European Journal of Scientific Research
ISSN 1450-216X Vol.15 No.1 (2006), pp.6-17

http://www.eurojournals.com/ejsr.htm

Collision Avoidance in Crowd Simulation with Priority Rules

Cherif Foudil
Computer Science Department, Biskra University
Biskra 07000, Algeria
E-mail: foud_cherif@yahoo.fr

Djedi Noureddine
Computer Science Department, Biskra University
Biskra 07000, Algeria
E-mail: djedi_nour@yahoo.fr

Abstract

Motion planning for multiple entities or a crowd is a challenging problem in today's virtual environments. We describe in this paper a system designed to simulate pedestrian behaviour in crowds in real time, concentrating particularity on collision avoidance. On-line planning is also referred as the navigation problem. Additional difficulties in approaching navigation problem are that some environments are dynamic. In our model we adopted a popular methodology in computer games, namely A* algorithm. The idea behind A* is to look for the shortest possible routes to the destination not through exploring exhaustively all the possible combination but utilizing all the possible directions at any given point. The environment is formed in regions and the algorithm is used to find a path only in visual region. In order to deal with collision avoidance, priority rules are given to some entities as well as some social behaviour.

Keyword: Path finding, Collision avoidance, behavioural animation, Crowd simulation

1. Introduction

The autonomy of a virtual human is defined by its capacity to perceive, act and decide of its actions. The behaviour is usually described through several simple skills that can be mixed to generate a more complex and credible behaviour. One of the most important skills is the ability to navigate inside a virtual environment as it is part of a large number of behaviours. Reproducing this fundamental behaviour requires to address different topics such as the topological model of the environment, path planning and collision avoidance techniques.

In order to animate a crowd of pedestrians in real-time, each of these techniques should be optimized without leaving out behavioral studies. A crowd is not only a group of many individuals: crowd modelling involves problems arising only when we focus on crowds. For instance, collision avoidance among a large number of individuals in the same area requires different resolving strategies in comparison with the methods used to avoid collisions between just two individuals. Also, motion planning for a group walking together requires more information than needed to implement individual motion planning. 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 perceptual, behavioural 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.

In this article, we propose a general model, inspired by studies on human behaviour to simulate the navigation process in dynamic environments. We focus our work on the methods of collision avoidance.

2. Related Works

The simulation of behaviour has been studied since the earliest days of computer graphics research. Early work concentrated on animal behaviour, with birds a popular choice, but recently there has been a lot of work on human behaviour. Techniques for simulating a crowd as a single entity have been proposed, as well as those which consider each person in the crowd separately. In the virtual environments community, the most common approach to simulating group movement is to use flocking. The concept of flocking was introduced by Reynolds [15]. His boids-model described the behaviour of the units in a group using only local rules for the individual units. Later, Reynolds extended the technique to include autonomous reactive behaviour [16]. The idea is that units steer themselves in such a way that they avoid collisions with other units and the environment, while at the same moment they try to align themselves with other units and try to stay close to the other units.

In open areas this leads to rather natural group behaviour as can be observed in flocks of birds or schools of fish. When we also give the units a goal they will move toward the goal together. The big drawback of this approach is that the units act based on local information which easily gets them stuck in cluttered environments. Also, the combined steering behaviour can easily lead to the group breaking up.

Another widely used technique is grid searching in which the environment is divided into a grid that can be searched for a free path using A* like approaches [17]. Different units try to find a path through the grid while avoiding collisions with each other. This easily leads to units getting stuck in ways that can only be resolved by rather unnatural motions (or cheating like penetrating the walls).

The social potential field technique [14] defines potential force fields between units of the group. Desired behaviour is then created by defining the correct force fields. However, the same problem as in flocking arises because only local information is taken into account.

Kamphuis and Overmars [9] developed a method for planning the motion of a coherent group of units using a multiphase algorithm. First, a path is planned for a deformable rectangle, representing the group shape. Second, the internal motion of the units inside this deformable rectangle is calculated using social potential fields. Third, the global and local paths are combined to give the total motion of the units. Although the technique guarantees coherence, it lacks completeness. The approach also generates unnatural behaviour when a group enters or leaves a narrow passage.

Bayazit, Lien and Amato [4] have combined the probabilistic roadmap approach (PRM) approach with flocking techniques. The units use the roadmap created by PRM to guide their motion toward the goal while they use flocking to act as a group and avoid local collisions. While this indeed leads to better goal finding abilities, groups still split up easily.

