Group Simulation of Three Agents using Unity3D

Vincent Wong vwong@kth.se
Max Turpeinen maxtu@kth.se

KTH Royal Institute of Technology | Supervisor: Christopher Peters

Link to our blog: http://www.crowdsimulationkth.blogspot.se/

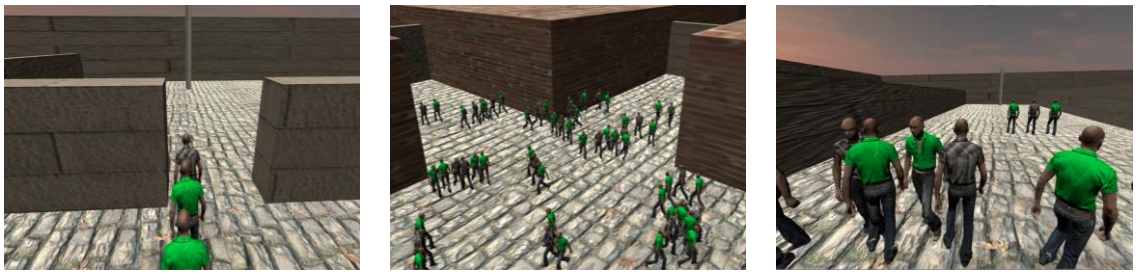

Figure 1: Screen captures from our scene with the latest implementation of our model.

Abstract

Crowd simulation refers to simulating a large number of agents trying to replicate collective behavior in 3D computer graphics. For this, we have been using the game engine Unity 3D. In our work we are mainly focusing on group formations consisting of 3 agents. Each group is assigned a leader which keeps track of formations and evading collisions. The groups can take on four different formations, working like a finite state machine. In our general scenario, two groups could face, making it useful to adapt the formations and movement direction to avoid each other.

We have been implementing our own model after having looked at different major concept already existing in the world of crowd simulation. You could define our model as a hybrid rule-based one, seeing that we have a smaller group working in a way like a single entity, making its judgment dependent on checks with corresponding rules.

1. Introduction

Crowd simulation is all about simulating a large number of entities, also known as agents as a greater group, instead of individuals. The general technique is needed when the number of characters exceeds the given tolerated computer performance. The idea is to make the software cost efficient which in many cases requested. The concept is applied in Virtual cinematography reflecting for most parts real human behavior in one or several groups interacting. The major industry lays in making films and games using 3D computer graphics but is also used for public safety planning. During e.g. sporting and general spectator occasions crowd controlling strategies can be made beforehand using crowd simulation. This is done to prevent accidents during emergencies. Generally, crowd simulations are based off of two

different categories; microscopic models and macroscopic models (discussed in the section "2. Related work"). There are several software of crowd simulation, but none (what we could find) so far specified on three agents. That is what we have been focusing on with our own rule-based algorithm (which can be seen in "3. Implementation"). We have also been discussing the different techniques as well as our own implementation (in section "4. Result & Discussion") to finally reach our conclusion ("5. Conclusion").

2. Related work

Throughout the years several different techniques to model agents forming crowds have been proposed. Especially now seeing the immense growth of interest in areas such as movies and computer games. The concept is to compute a large number of agent's paths towards shared or different goals, while simultaneously avoiding collision in between each other as well as other obstacles. Not to forget, replicating the human behavior. The problem in itself can be seen in a broader scale using global planning as well as local behavior. It all can be traced way back to the works of Reynolds [2] who proposed "Boids" which is implementing simple rules commonly as forces to avoid collision while preventing flock cohesion. Methods such as seek flee and pursue were later added. The rule-based approach became very popular and created the basis of what today is known as crowd simulation.

The subject has also been looked at from many aspects and viewpoint whereas in general, the different methods can be divided in to two groups; microscopic & macroscopic. The two categories focus on crowd simulation on two different levels, which results in different detail-of-level which also cause different features.

2.1. Microscopic

The microscopic [4] approach focuses on the issues on an agent level, not taking the group as a whole in to account. Usual implemented aspects for the pedestrian in such approach is psychological and social behavior. The Individual agents thereby make their own decisions as well as sending information in between each other. A distinguishing feature for a microscopic model is its accuracy and its high resolution, since it describes how each pedestrian behaves in every single step. However, as a result, the cost in computation complexity is very high. Increasing the scale of the crowd, the approach can no longer sustain its efficiency and smoothness.

