the movement of pedestrians seems a little "mechanical". Because the velocity is discrete and it's hard to take acceleration into consideration.
2. Inadequate for intelligent real-time path planning, especially simulating large number of pedestrians. Users often have to add certain kinds of navigations by hand.

Aiming at simulating accurately and effectively the movement of thousands of pedestrians through large, geometrically complex 2D space, a 3-tier Architecture is presented in this paper, which combines many algorithms and models to deal with the problems such as "mechanical movement", intelligent real-time path planning and so on.

1 Sponsored NSFC (National Natural Science Foundation of China) Project: "Security and Reliability of Urban Mass Transit Operation", Approved Code: 50478105.

2 How the issue raised

In the field of Game Programming and Artificial Intelligence, there are many studies on multi-agent movement. Programmers are usually confronted with several common problems, e.g. “conflicts in path planning”, “realistic movement” and “long-distance path finding”.

2.1 Conflicts in path planning

In the multi-agent system, one agent may easily collide with others when moving along with the path that previously planned. Because the map for search algorithm is changing with time (other agent as obstacles is moving).

One of the strategies to solve the problem is “Waiting until others go away”. With this strategy, queuing effect can be observed sometimes. Unfortunately, it may lead to “dead lock” when two agents are waiting for each other. After that, more and more agents get together stupidly. Another strategy is “re-plan the path when conflicts occur” (e.g. “Local Repair A*”, which describes a family of algorithms widely used in the video-games industry). Although it can solve the conflicts sometimes, often incur substantial computational overhead.

In order to avoid game characters huddle together stupidly, many algorithms have been developed, e.g. David Silver [1] and Alborz Geramifard [2] (Department of Computing Science, University of Alberta, Canada) proposed “Cooperative Pathfinding”. This algorithm add time dimension into search space, which is originally 2D. With full information about the routes of others, agents could find non-colliding routes to separate destinations.

After experiments we found that, there are still some problems if applying “Cooperative Pathfinding” to pedestrian simulation:

1. Agents can’t achieve acceleration variable, curvilinear movement, or it’s too complex to search non-collide path.
2. Hard to keep efficiency when the number of agents is too lager (10,000 for example).

So we strike out in another direction: do not try to solve these problems within pathfinding, while separate the tasks and set a tier to take charge of them. The paths by search algorithm are only used as rough navigation. In this way, the agents not only could achieve acceleration variable, curvilinear movement, but also avoid each other smoothly.

2.2 Realistic movement

Since the output paths by basic search algorithm seem too stiff and often have “zigzag effect”, many methods have been proposed to make the movement of game characters more realistic and visually interesting.

For example: Marco Pinter, expert of game design and development, from California, suggests that we can add some realistic curved turns for agents, so that they don't appear to change direction abruptly. (As Fig. 1)

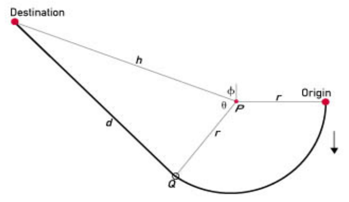

Fig. 1 Adding Realistic Turns

While after some tests we found that: comparing with calculating curved turns in geometric way, better performance and less calculation costs can be achieved when turning to consider the physical parameters of the agents. (Force exerted by other objects, acceleration, speed, etc.) It seems a simpler, more natural way to achieve smooth and realistic movement.

2.3 Long-distance path finding

Generally speaking, a map with 10,000 nodes can be called "large scale". Since the computational cost increases rapidly with size of search space, pathfinding on large maps can result in serious performance bottlenecks.

R.C. Holte (Computer Science Dept., University of Ottawa, Canada) [4] presents a hierarchical approach for reducing problem complexity. This technique abstracts a map into linked local clusters (as Fig. 2-2), Small clusters are grouped together to form larger clusters. At the global level, clusters are traversed in a single big step. In this way, large number of meaningless nodes (e.g. the nodes in cluster 1and 4, Fig 2) can be avoided. A hierarchy can be extended to more than two levels.