Li and Chou [12] developed an approach that allows dynamic structuring of the units such that the centralized planning of the motions is greatly improved. Again, this approach lacks the ability of guaranteeing coherence.

Crowd simulation also investigates the movement of large numbers of units in a virtual environment. This research area has received vast amounts of attention over the last few years, such as [13],[20]. Although related to our research, the area has a different goal. The global idea behind crowd simulation is to have virtual units behave in a natural way, interacting with each other, based on (social) rules. The emergent behaviour of the units is then studied.

Other work in this area is by Feurtey [6], who uses a space-time approach to predict collisions with other actors, Helbing and Molnar [8], who use a social force model to simulate movement based

on motivations, Blue and Adler [5], who use a cellular automata model, Gillies and Dodgson [7], who concentrated on obstacle avoidance and the simulation of attention, and Lamarche and Donikian[10], whose work on path finding also includes ideas on behaviour simulation.

Other recent work done by Rymill and Dodgson [18] simulating human behaviour in crowds in real-time, concentrating particularly on collision avoidance. The algorithms used are based heavily on psychology research and their approach gives better results than conventional methods.

Shao and Terzopoulos [19] address the difficult open problem of emulating the rich complexity of real pedestrians in urban environments. Their artificial life approach integrates motor, perceptual, behavioral, and cognitive components within a model of pedestrians as individuals. They 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

3. The problem of path finding

Path planning consists in finding an optimal path (generally the shortest one) between a starting point and a destination point in a virtual environment, avoiding obstacles. Traditionally, path planning has been solved using a heuristic search algorithm such as A* [1],[3] directly coupled with the low-level animation of the agent. The use of A* for path planning is based on a two step process. The virtual environment is first discretised to produce a grid of cells. This grid is formally equivalent to a connectivity tree of branching factor eight, as each cell in the discretised environment has eight neighbours. Searching this connectivity tree with A* using a distance based heuristic (Euclidean distance or Manhattan distance) produces the shortest path to the destination point. This path is calculated offline, as A* is not a real-time algorithm, and the agent is subsequently animated along this path. As a consequence, this method cannot be applied to dynamic environments. This direct integration of A* with low-level animation primitives is faced with a number of limitations.

In order to let an object or character move inside a scene from one location to another, a path has to be planned that guarantees a collision-free translation from the start to the goal position. Hence, the whole task of path planning is usually broken down into three sub-problems:

- First, one has to find a suitable discretization of the ground on which one can build a graph. This can be done offline in a pre-processing step. The resulting graph should be as lean as possible to allow a fast search. If the graph is too large, the search will be significantly slowed down. One the other hand, the discretization should be as fine as possible so that the areas corresponding to graph nodes are not too large. This would lead to an approximation error which ends up in suboptimal paths.
- Then, the graph has to be searched for a solution which connects the found nodes. For static environments as expected, the A* algorithm is commonly used.

Afterwards, the resulting sequence of graph nodes needs to be transferred back to the original environment.

4. A* Algorithm

The standard search algorithm for the shortest path problem in a graph is A*. It is a directed breadth-first search and combines the advantages of uniform-cost and greedy searches using a fitness function: $f(n) = g(n) + h(n)$;

Where $g(n)$ denotes the accumulated cost from the start node to node n and $h(n)$ is a heuristic estimation of the remaining cost to get from node n to the goal node.

During the search, the A* algorithm maintains two lists of nodes: The *open list* contains the nodes that have to be considered next and the *closed list* which contains the nodes already visited. The algorithm itself consists of expanding the one node from the open list whose fitness function is

minimal. Expanding a node means putting it into the closed list and inserting the neighbours into the open list and evaluating the fitness function. The algorithm stops, when the goal node gets expanded.

The choice of a good heuristic is necessary in order to achieve both quality and efficiency of the search. As long as the heuristic underestimates the real cost, the shortest path is guaranteed to be found. Nevertheless, underestimating can easily lead to an expansion of too many nodes. But when the heuristic is allowed to overestimate the remaining cost, faster results can be achieved because fewer nodes get expanded. If overestimating the distance to the goal, the A* algorithm tends to expand nodes that lie on the direct path to the goal before trying other nodes. But this can also lead to significantly slower searches if the final path contains directions that lead away from the goal [11].