There are lots of approaches within this category; Cellular Automata, agent-based, particle system, social force and rule based etc.

2.1.1. Cellular Automata

Also known as "CA" [5] is one of the earlier approaches doing crowd simulations. This approach can be seen as having a navigation mesh divided into identical cells using a discrete function and having a state for each cell. The state of each cell is based on a set of local rules. The state of a grid cell changes between zero and one indicating whether or not a cell is occupied by an agent or not. The agents then move towards its goal only able to move in between adjacent grid cells.

2.1.2. Particle Systems

Particle systems [6] were initially made to render fuzzy objects like smoke and fire, though the idea was also applicable to the area of crowd simulation. Here, the pedestrian are defined in a more vague sense seeing them as mere objects. Like in other microscopic models the behavior is rule based, but can be based off of various algorithms that define the behavior or the necessary path calculation.

2.2. Macroscopic

On the contrary to the microscopic approach, the macroscopic [4] approach cannot bring the same high-resolution result for every

pedestrian, nor does it use the social and psychological elements of each pedestrian. In other words, the approach contains overall less information, which makes it more durable for large-scale crowds and highly concentrated populations. This is due to the fact that it takes the crowd as a whole and looks more on how the environment may affect the behavior of crowd. Local interaction in between pedestrians in the crowd is not taken in to account when it comes to macroscopic approach. One of the more known models of this type is the flow-based model as well as the fluid model. A general way of looking at crowd in the macroscopic sense is using partial differential equations to describe the change in time for the relationship between density flow and velocity of a crowd.

2.2.1. Fluids

As mentioned earlier, one of the greater approaches in the macroscopic category is the fluid [7] one. This approach has the idea of assigning the large group into different areas, also known as fields. Depending on the specific approach we can have several fields with the corresponding aspect, such as density fields, velocity fields, and dynamic fields where e.g. dynamic fields are used to handle obstacles. Since the crowd is seen either as a fluid or a gas, physical laws of fluids dynamics are of great value doing the modelling. Important to note is that the group should share the same goal.

2.2.2. Chaos Models

The chaos model refers to a crowd with a chaotic behavior known to be implemented by Saiwaki et al., 1997 [8]. The agents work on their own, but what makes them a uniform group is that each agent is share the same set of parameters and rules. This might be seen as a microscopic approach but the focus lies in the group, and not the individual agent. By changing the few existing parameters and rules for all of them, very different and interesting behavior can be observed. This make the model and its agents more controllable, and just a plane visualization of agents moving by random numbers generated.

2.3. Hybrids

Hybrids, as well in this case as in many others, refer to a combination of two commonly used implementations. When talking about crowd simulation, the combination is about the microscopic- and the macroscopic approach. Many different combinations of the two models have been proposed trying to unite the advantages of each with various results.

An example of a hybrid technique is the one implemented by Narain R. et al. 2009 [3], where they use a dual representation of the microscopic- and the macroscopic model. Here the idea is to use discrete agents along with a single continuous system where the continuous setting steers the large-scale behavior of the crowd as well as avoidance between the agents in dense scenarios.

2.4. Pedestrian

In a crowd simulation the pedestrians also play a part. What is the most common thought is that the pedestrian should fit your environment, though what is also worth mentioning is the detail of it. The cost between simulating a group of high resolution 3D characters and a group of low resolution 3D characters can be of great importance.

A solution to this problem is to use imposters [9]. Imposters is a way of rendering a 3D character in 2D and placing it in front of the real 3D one, which takes less time than actually rendering the real 3D one. This is commonly rendering background objects, but can also be used in the case of crowd simulation, implementing it on pedestrians either meant to be far away or not placed in center (not meant to have focus).

3. Implementation (model)

In general, every crowd simulation consists of a model of an agent, as well as an implementation of an algorithm. In this

project the main focus has been on the approach of controlling and steering crowds, both on group and individual level, i.e. the hybrid model.

3.1. Formations

A small group of pedestrians usually strive to walk in a line formation, if possible (a way of socializing). The formation however changes or in so cases splits up to avoid obstacles [10], the choice of avoidance depends on the obstacle and the size of the group, the bigger the group is, the more likely of a split.