Based on "Hierarchical Pathfinding", this paper abstracts the behavior of agents into a series of "events", then separates the task of "big-step search" and set a tier to manage these events. The events also can be organized in hierarchical structure, so similar results can be obtained with more flexibility.

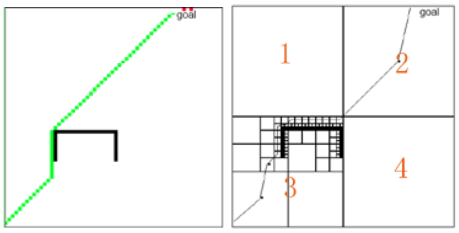

Fig. 2 Hierarchical Pathfinding

From the above analysis, we can conclude that: to these difficulties in pedestrian simulation, it's usually very hard to be solved within one level. So we decoupled the task and proposed an architecture, which combined many algorithms in different levels. Based on the architecture, computer program is clearly structured, explicit in division of responsibility, and easy to implement.

3 Structure of the architecture

Comparing with automobile, pedestrian is more free and flexible, whose behavior is complex and is easily affected by other pedestrians or the environment. So we can hardly describe the nature of crowd with one set of formulas or simulate pedestrian behaviors with one kind of model. In this paper, pedestrian's complex behaviors of are decoupled into 3 tiers, which are from macroscopic to microscopic (as Fig. 3):

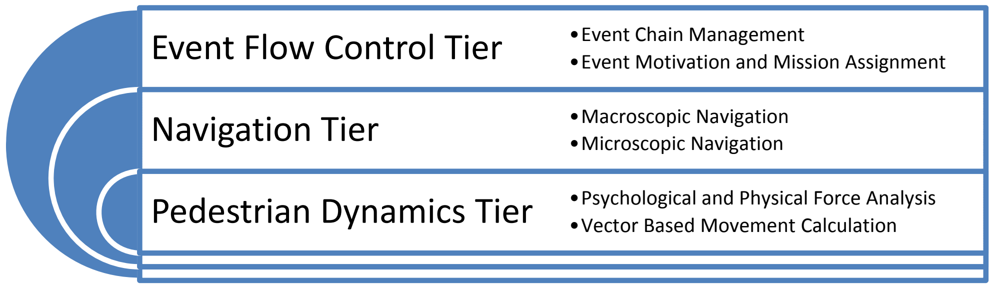

Fig. 3 Structure of the Architecture

3.1 Event Flow Control Tier

The first tier called "Event Flow Control", which is responsible for event flow management (such as walking, queuing, buy tickets, etc.) and event motivation (such as detour, overtaking, etc.).For the sake of convenience, the following paper will take the passengers who enter the subway station as an example, to demonstrate how the architecture works over the simulation process.

In general, the behavior of agent in simulation system is composed of a series of events. To passengers who enter the subway station, it might be such a flow:"enter - buy tickets - check in - through staircases - waiting train - boarding". Each event can be divided into many sub-events, e.g. "buy tickets" may contain: "choose a tickets area - walk to the area - choose a tickets device - queuing - buy tickets". Each sub-event also can be divided into even smaller events, e.g. "walk to tickets area" can be made up of "route choice", "speed up/down", "make a detour", etc.

Therefore, events are divided into 3 categories in this paper: "macro-events", "meso-events", and "micro-events". Proper data structures (class) have been designed for each category, and an enumerable variable has been set to identify the status of the agent. Combining events of different levels, we can get a tree-like diagram of status (due to the limitation of space, there are only the events mentioned above):