5. System Overview

The following section discusses various aspects of our solution. First, we present how to create the virtual environment and how to discretizate the scene into cells in order to form a graph. Second, we present our modified algorithm of A* in order to be used in dynamic environment. This algorithm is used by every individual inside its visible region. Then we focus our discussion on collision avoidance types and situations. The individuals must avoid collisions with the environment and with each other.

Given a pedestrian's current location and a target destination, it exploits the topological map at the top level of the environment model. By applying path search algorithms within the path maps associated with each region, the pedestrian can plan a path from the current location to the boundary or portal between the current region and the next. The process is repeated in the next region, and so on, until it terminates at the target location.

5.1 Scene Modelisation

In our problem setting we are given a virtual environment in which individuals must move from a given start point to a given goal position. The information to be given are:

- The number and position of the obstacles in order to form the desired environment.
- The number of agents, the position and the goal point for each agent. More than one agent could have the same goal position

5.2 Dicretization

Our approach is a cell decomposition approach which used a modified A* to find paths for a set of agents. The first task is to discretize the scene into obstacle-free regions. Each agent occupied one cell but an obstacle can occupy several cells in the environment

5.3 Path finding

Autonomous pedestrians are capable of automatically planning paths around static and dynamic obstacles in the virtual environment. After the discretization of the scene, each agent computes its path by applying the A* algorithm in its visible region. The boundary or portal between the current region and the next becomes the intermediate target location. At this stage the algorithm takes into account only the static obstacles.

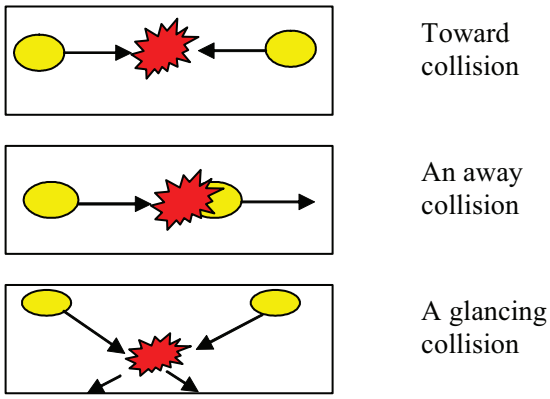
The next stage of the simulation is the pedestrian's movement. So, for each frame of the behavioural animation and before getting to the next position, each agent must process the collision prediction to avoid collision with the other agents of the environment.

5.4 Collision prediction

In each frame of the simulation, every agent needs to check for future collisions with all other agents in the scene. If a collision has been predicted the type of collision must be determined. In real life, there are three possible types of collision, called towards, away and glancing; there shown in figure 1.

- **Toward collision :** or face to face, occurs if the agents are walking toward each other;
- **An away collision:** or rear, when the agent is behind the collidee (an other agent or an obstacle);
- **A glancing collision:** is a side-on collision between two agents walking in roughly the same direction;

Figure 1: The three collision types



5.5 Collision Avoidance

Collision avoidance between agents can involve some problems that only appear when we deal with many agents. A method to avoid collision between individuals can be not efficient when we have several ones. There are more constraints and variables when a complex environment (with fixed obstacles, mobile obstacles, and small regions to walk) includes many virtual human agents. Yet, if the structure of the group has to be preserved, this adds another parameter in the complexity of crowd collision avoidance.

First, consider how we avoid collisions in our human life for collision avoidance. It is very complex but it can be defined by some simple rules. In general, one is reluctant to be far away from one's path. Therefore, one just goes ahead if the other goes out of way to avoid ahead-on collision. In case of overtaking, one prefers a wide side or follows the other if obstacles (or other characters) exist somewhere near. In case that a collision occurs while proceeding to different direction, people pass on the backside or speed up; otherwise speed down or wait generally [2].

5.5.1 Towards Collisions

The first stage is to determine whether the collidee is to the left or right of the agent. People will prefer to pass on the side with least deviation from their path. Observations show that the agent has three different ways of avoiding the collision:

- Changing direction only;
- Changing speed only;
- Changing direction and speed;

If no behaviour has been found that will avoid collision, the agent simply stops walking. This will allow the other agent involved in the collision to avoid the subject, who can then resume walking.

5.5.2 Away Collisions

An away collision is one where the collidee is in front of the agent, but the agent is walking faster than the collidee, so will bump into the rear of the collidee. To deal with this situation, the agent has two choices:

- Slow down to the same speed as the collidee and walk behind it.