To begin with a simple correlation is needed between the agents in a group. By taking in consideration of [10] this project mainly focuses on groups consisting of three agents with 4 different formations to change between. The implemented approach was having a leader placed in the center and the agents adjacently positioned on each side of the leader. This forms our standard formation, also known as line formation. There are also three other formations which can be seen in Figure 2. Each formation is predefined and static with a list of 3D vectors. The leader's formation position is always at (0,0,0) with the followers position adjusted according to the leader.

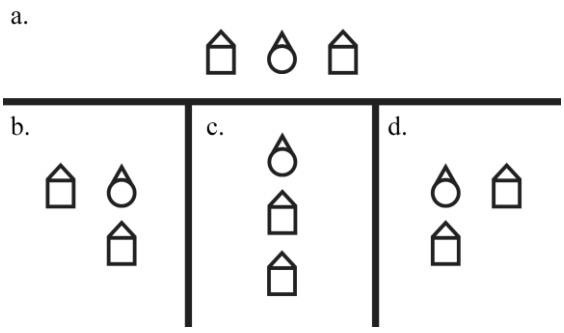

Figure 2. a. Line Formation, b. Left Formation, c. Straight Formation, d. Right Formation.

3.2. Goal

Either by generating a goal position manually or randomly, a 3D-coordinat position is placed for a group and its leader. The goal point can be seen as the leaders target position, the group rotates towards the goal while translating, creating a nice and beautiful spline. There is also a fixed tolerated radius, if either one of the agents or the leader reaches the inside of the area, the group disappears since they have reached their goal.

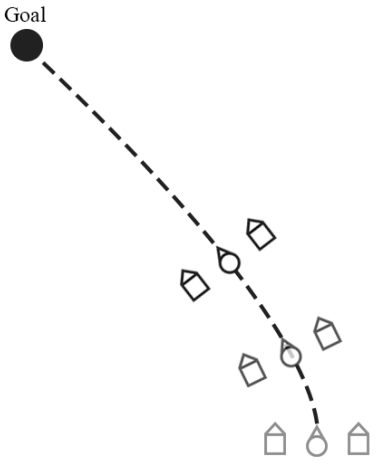

Figure 3. The direction is towards the goal but the rotation is time dependent, giving the group a more natural way of moving.

3.3. Position Holding

3.3.1. Distance to Position

The agents in each group are dependent on the leader in such way that the leader generates a formation position for each agent. This position is for e.g. the line formation two adjacently placed 3D-coordinate. Agents will always try to hold the formation and in case an agent should fractionate it will adjust its speed to maintain the given formation. This happens e.g. when followers move to fast/slow compared to the leader (Figure 4.).

Figure 4. If d_i is greater or less than x the agent will have to decrease or increase the speed accordingly.

3.3.2. Rotation of Group

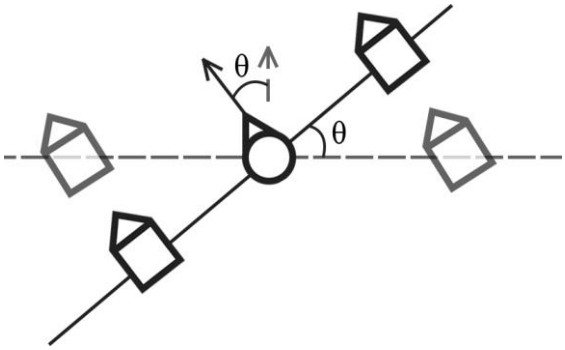

Figure 5. The solid agents are where the agents are supposed to be.

One problem that occurs while holding the formation is when the group is supposed to rotate as seen above (Figure 5). In order to keep the formation during rotation the scalar product of the direction from the leader towards the goal V_G and the adjacent right V_R of the leader will return a positive value if the rotation is to the right, and if we get a negative outcome, the rotation is to the left. We now know which agent needs to slow down and which to speed up in order to maintain the formation. The speed of each agent is randomized from an adapted allowed interval, which gives a realistic looking behavior to the group.