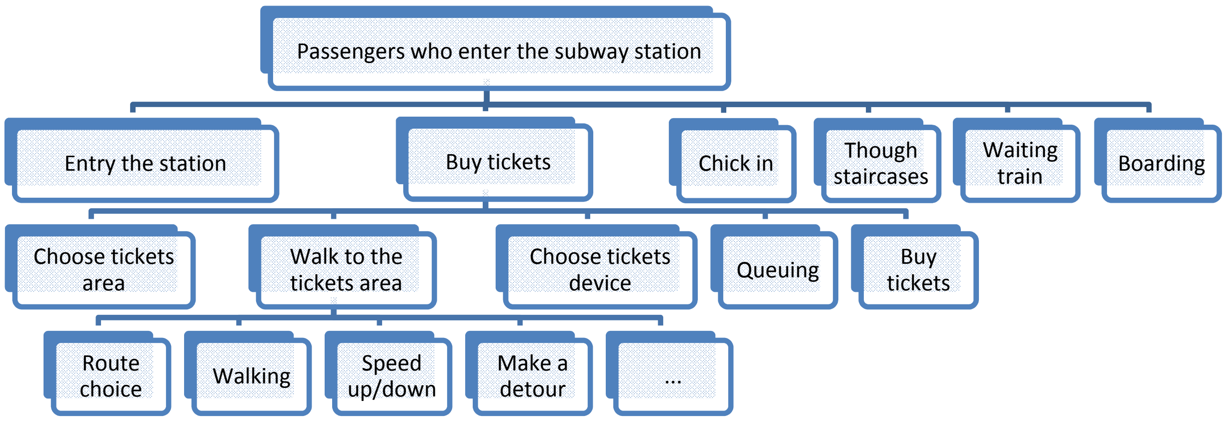

Fig. 4 Tree-like diagram of agent status

The flow of macro-events can be specified by users; otherwise, some search algorithm (such as Ants Colony) also can be used if you like the agent smarter. Relevant floor, facilities and flow of meso-events can be stored in the data structure of the macro-event. Since the motivation of micro-events is relatively random (they can't be managed in the way of flow), so we set an engine to invoke micro-events on certain condition. By the way of changing the status variables, tiers can communicate with each other.

In brief, events at all levels are managed in a hierarchical way. To different situations (such as Olympics and expo, etc.), corresponding event control tier can be designed to meet specific needs. Furthermore, some complex social behavior model can be integrated into the architecture, served as the first tier.

3.2 Navigation Tier

After the "Event Flow Control Tier" makes sure the status of a agent, it may assign some missions, and transmit parameters to lower tiers, depending on the characteristics of current status diagram (is it leisure or in hurry, walking on passageway or staircase and so on). Among all missions, the most common one is "move", so navigation is crucial.

In real life, pathfinding of pedestrians is usually not completed at one time. On the contrary, they would like to divide the whole journey into many small parts, and set many local targets (e.g. a guidance, a corner, etc.), and then find a comfortable local path.

Hence, the "Navigation Tier" is divided into 2 levels:

1. Macro-navigation: refers to identify a rough route from the source to the destination;

2. Micro-navigation: is responsible for real time path planning (depending on density, geometry, etc).

For effective pathfinding, we adopt A* algorithm, which is a typical heuristic search algorithm and is widely used in Game Programming, Artificial Intelligence and robotics. Heuristic search means estimating the cost of every location in the search space, and choosing the lowest one for next step, so tremendous unnecessary searching can be avoided and high efficiency can be obtained.

However, basic A* algorithm is inadequate for pedestrian simulation:

1. To large-scale pedestrian simulation system, there can be thousands of agents require real-time pathfinding, and that may easily incur computational overhead. So we need to optimize A*.
2. For A* is grid-based, the output path usually get "zigzag effect" (see Fig. 5 Left), while nobody would like to walk in that way in real life.
3. The output paths by basic A* seems "clinging to obstacles" too much, but pedestrian would like to make a detour earlier before the obstacle in real life (see Fig. 5 Right).