- Walk faster and overtake the collidee by choosing the appropriate side.

5.5.3 Glancing Collisions

This type of collision is dealt with a similar way to towards collisions.

Each agent has a goal trying to reach it by following its initial path. After avoiding collision, the agent should return to its path smoothly in order to look natural or change the path completely.

6. Avoidance behaviours

As we have seen before, each collision avoidance needs different behaviours and different treatment. The list of behaviours that can be used in avoidance collisions are:

- moving forward,
- Changing directions (left or right)
- Waiting
- Speeding up
- Slowing down
- Moving back.

Each agent has a priority and several ones could have the same priority. These priorities are taken from the sociological and psychological of the human society (the priority is given to the old human, a handicap, a pregnant women, etc…).

A second level of priority is given to the behaviours. It is complex phenomena because it depends on the type of collision and the avoidance collision situations: (two individuals, crossing groups, queuing in an exit, queuing in two directions, crowded environment, etc…).

The process of collision avoidance is similar for all these situations by applying the above algorithm for each pair of agents in collisions. The process is described in the figure3. The overview of the system is described figure 2.

Figure 2: Overview of the system

```
For each agent apply A* to find its itinerary.
For each frame of animation do
        For each agent do
          Collision prediction
          If  not collision then
                the agent goes on its way.
          Else
                    Apply the collision avoidance agent to
                    agent algorithm.
            End if
     End for
End for
```

Figure 3 Agent to agent collision avoidance algorithm

For each pair of agents **do**
If toward collisions **then**
If agents have the same priorities **then** Choose one agent in random way
If collision 1 then (two agents reach same cell figure 4)
- Agent1 move forward, agent 2 change direction to the right if there is space to move
else change direction to the left
- Agent 1 change direction to the right, agent 2 move forward
- Agent 1 change direction to the left, agent 2 move forward
- Agent 1 move forward, agent 2 wait.

Else (**collision 2**agents reach the cell of each other figure 4)
- Agent1 move forward, agent 2 change direction to the right if there is space to move else change direction to the left
- Agent 1 change direction to the right, agent 2 move forward
- Agent 1 change direction to the left, agent 2 move forward
- Agent 1 wait, agent 2 wait (these agents are blocked)

End if
Else begin with the agent witch has the higher priority and process the treatment as in the same priorities.
End if
Else if away collisions **then** (agent 2 behind agent 1)
- agent 1 move forward, agent 2 change directions right, left to overtake, or slowing down

else (glancing collisions)
- Agent1 move forward, agent 2 change direction to the right if there is space to move else change direction to the left
- Agent 1 change direction to the right, agent 2 move forward
- Agent 1 change direction to the left, agent 2 move forward
- Agent 1 move forward, agent 2 wait.

End if
End if
End for

We can summarize the treatment of collision avoidance by using the priority rules in the table1. For example we pass to the priority two if there is no space to move.

Table 1: Priority rules in collision avoidance

	Collision type	Agent1	Agent2	Priority
towards collision	Collision1	forward	right	1
		forward	left	2
		right	forward	3
		left	forward	4
		forward	wait	5
	Collision2	forward	right	1
		forward	left	2
		right	forward	3
		left	Forward	4
		wait	wait	5
Away collision	Collision2	forward	Overtake by right	1
		forward	Overtake by left	2
		forward	Slowing down	3
Glancing collision	Collision1	forward	right	1
		forward	left	2
		right	forward	3
		left	forward	4
		forward	wait	5

6.1 Crossing groups

This situation is found in a crowded environment, when two groups of agents moving in opposite directions and try to avoid each other. In real life, they formed opposite lines consisting of pedestrians with the same direction. (figure 5.1)

In case of collision the agent to agent collision avoidance algorithm is applied. The collision avoidance behaviours used are: moving forward, changing directions, waiting and moving back. The last behaviour is used in the case of blocking when there is no space to move forward.