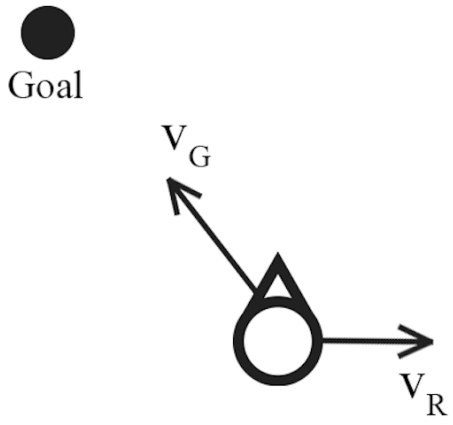

Figure 6. V_G is the direction to the goal; V_R is the leader's right vector.

3.4. Avoidance

To avoid collisions and other hindrance's e.g. other group of agent or obstacles; the FSM [11] (Finite state machine) model is implemented. The finite-state machine is a mathematical model, conceived as an abstract machine that can be in only one of a finite number of states at a time. It changes from one state to another state when triggered. There is a number rays for each set of group which can be seen in Figure 7. These rays check the area in front of the group and depending on the combinations of hit the state changes accordingly.

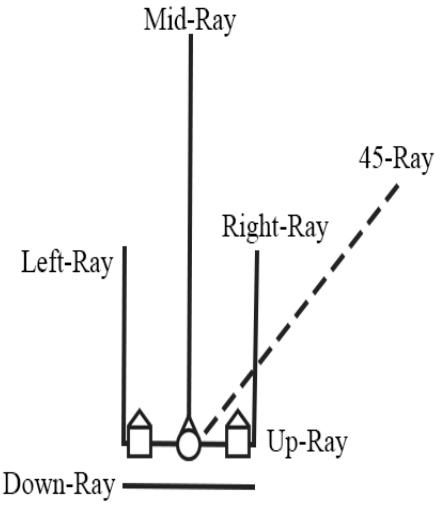

Figure 7. The solid ones checks for collision with obstacles and agents, whereas the dotted one only detects agents.

3.4.1. Avoidance of Obstacles

There are in total five rays that are used to detect obstacles as can be seen on Figure 7. These rays follow the group and always hold the standard formation i.e. they will not change position even if the group changes formation. One can say that it is static to the group. The left-, right-, up- and down-rays (basic rays) holds up to three different states; these states depend on the combination of hits. Whereas the mid-ray with double the length to the side-Rays, creates temporary rays on each side of mid-ray with θ (15°) shifting. Whenever one of these temporary rays does not detect any obstacle, that point will be the temporary goal position and if detection occurs, another 15° will be added. The algorithm can be seen below (Algorithm 1.).

```
If Mid-Ray is true
    θ+=15
    Create temporary side-rays, check sides of Mid-Ray with θ
    If one of side-rays is false
        Temporary goal position is found
If Up-Ray or Down-Ray  is true
    Hold previous state
If Left-Ray & Right-Ray is true
    Change formation to Straight Formation
If Right-Ray is true
    Change formation to Left Formation
If Left-Ray is true
    Change formation to Right Formation
```

Algorithm 1. Avoids obstacles

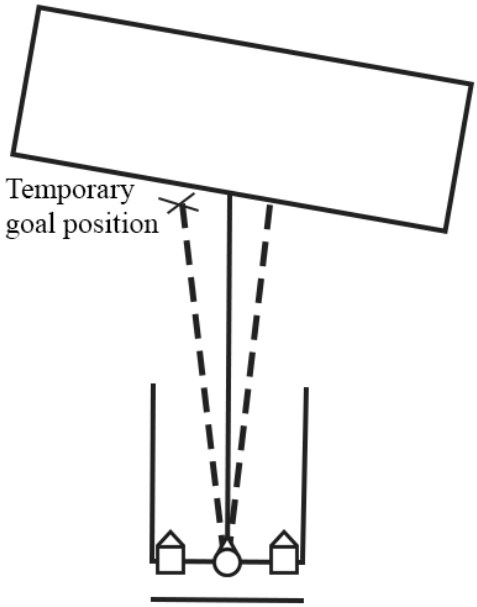

Figure 8. In this scenario, the group will rotate and pass the obstacle to the left.