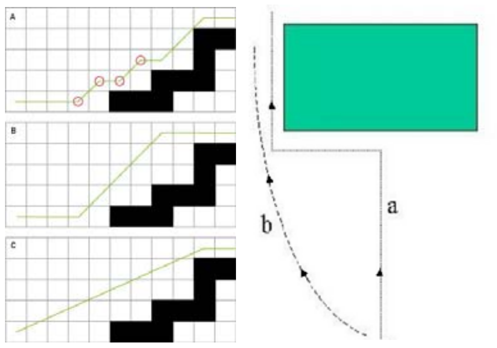

Fig. 5 Problems of basic A*

To solve the problems mentioned above, many modifications have been done:

1. Special data structure is designed for the implementation of A*, so it can achieve high performance and meets the pathfinding needs of numerous agents.
2. The problem of the output path "clinging to obstacle" of the basic A* has been solved by imposing penalty on nodes near obstacles in map preprocessing.
3. A "Trimming Algorithm" is developed, which could reduce the "zigzag effect" and achieve smooth straight-line movement.
4. Several heuristic functions are compared and improved. A* has an ability of vary its behavior based on the heuristic functions; we take advantage of the ability to embody pedestrian's route preference.

For more details, please refer to "A Modified Heuristic Search Algorithm for Pedestrian Simulation" [5].

3.3 Pedestrian Dynamics Tier

After the problem of Navigation, the next question is how the agents move and how to avoid each others. One of the simple ways is CA (Cellular Automata): the movement of agents based on grid and they move one square for each step. Although this approach is easy to implement with less computational costs, it's hard to embody complex interaction between agents.

In this paper, we introduce "Social Force Model" into Pedestrian Dynamics Tier, which is responsible for psychological and physical force analysis, and vector based movement calculation (see Fig. 6).

$$\underbrace{m_i \frac{d\vec{v}_i(t)}{dt}}_{\text{Acceleration}} = \underbrace{\frac{m_i}{\tau_i}\left(v_i^0 \vec{e}_i(t) - \vec{v}_i(t)\right)}_{\text{Driving Force}} + \underbrace{\sum_{j(\neq i)} \vec{F}_{ij}^{ww}(t)}_{\text{Interactions}} + \underbrace{\vec{F}_i^{b}(t)}_{\text{Borders, Fire}} + \underbrace{\sum_k F_{ik}^{att}(t)}_{\text{Attractions}} + \underbrace{\vec{\xi}_i(t)}_{\text{Fluctuations}}$$

$$\downarrow$$

$$\vec{F}_{ij}^{ww}(t) = \underbrace{\vec{F}_{ij}^{psy}(t)}_{\text{Psychological Repulsion}} + \underbrace{\vec{F}_{ij}^{ph}(t)}_{\text{Physical Interactions}} + \underbrace{\vec{F}_{ij}^{att}(t)}_{\text{Attractions between People}}$$

Fig. 6 Basic equations of Social Force Model

Social Force Model is presented by Dirk Helbing et al. [6] in 1998. “Social force” represents the effect of the environment (e.g. other pedestrians or borders) on the behavior of the described pedestrian.

This equation in Fig. 3-4 can be interpreted as: the “social force” that imposes on a pedestrian at a time can be decomposed into 5 forces:

1. Driving Force: the motivation of getting to its destination.
2. Interactions: the motion of a pedestrian is influenced by other pedestrians.
3. Repulsions: repulsive force exerted by obstacles, fire and so on.
4. Attractions: something interesting attracts the pedestrian.
5. Fluctuations: some stochastic fluctuations.

Among the 5 forces, “Interactions” also can be decomposed into 3 forces:

1. Psychological repulsions: Pedestrians like to keep certain distance from each other.
2. Physical interactions: physical force exerted by others with body contact.
3. Attractions: attractions between people, e.g. friends and relatives.

Social Force Model can simulate acceleration and deceleration of pedestrians accurately, so we can gain experimental data to support facility comfort evaluation. Besides, it can solve the problem of path conflicts and achieve agents avoiding each other smoothly. The series of points by micro-navigation algorithm can serve as local targets for Social Force Model. One thing need to be emphasized is that the distances between each two points need some more considerations: If the distances are too large, then the agents are constrained on the path and Social Force Model could not work well. On the contrary, if the distances are too small, then the agents may easily get lost.