6.2 Bottlenecks (queuing in an exit)

Bottlenecks or passing direction of pedestrians is found in applications such as the entrance into corridors, staircases, subways, or doors. In real life, the priority is given to the nearest to the centre of the bottleneck. The A* algorithm resolves directly this situation and in case of collision the agent to agent collision avoidance algorithm is applied. The collision avoidance behaviours used are: moving forward, changing directions, and waiting. (figure 5.2)

6.3 Queuing in two directions

Pedestrians form queues in front of exits or doors of vehicles. When a vehicle arrived, pedestrians wait for the inside passengers to get off the vehicle and then get on it. This type of movement can be seen at elevator halls, platforms of railway stations, bus stops, and so on. (Figure 5.3)

There will be two types of situations: pedestrians gather in front of the entrance without leaving the way for inside passengers to get off the vehicle and pedestrians leave the way for inside passengers to get off the vehicle.

In our system a priority is given to the agents of one direction, the others wait until there will be a free space to move. In case of collision the agent to agent collision avoidance algorithm is applied. The collision avoidance behaviours used are: moving forward, changing directions, waiting and moving back. The last behaviour is used when passage is blocked; there is no space to move. The agents which have less priority have to choose between two possibilities:

- Moving back to let passage to the other agents.
- Or applying the A* to find a new path if it is possible.

6.4 Narrow passage

This situation is observed inside corridors, in pavements, or pedestrian's passage. Generally the pedestrians formed line segregation, each group takes a direction and the choice of the direction is taken from the sociological behaviours of the pedestrian (right or left). The A* algorithm provides the path to be followed with just changing direction to overtake in case of away collision. (figure 5.4)

Figure 5.1 *Crossing groups*

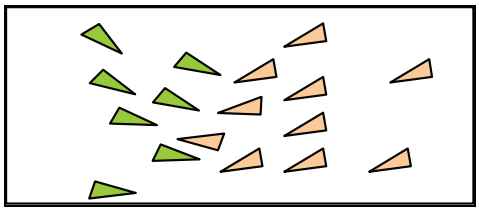


Figure 5.2 *Bottlenecks*

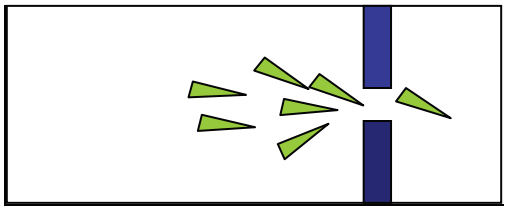


Figure 5.3 *Queuing in two directions*

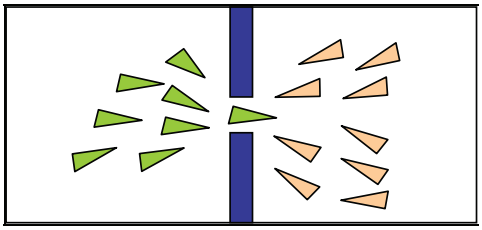
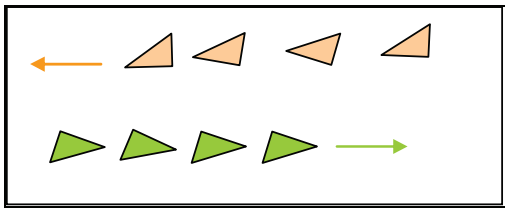


Figure 5.4 *Narrow passage*



7. Results and Discussion

The behavioural algorithms described above have been implemented in C++ using Open GL library. The system is a 3D software designed to be used in real time, so different ideas and situations can be simulated. A crowd of about 600 agents can be simulated at an acceptable time on a Pentium IV, 3 GHz with 256 Mo of main memory.

The user interface has been designed to allow easy testing: the simulation can be paused at any time, and replayed at any speed, or frame by frame, meaning that a collision can be viewed from a variety of angles. He can create his virtual scene in different ways, by placing the obstacles and the agents wherever he wants; He can also choose the destination of each agent. The system can simulate any type of collision situations:

An environment with agents and obstacles or without obstacles, a populated environment, a narrow space to produce a large number of potential collisions, agents acting as bottleneck, crossing groups, queuing in two directions,etc…

Figure 5 show some results of our system. The colored cells are the goals of the agents.

Our algorithm is used for all the collision avoidance situations, with minor changes in the priorities of the agents and the behaviors. We have separated them in order to better understand and to compare the results of the system with the real life figure 6.

The combination of A* with the application of the behaviors of collision avoidance gave good results similar to the everyday life. The agents moving back could use the A* to compute the new optimum path from the actual position to the destination target. After avoiding collision, the agent has two choices: to return to its itinerary or compute a new path to its goal from that position.