3.4.2. Avoidance of Agents

The same concept used for obstacle avoidance is applied for agents as well. With the basic rays doing the same thing for agents as it does for obstacles. However for agents, the Mid-Ray does cast two new temporary rays upon detection, instead it checks for the other group's direction. If the other group is moving towards the group, both group will turns to the right (similar to Swedish pedestrian traffic). Then there are cases when the other group in front is moving with lower speed, the group behind will then adjust its speed to the same as the group in front.

```
If 45-Ray is true
    If α is 42-47° and hit.Speed=this.Speed
        Slow down
If Mid-Ray is true
    If α >170°
        Move to the right
    Else if (α <10° and hit.Speed<this.Speed)
        Slow down
If Left-Ray & Right-Ray is true
    If α>170° & distance<x
        Change formation to Straight Formation
    Else if (α <15° & (hit.Speed < this.Speed or distance<x*1/3))
        Slow down
If Right-Ray is true
    If α >170° & distance<x
        Change formation to Left Formation
    Else if (α <15° & (hit.Speed < this.Speed or distance<x*1/3))
        Slow down
If Left-Ray is true
    If α >170° & distance<x
        Change formation to Right Formation
    Else if (α <15° & (hit.Speed < this.Speed or distance<x*1/3))
        Slow down
```

Algorithm 2. Avoids agents.

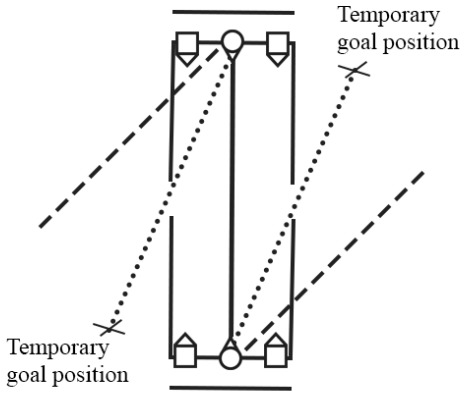

Figure 9. A representation of the so called "right rule" to avoid collision.

With the addition the 45-ray collision from the right side will be avoided if the detected group is moving towards the detectors with 90° (Figure 10.) the detectors will slow down.

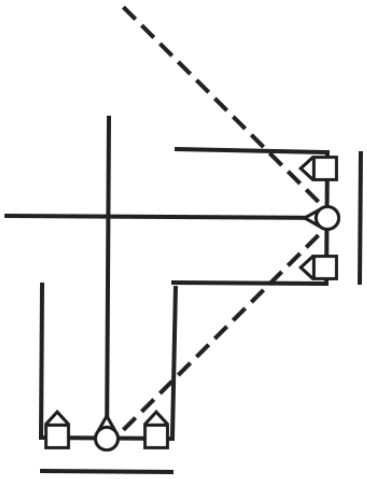

Figure 10. A 45° check to avoid following scenario.

4. Result & Discussion

4.1. In General

Typical approaches either resemble particle simulations where agents usually lack their own control of orientation, or as a more conservative approach, having agents lack the psychological state, not being able to be pushed. Others as social-force models tend to look more like particle simulations than realistic human motion. Cellular Automata-models limit agent spatial movements and tend to fail when dealing with larger dense crowds. Therefore it is also often an excluded approach in computer games. Finally, rule-based models, like the one we have implemented, either do not consider collision detection at all or malfunctions when trying to adapt to slowing down and waiting rules, which works fine, until you reach the state of having a high-density crowd.

Human usually have a great sense of direction and knows where they can walk and where not to walk e.g. walking along a wall before proceeding to rotate towards the goal. This is where our implementation fails, it only moves in two dimensions and it always tries to rotate towards the goal. We simply did not focus on translation at all since this project main focus was on groups of three's behavior. But there are a few methods to solve this problem, one solution would be to use navigation mesh, a mesh where the pedestrians are allowed to walk, this does not only solve the rotation towards goal problem but also translation in 3D. A good example of a navigation mesh solution with pathfinding would be A* Pathfinding algorithm [13].

Another major issue is that in most cases you require a human crowd simulated in real time with preferably higher level of detail and an accurate realism in behavior. This obviously points out the correlation between the accuracy of realism in the behavior and the computational cost of the simulation. To satisfy both at the same time is in particular a challenge of major importance. The state of the art is to reduce the computational cost while maintaining the high level detail of simulation.