Although Social Fore Model could simulate some behaviors in the perspective of pedestrian dynamics, it also has two shortcomings: first, it has no ability for path planning; second, computational cost is high. To the first question, we combine navigation algorithm with Social Force Model, then solved the problem. To the second question, we add an inter-medium (called “info space”) into Social Force Model, then the agents exchange information through “info space” other than communicating directly, so large amount of nonsensical communications are voided.

4 Computer experiments

In order to collect human and social behavioral data, specify the parameters of models in the architecture, and validating consistency and reliability of our system, we made many on-spot investigations in Shanghai. Such as transfer station "People Square", intersection of Middle Xizhang Road - West Nanjin Road etc. The results of the case studies show that the architecture is feasible and effective in simulating intelligent human behaviors.

4.1 Finger Effect

In the high density crowd, especially bidirectional flow, pedestrians usually follow others to reduce resistance. Then the two opposing flows self-organize into long chains of people passing each other, like conga lines at a party. This is a phenomenon unique to interactive systems but not fluid: so called "Finger Effect".

"Finger Effect" can be obviously observed in real life (Fig. 4-1 left), as well as in the animation of our simulation system (Fig. 7 right).

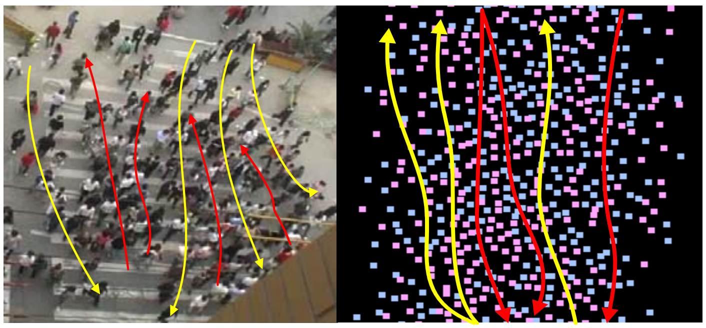

Fig. 7 "Finger Effect" in real life and in our simulation system

4.2 Edge Effect.

According to study of G. Keith Still (University Warwick of London [7]): in a unidirectional high density flow, pedestrians at the edge can move faster than whom at the center. By the "density map", which is calculated by our simulation system automatically, we also witnessed this phenomenon clearly.

4.3 Case study with "dead end"

As Fig 8 shows, in this case, there are large numbers of pedestrians enter from the right-top corner of the map and exit at the left-bottom corner. There is a "dead end" in the middle of their journey, which they can see it clearly before entry. Generally speaking, pedestrians would never walk deeply into the dead end in real life, since they can see it. But to simulation system, it's not very easy to do this without incurring computational overhead.

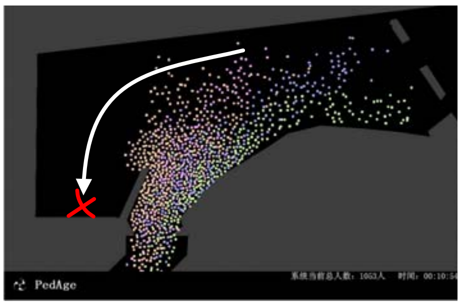

Fig. 8 Case study with "dead end"

Since the architecture proposed in this paper has the ability of real time path planning, agents would never "been trapped", and the simulation result is consistent with reality.