The use of priority rules solved the problem of disorder in the crowd movement. The traversed time depends on several parameters, amongst other things, the density of crowd and the methods of collision avoidance. We have tested the system with these priority rules and without them. And in

almost the situations the traversed time is better with the use of these rules. The table 3 and table 4 show this difference in two situations for example, narrow passage and queuing in two directions.

Table 2: Traversed time in seconds in narrow passage

Agents	With priority rules	Without priority rules
1	50	85
2	58	99
3	43	69
4	52	73
5	44	97
6	58	107
7	54	108
8	64	114
9	56	100
10	49	90

Table 3: Traversed time in seconds in queuing in two directions

Agents	With priority rules	Without priority rules
1	45	55
2	39	59
3	49	60
4	56	50
5	33	43
6	32	34
7	44	59
8	32	50
9	29	29
10	46	59
11	40	54
12	34	55

8. Conclusion and Future Works

This paper has presented useful ideas towards a system that can simulate human behavior based on theories from sociology and psychology of the human being. We concentrate our discussion on techniques for collision avoidance as well as the path finding in dynamic environment. These two techniques add realism to the simulation. The system can simulate a large number of agents in real time (about 640 agents).

The use of priority rules solved the problem of disorder in the crowd movement and in almost situations the traversed time is better with the use of these rules.

Our model could be used as a framework to simulate real situations such as: the rise and descent of a subway or a bus, the walk in pavement, the entry or the exit of a supermarket…

There are many ideas for future work to improve the realism of the simulation:

- Actually, all agents move through the scene on their own, further work would allow agents to move in small groups.
- Currently each agent is assigned a goal to move towards; a better system would involve the agent being given a plan rather than a simple goal. For example, an agent could be told to visit all the sites in a museum, get out of the scene etc...
- The notion of groups should improve the realism of the simulation by performing group avoidance rather than individual avoidance and will allow modelling both group and individual behaviours.