4.2. Our Implementation

Seeing all the related works there several possible approaches to be implemented, with the choice of different hybrid combinations. In our work we have implemented our own algorithm mainly based off of the rule-base model. To note, is that ours differs from previous ones since we are focusing on three agents for each group.

Since we have built our own model and algorithm for groups of three, we have had to confine the ways possible for a group to act. In other words, our model cannot take every exception in to account. Especially when seeing all the other models in related work being far from optimal.

To begin with, we added the object oriented essence with each agent having a physical state. This comprises of attributes as position direction, speed, (size). This felt like the most intuitive approach doing it all from scratch. To have a macroscopic approach like a fluid one for only three agents each felt excessive.

Instead, we let the actual leader be a leader, having the other two agents following him and his lead. The leader alongside the implemented algorithm is in other words the one calling the shots. The attached FSM and the leader then keep track of the current formation, and eventual upcoming changes. Our approach can in a sense be seen as a hybrid, not fully have separated agents walking on their own, but nor having the leader fully calling the shots with just two agents attached.

When walking in groups of three, the most common formation to form is on a straight horizontal line. Therefore we have made the line formation our standard one. For most parts the person in the middle might be placed a step behind the two others, making it possible to create eye contact with each and everyone in the group. In our case, we have named the middle agent "the leader", and therefor do not place him behind the other, will instead, seeing it as the others following the leader. This becomes much more clear in the straight formation were the leader leads the group.

When deciding on the different formations, the simpler and more natural ones come to mind. When facing obstacles such as agents or blocks, what is the most realistic move to make? Walking on a horizontal line, as well as having someone move to the back when interfering with something on either side, makes the most sense. Also, the group should be able to pass a narrow road, with the only possibility to form a straight vertical line. This gives us the few most useful formations.

To solely walk in a straight line forward does not take much, but when initially implementing the rotation we tried to solve it analytically as can be seen in Figure 11. The problem is that the angle is very small for one frame which makes it almost negligible. We tried to compensate for the small magnitude of the angle by adding a scalar factor, but zero times a million is still zero. We are also always one step behind with this approach. Instead, we went with the scalar product approach (as can be read on 3.3.2 Rotation of Group)

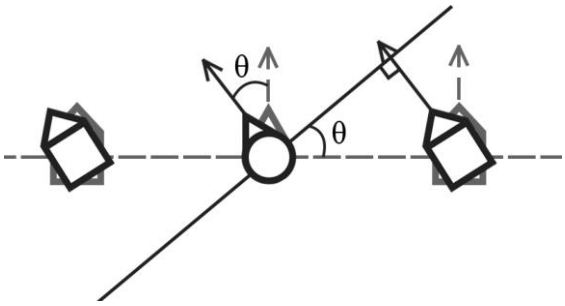

Figure 11. Analytic solution to 3.3.2

In some cases, like the one discussed by Akker et al. 2010 [12], some approaches can be seen as NP-hard when dealing with a dynamic multi-commodity flow problem. This means we have to find better solutions for the crowd simulation problem, where Moore's Law can only help us so much.

A thing to notice in crowd simulations is often a common way for groups to be repetitive. Like in our algorithm, we simplified the execution for our group when interfering with other groups of agents. When facing a total collision our group always turns to the right to find an opening. This recurrent behavior can be seen in other approaches as well, especially the rule-based ones. Adding a stochastic factor in situations like total collision gives the model a more human-like behavior. The possibility to check on side of the obstacle would also work, thinking that there is an optimal way for humans to walk, but what happens if both sides are equally long?

4.3. Final Scenarios

The final obtained result proved to be better than expected, with relatively high polygon 3D characters and multiple groups checking for avoidance simultaneously. The simulation did not lag and did run quite well regardless of the number of entities.

Figure 12. Final scene, avoiding obstacles.

The algorithm did works well in scenarios with low density, it avoids both obstacle and agents by changing formations (Figure 12), adjusting speed or changing directions. However when the density increases by a third errors starts to occur as can be seen in Figure 13, with an agent walking into the wall.

