CROWD SIMULATION

by Team08 - Simulanti

Calm

Behavioral states

Our behavioral model is based on state machine with 4 states - **Calm**, **Path finding**, **Following path** and **Evacuated state**. At the beginning of the simulation, all agents are in the **Calm state**, in which they just randomly move around the map. As soon as they discover fire, or they are passed the information about fire, their state changes to **Path finding state**. In this state, every agent tries to compute his own evacuation path to avoid fire and get to safety. If there is such a path, his state changes to Following path state and agents starts to run towards his goal - exit door. If there is no such path (agent is blocked by fire), agent does not change state, but he tries to get as far from fire as possible. When agent reaches the exit door, he enters the **Evacuated state**, which means he escaped from building.

Graph algorithms

Searching the most efficient path for the agent is calculated by **customized Q-learning algorithm** which precalculates distances among particular meshes in map. If agent cannot find the path because of the fire emeregancy, additional computation of the save path is performed by **A* algorithm**. If such path does not exist, agent just run away as distant from the fire as possible.

Fear factor

In our solution, every agent has his **own level of fear**. At the beginning, he is calm and his fear factor is low. As soon as he discovers fire, level of fear rises up. This sudden increase ends in change of state of the agent. **Calm state** is replaced by **Path finding state** and agent tries to find the nearest exit. While escaping from building, this agent spreads the information about fire. Therefore, if he enters the room, where there is no information about fire yet, level of fear of every agent within room raises as well. Moreover, fear factor affects the **speed of movement of the agents**. It also decreases over time, which brings another aspect of realism in our simulation. As agents move away from fire, their level of fear decreases and they slow down as well.

Path finding

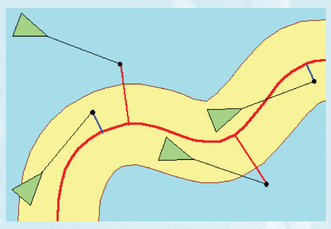
Following path

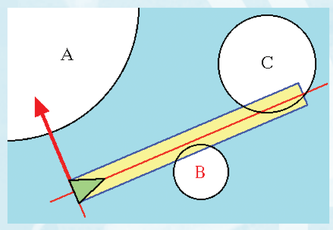
Evacuated

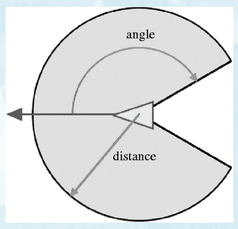
Steering vectors

In our work we use **two types of steering behaviours**. One that ensures that the agent won't walk into wall is called **obstacle avoidance**. Our implementation of avoidance basically checks distance length between agent and nearest walls. In case that length is under specific threshold, the avoidance vector pointing away from the wall is applied to agent's movement. The other steering behaviour we incorporated is called **separation** and is taking care of **preventing collisions between agents** themselves. Basic principle is the same as with avoidance, but in this case the distance is computed between two agents. Similarly, separation vector is pointing away from the adjacent agent.

Resultant agent movement is computed as weighted sum of path following, obstacle avoidance, separation and flow field vectors.

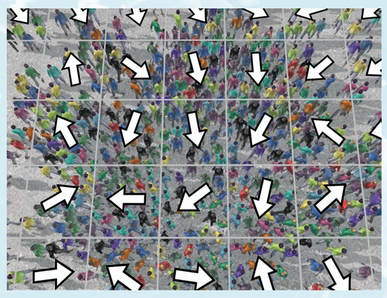






Flow field

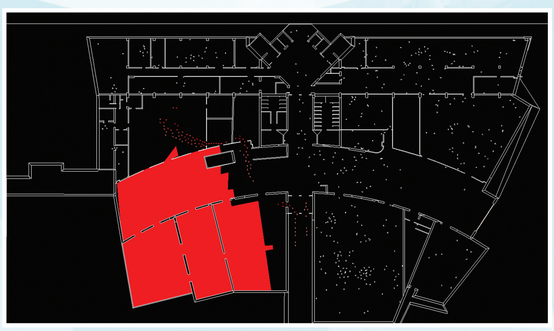
We take a certain area around agent, compute the vector which is **resultant movement vector of all agents** in this area at given moment. After steering vector of movement is computed, we let this area movement vector to affect **resultant steering vector of agent**. This simulates fact, that human in evacuation situations usually **follow the rest of the crowd**.

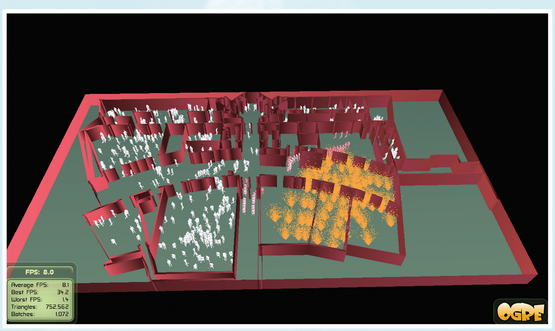


Distributed computing

Realistic crowd simulation brings many difficulties - main ledge is computing complexity. Our solution needs a lot of computing power. Having this fact in mind from very beginning, we designed the application architecture, so that it can be **used on more than one computer simultaneously**. Usage of distributed computing makes development much more difficult, but it also gives us brand new possibilities in crowd simulation. Distributed computing is realized by **MPI**, particularly **MPICH2**.

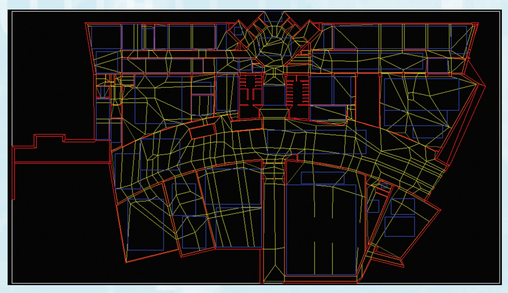
Visualisation





Map representation

The simulation environment is based on maps data which are defined in **DXF file format**. DXF is interchangeable file format which can be converted from internal formats of various programs (e.g. **3Ds**, **DWG**, **OSM**, **C4D**). Map contains several layers which represent its properties. Low-level layers describe **constraints**, **obstacles**, starting and target **locations of agents**. High-level layers contain **navigation mesh** which is used to aid in path-finding through large spaces. This way we have extensive capabilities of creating and **simulating real situations**.



S T U
F I I T
Institute of Informatics and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava, Slovakia

Team08 - Simulanti
Michal Fornádeľ, Lukáš Pavlech, Adam Pomothy, Martin Košický, Daniel Petráš, Marek Hlaváč
tim08@ googlegroups.com