Open Source Continuum Crowds on the CPU & GPU
Patrick Crain, Prof. Joel Adams (advisor), Calvin College

Abstract

Pathfinding is a very important subfield of AI, with applications in physics, robotics, simulations, searching, and navigation. Since its inception, A* and its variants have been the standard algorithms for pathfinding, due to their relative simplicity and efficiency. Even so, most variants of A* fall short when it comes to long term dynamic planning and local collision avoidance.

The continuum crowds algorithm offers significant improvements over the A* family concerning planning and collision avoidance; additionally, it can handle tens or even hundreds of times more agents with little penalty to speed, and no penalty to accuracy. Despite this, the algorithm sees less widespread use due to its relative complexity and the lack of source code to build from.

Our project aims to supply well-documented source code for others to reference and build upon, in hopes that continuum crowds are further improved and explored as an alternative to the A* algorithms.

Example: Vortex

Agents using a naïve A* search would cut through the center to reach their goals, quickly blocking other agents' paths. Agents using continuum crowds notice this congestion, and take a longer path to ultimately save time.

Example: Density

Agents initially move symmetrically straight towards their goals, but as the 4 groups approach the center, they trickle along outward paths to avoid an overly-crowded deadlock in the center, eventually sorting themselves out neatly.

Example: Impasse

The blue and red agents both initially head towards the same openings to reach their goals on the opposite side. Red concedes at the first opening due to a prohibitively long path for blue, whereas blue concedes at the second opening due to red's excessive numbers.

Example: Vortex

The blue and red agents both initially head towards the same openings to reach their goals on the opposite side. Red concedes at the first opening due to a prohibitively long path for blue, whereas blue concedes at the second opening due to red's excessive numbers.

Example: Vortex

The blue and red agents both initially head towards the same openings to reach their goals on the opposite side. Red concedes at the first opening due to a prohibitively long path for blue, whereas blue concedes at the second opening due to red's excessive numbers.

Example: Impasse

Motivation

The original continuum crowds algorithm was developed in 2006 by a group from the University of Washington [1]. Aside from a fairly robust GPU version developed by AMD in 2008 [2], there have been very few improvements to the general algorithm. Of the few groups who have attempted to improve the algorithm and published papers on their work, none (to our knowledge) have published their source code, effectively making collaboration between groups impossible and any further research independent.

For large groups of agents, continuum crowds offer significant speed and accuracy improvements over the more commonly used A* variants. We hope that by maintaining an open-source repository, incorporating implementations of both CPU and GPU continuum crowds, we will be able to provide an easy-to-understand starting point for those wishing to experiment with the algorithms, as well as a hub where potential improvements can be discussed, merged, and shared.

Motivation

The original continuum crowds algorithm was developed in 2006 by a group from the University of Washington [1]. Aside from a fairly robust GPU version developed by AMD in 2008 [2], there have been very few improvements to the general algorithm. Of the few groups who have attempted to improve the algorithm and published papers on their work, none (to our knowledge) have published their source code, effectively making collaboration between groups impossible and any further research independent.

For large groups of agents, continuum crowds offer significant speed and accuracy improvements over the more commonly used A* variants. We hope that by maintaining an open-source repository, incorporating implementations of both CPU and GPU continuum crowds, we will be able to provide an easy-to-understand starting point for those wishing to experiment with the algorithms, as well as a hub where potential improvements can be discussed, merged, and shared.

References


This work was made possible by NSF DUE #1225739.