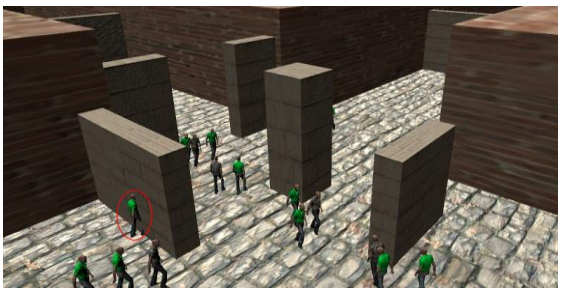

Figure 13. An agent stuck in the wall marked with red.

5. Conclusion

In general, which we have seen throughout the project (and which we have also discussed in the "Result & Discussion"-section), are that some models and approaches are better than others at doing different things. When wishing for high resolution you might not be able to have as many agents or when having a huge number of agents it might not be possible for them to act like individuals. If you decide upon either having the highest possible resolution for a single agent or having as many moderate agents as possible, a suited approach can be given to you, but having them both is what we would call the state of the art.

For future work regarding our current rule-based model, in theory unlimited checks and conditions could be implemented to solve all problems that may occur. Needless to say, theory and practice does not always reflect each other. Instead, the focus lies on trying to optimize the algorithm, only including the most crucial checks allowing the majority of scenarios to be handled in a realistic way. What is not to be forgotten is that simulations are meant to be a cheaper alternative to reality, making its realism the key component.

6. References

[1] Rojas F., Tarnogol F., Yang H.S.: Dynamic social formations of pedestrian groups navigating and using public transportation in a virtual city. KAIST A.I. & Media Lab., Daejeon, Republic of Korea. PsyTech LLC, Buenos Aires, Argentina (2015)

[2] REYNOLDS C.: Flocks, herds, and schools: A distributed behavior model. In Proceedings of ACM SIGGRAPH 87, Annual Conference Series. (1987)

[3] Narain R., Golas A., Curtis S., Lin M.C.: Aggregate Dynamics for Dense Crowd Simulation. Novel, UNC, North Carolina, United States (2009)

[4] Saad A., Nishino K., Manocha D., Shah M.: Modeling, Simulation and Visual Analysis of Crowds: A Multidisciplinary Perspective, chapter 3, p. 44 (2013)

[5] Tasse F. P.: Crowd simulation of pedestrians in a virtual city, Bachelor's thesis, RU, South Africa (2008)

[6] Brogan D., Hodgins J.: Group Behaviours for Systems with Significant Dynamics. Autonomous Robots, 4. p. 137-153. (1997)

[7] Ristic R., Berglund J.: High-density Real-time Virtual Crowds via Unilaterally Incompressible Fluid Simulation. Degree Project, KTH, Stockholm, Sweden (2015)

[8] Saiwaki N., Komatsu T., Yoshida T., and Nishida S.: Automatic generation of moving crowd using chaos model. Proceedings of the IEEE International Conference on System, Man and Cybernetics (1997)

[9] Ashley D.: Dynamic 2D Imposters: A Simple, Efficient DirectX 9 Implementation. Online article (2006)

[11] Harmeet Singha, Robyn Arterb, Louise Doddb, Paul Langstonb, Edward Lesterb, John Druryc.: Modelling Subgroup Behaviour in Crowd Dynamics DEM Simulation: Applied Mathematical Modelling, VOL 33 NO.12, p. 4408–4423. (2009)

[11] Saad A., Nishino K., Manocha D., Shah M.: Modeling, Simulation and Visual Analysis of Crowds: A Multidisciplinary Perspective, chapter 8, 187 (2013)

[12] Akker M., Gerearts R., Hoogeveen H., Prins C.: Path Planning for Groups using Column Generation, Technical Report UU-CS-2010-019, Utrecht University, Utrecht, The Netherlands (2010)

[13] Cui X., Shi H.: A*-based Pathfinding in Modern Computer Games. International Journal of Computer Science and Network Security, VOL.11 No.1 (2011)

7. Acknowledgements

We would especially like to thank our supervisor Christopher Peters who has been with us throughout this project, scheduling meetings and giving us useful guidance. Other thanks go to our fellow students Staffan Sandberg and Martin Funkquist who have been collaborating with us during the project.