5 Conclusions

In the field of Game Programming and Artificial Intelligence, there are many studies on multi-agent movement. A number of approaches have been proposed to solve some common problems, such as “conflicts in path planning”, “realistic movement” and “long-distance path finding”. Being confronted with similar problems in pedestrian simulation, we designed a new architecture to combine many algorithms to handle them, other than trying to do this within one tier. The advantages are:

1. With clear structure and explicit components responsibility, the architecture is flexible and scalable;
2. The 3 tiers is from macroscopic to microscopic, which is consistent with behavior process of pedestrians in real life;
3. Pedestrian agents based on this architecture have ability of real time path planning. Comparing with the simulation systems that require user to add navigation objects manually, our system can obtain more real and objective results with much less efforts.
4. Due to Social Force model being integrated, the movement of agents is more realistic than Cellular Automata, so our system can provide experimental data support for comfort evaluation. Besides, the problem of “conflicts in path planning” is solved.

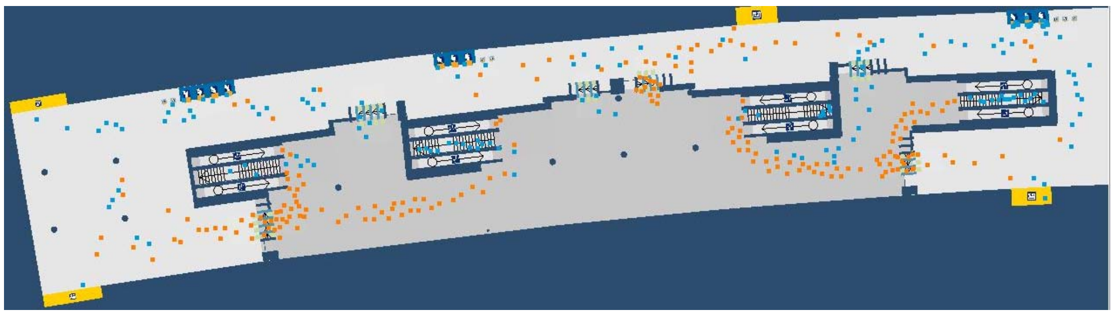

Fig. 9 Simulation snapshot of transfer station “Zhongshan Park” of Shanghai

Our PSS (Pedestrian Simulation System for Urban Mass Transit Station) based on the architecture has been developed in C#. It’s a self-contained system characteristic by “AutoCAD Drawing Processing Technology”, “Image Analysis Technology”, and Agent oriented Simulation.

The feature of real-time path planning in our system can significantly reduce the workload of creating simulation projects (no longer requires user to add navigation objects manually). Besides, it reduces the impact of human factors (users) on simulation results. Fig. 9 shows Simulation snapshot of transfer station “Zhongshan Park” of Shanghai.

Since the tiers are independent, you can design a new “Event Flow Control Tier” or integrate a more complex “social behavior / decision model” instead to meet specific needs. In a word, the architecture is open and scalable, and can be easily applied to other situations, such as Olympics and expo, etc.

References

1. David Silver . Cooperative Pathfinding . [A]. R. Michael Young, John E. Laird (Eds.): Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference [C]. June 1-5, 2005, Marina del Rey, California, USA. AAAI Press 2005
2. Alborz Geramifard, Pirooz Chubak . Efficient Cooperative Path-Planning . Computing Science Department, University of Alberta. October, 2005
3. Marco Pinter . Toward More Realistic Pathfinding [J]. Game Developer, 2001, 3

4. R.M. Zimmer, R.C. Holte, et al . Hierarchical A*: Searching Abstraction Hierarchies Efficiently [M] . Lecture Notes in Computer Science . Berlin / Heidelberg : Springer, 2005
5. Gao Peng, Xu Ruihua and Zou Xiaolei . A Modified Heuristic Search Algorithm for Pedestrian Simulation . [A]. Proceedings of the 7th International Conference of Chinese Transportation Professionals[C]. Shanghai, China: ASCE, 2007.9
6. Dirk Helbing .Traffic Jams, Pedestrian Streams, Escape Panics, and Supply Chains [J]. Reviews of Modern Physics, 73(4), 1067-1141 (2001).
7. G. Keith Still. Crowd Dynamics [D]. London: University of Warwick, BSc Physical Sciences (Robert Gordon's Institute of Technology, Aberdeen 1981), 2000