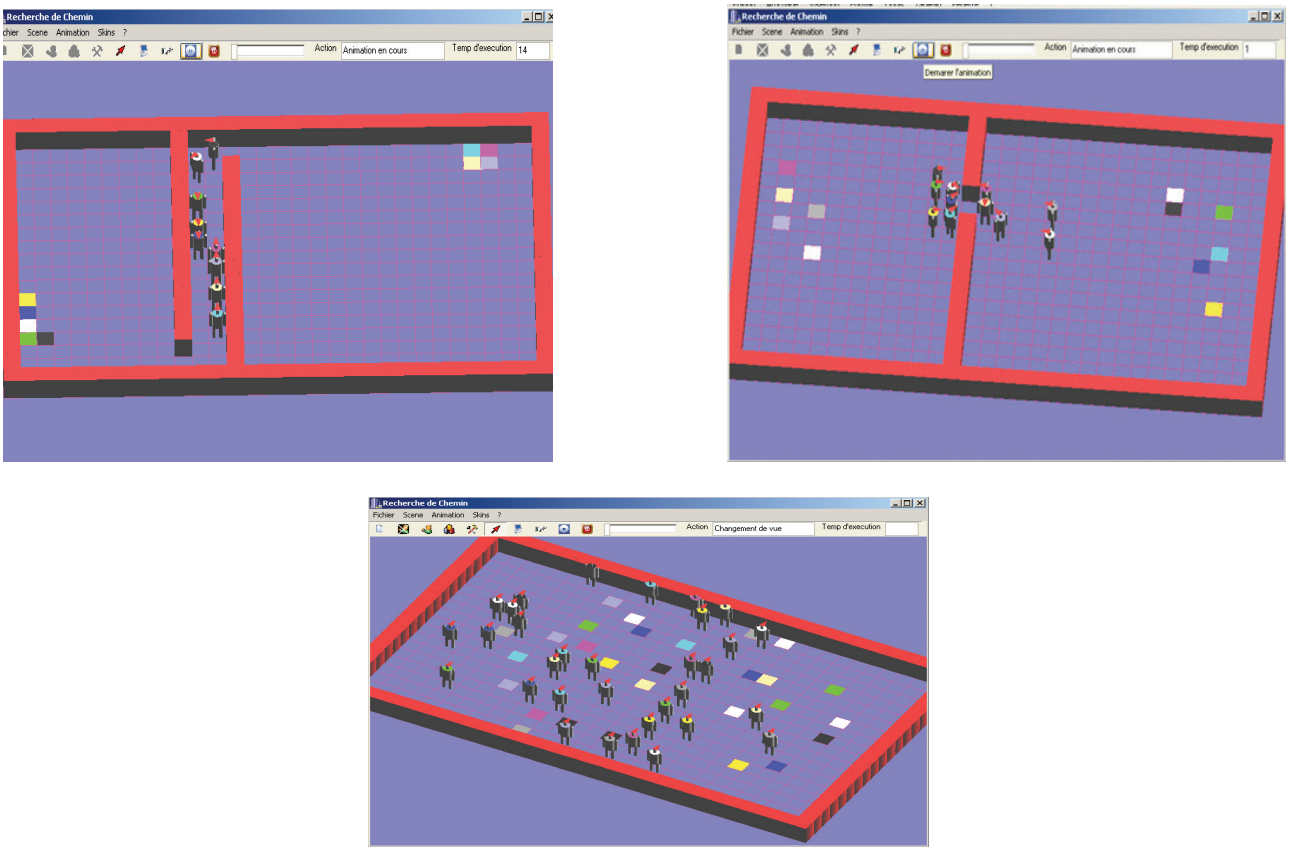


Figure 6: a) queuing in two directions; b) open area without obstacles c) narrow passage

References

[1] N. Badler, C. Phillips, and B. Webber, Simulating Humans: Computer Graphics Animation and Control. Oxford University Press, New York, NY, 1993.

[2] S. Baek, I. K. J. Inho, Lee Motion Generation Using Motion Mining System, International Conference on Artificial Reality and Telexistence ICAT2003, Japan, 2003.

[3] S. Bandi, and D. Thalmann, Space Discretization for Efficient Human Navigation.Computer Graphics Forum, 17(3), 1998, pp.195-206

[4] O. B. Bayazit, J. M. Lien, and N.M. Amato, Better flocking behaviors using rule-based roadmaps. In Algorithmic Foundations of Robotics V, Springer Tracts in Advanced Robotics 7,. Springer-Verlag Berlin Heidelberg, 2004, pp. 95–111

[5] V. Blue, J. Adler, Cellular automata model of emergent collective bi-directional pedestrian dynamics. In Artificial Life VI, the Seventh International Conference on the Simulation and Synthesis of Living Systems, 2000.

[6] F. Feurtey, Simulating the Collision Avoidance Behavior of Pedestrians. Master's thesis, Department of Electronic Engineering, University of Tokyo, 2000

[7] M. F. P.Gillies and N. A. Dodgson, Attention based obstacle avoidance for animated characters. In Virtual Reality. To appear

[8] D. Helbing and P. Molnr, Social force model for pedestrian dynamics. Physical Review, 51, 1995, pp. 4282– 4286

[9] A. Kamphuis and M. H, Overmars, Motion planning for coherent groups of entities. In IEEE Int. Conf. on Robotics and Automation. IEEE Press, San Diego, CA, 2004,

[10] F. Lamarche, and S. Donikian, Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. In Computer Graphics Forum, vol. 23, 2004, pp. 509–518

[11] C. Niederberger, D. Radovic, M. Gross, Generic Path Planning for Real-Time Applications, Computer Graphics International (CGI'04), 2004, pp. 299-306.

[12] T. Y. Li, and H. C. Chou, Motion planning for a crowd of robots. In International Conference on Robotics and Automation (ICRA). IEEE Press, San Diego, CA, 2003.

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

[14] J. Reif and H. Wang, Social potential fields: A distributed behavioral control for autonomous robots. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, International Workshop on Algorithmic Foundations of Robotics (WAFR), A. K. Peters, Wellesley, MA, 1995, pp. 431–459.

[15] C. W. Reynolds, Flocks, herds, and schools: A distributed behavioral model. Computer Graphics, 21(4), 1987, pp.25–34

[16] C. W. Reynolds, Steering behaviors for autonomous characters. In Game Developers Conference, 1999,

[17] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach. Prentice Hall, 1994.

[18] S. J. Rymill, and N.A. Dodgson, A Psychological-Based Simulation of Human Behaviour. Theory and Practice of Computer Graphics. EG UK, 2005, pp 229-236.

[19] W. Shao and D. Terzopoulos, Autonomous Pedestrians. Eurographics / ACM SIGGRAPH Symposium on Computer Animation. USA. 2005, pp. 19-28.

[20] B. Ulicny and D. Thalmann, Crowd simulation for interactive virtual environments and vr training systems.