13 Prototyping movement using Particle swarm optimisation

J. Pearse

By treating each zombie in a swarm as an individual particle and implementing Particle Swarm Optimization (PSO) we can test that neighbourhood awareness is functioning correctly. PSO is relatively simple and requires most of the features which can later be used to implement a flocking algorithm. The way in which space is divided up by the Tile-Viewer system (see section:9.2) means the algorithm functions as though there are multiple overlapping particle swarms, the only things which had to be implemented that are not useful in flocking is the fitness function and storing the previous fittest position in the zombie state. For this we used a simple distance measurement to a human, the 'optimisation' is to try and reduce the mean distance to the human across the swarm.

13.1 Awareness of neighbours

A zombie gets a list from the viewer associated with the current tile containing all the positions of entities on neighbouring tiles, this list is first sorted by distance then truncated to remove all entities outside the zombies sensory range. The resulting list is an Erlang map with the entity type as the key, allowing efficient retrieval of all the target entitys first. For the PSO, target entity's are of fitness zero (there is zero distance between a target and itself). If no target is visible the next lowest fitness individual becomes the focus of the optimisation. The zombie uses the (clamped) difference between the fittest neighbour and its own previous fittest position as its new position.

13.2 Usefulness

Implementing this simple PSO allowed us to uncover and remove bugs from the neighbourhood awareness system and prove that it was functioning as expected, we were forced to make the state of entities much more generic and update the system of state reporting, to incorporate state messages with an arbitrary number of variables. All of the required trigonometry and improvements to the tile-viewer message system will be invaluable to the next stage of development.

13.3 Conclusion

The PSO prototyping implementation was successful in it's aim of testing the system with a reasonably simple movement algorithm [2], it took a week to implement and almost all of it is usable going forward. We didn't fix those bugs which were deemed specific to PSO fitness and velocity as the entire PSO fitness function is to be discarded in the next iteration. Trigonometric functions were placed into a separate module as these will be reused later as were the functions for the PSO fitness optimisations. We enjoyed watching the results of the PSO running as the zombie particle swarm gave a believable impression of coordinated movement. There's nothing intrinsically wrong with using PSO to provide the swarm movement, but the emergent behaviour is lacking. The ability of an individual zombie to be 'aware' of the fitness of it's neighbours is questionable in our stated aim of realism, combining this with the simplistic nature of PSO we have evaluated that a better option would be a flocking algorithm.

[2]The specific function used to calculate movement in our PSO was $V = IV_c + R_i(B - C) + R_j(T - C)$ where V is the velocity along a given vector, I is inertia, V is the new velocity, V_c is the current velocity R_x is a random number such that $R_x \leq 1$, B is the previous recorded fittest position ,C is the current position and T is the targets position.