Optimization & vectorization

UU Crowd Simulation Software

Roland Geraerts
October 14, 2019
UU Crowd Simulation R&D
Unity3D Plugin
However...

• Global framework
  – Agents are simulated in parallel using OpenMP

• Real-time performance
  – UUCS: simulates 15K agents
  – Unity: animates and visualizes 1.5K agents
30K in real-time on a fast laptop
What changed?

• UUCS
  – Made some remaining code run in parallel

• Unity
  – From objected oriented to data driven implementation
    • Entity component system & Job system
  – From main thread to separate threads
  – From CPU animations to GPU shader-based animations
However...

- Current optimizations in UUCS
  - Theoretical running times: \( O(...) \)
  - Parallel code using OpenMP
- So much is still possible...
  - ...but we first need to understand the framework
How can you simulate a human crowd interactively?
Crowd simulation framework

- Representation environment
- Level 5
  - Plans actions
- Level 4
  - Creates indicative routes
- Level 3
  - Traverses the routes
  - Yields speed/direction pairs
- Level 2
  - Adapts routes
  - E.g. to avoid collisions
- Level 1
  - Moves the agents
Crowd simulation framework

- Representation environment
- Level 5
  - Plans actions
- Level 4
  - Creates indicative routes
- Level 3
  - Traverses the routes
  - Yields speed/direction pairs
- Level 2
  - Adapts routes
  - E.g. to avoid collisions
- Level 1
  - Moves the agents
Representation of the environment

- Computation of walkable areas and navigation mesh

Crowd simulation framework

- **Representation environment**
- **Level 5**
  - Plans actions
- **Level 4**
  - Creates indicative routes
- **Level 3**
  - Traverses the routes
  - Yields speed/direction pairs
- **Level 2**
  - Adapts routes
  - E.g. to avoid collisions
- **Level 1**
  - Moves the agents
Level 4: Indicative routes

- We use the Explicit Corridor Map (ECM)
  - Compact navigation mesh
  - Supports any agent radius
  - Multi-layered environments
  - Dynamic updates
Crowd simulation framework

- Representation environment
- Level 5
  - Plans actions
- Level 4
  - Creates indicative routes
- Level 3
  - Traverses the routes
  - Yields speed/direction pairs
- Level 2
  - Adapts routes
  - E.g. to avoid collisions
- Level 1
  - Moves the agents
Level 3: Path following

- Smoothly follow a desired path
  - Input: indicative route, non-smooth indication of the path
  - In each simulation step, compute an attraction point
  - Leads to a preferred velocity for the next level

- Indicative Route Method (IRM, 2009)

- MIRAN: improvement by Jaklin et al. (2013)
  - Supports weighted regions
  - Better smoothness/shortcut control
Crowd simulation framework

- Representation environment
- Level 5
  - Plans actions
- Level 4
  - Creates indicative routes
- Level 3
  - Traverses the routes
  - Yields speed/direction pairs
- Level 2
  - Adapts routes
  - E.g. to avoid collisions
- Level 1
  - Moves the agents
Level 2: Local movement

- Roughly move in the preferred direction, while...
  - responding to collisions with other characters
  - avoiding future collisions
  - adapting to the surrounding streams of people
  - maintaining social group behavior
  - etc.
Crowd simulation framework

- **Representation environment**
  - Level 5
    - Plans actions
  - Level 4
    - Creates indicative routes
  - Level 3
    - Traverses the routes
    - Yields speed/direction pairs
  - Level 2
    - Adapts routes
    - E.g. to avoid collisions
  - Level 1
    - Moves the agents

![Diagram of crowd simulation framework]

**Representation of the environment**
DEMOlITION TIME

The UUCS engine in action
IMPLEMENTATION DETAILS
Crowd simulation in the UUCS framework
Simulation step

- performStep(Δt)
  - For each agent: path following
    - Update pointers along indicative route
    - Update attraction point, preferred velocity
  - For each agent: collision avoidance
    - Compute new velocity vNew
    - Smoothen vNew (optional)
    - Compute collision forces F (optional)
  - For each agent
    - Update velocity: v := vNew + Δt · F/mass
    - Update position: p := p + Δt · v
  - Update nearest-neighbor data structure (There are actually many more substeps)

Why separate loops?
- Order of agents does not matter
- Agents are independent, each loop can be parallellized
Performance (without visualization)

- 1 thread (2015)
Performance (without visualization)

- 8 threads: 4 cores (2015)
Assignments
Collision avoidance 1 / 2

• Algorithm
  – Focus on Optimal Reciprocal Collision Avoidance (ORCA)
  – Appears to be the most expense part of UUCS

• Code
  – src/Simulation/CollisionAvoidance/CollisionAvoidance_RVO.cpp
  – 444 lines of code
  – Code includes solving a linear program

• Literature
Goal
- Optimize CPU code, or
- Convert to GPU implementation

Performance criterion
- Relative difference in total running time of collision avoidance during 60s (600 frames) in city environment with 25K agents
KD-tree 1 / 2

• Algorithm
  – NanoFlann
  – Computes and queries a nearest neighbors KD-tree

• Code
  – src/external/nanoflann/nanoflann.hpp
  – 1946 lines
  – Optimized templated C++ code

• Literature
  – https://github.com/jlblancoc/nanoflann
KD-tree 2 / 2

• Goal
  – Optimize C++ code

• Performance criterion
  – Relative difference in total running time of building the KD-tree and all nearest neighbor queries during 60s (600 frames) in city environment with 25K agents
• Algorithm
  – UUCS codebase

• Code
  – Mainly src/Simulation/*
  – 10K lines?

• Literature
  – Framework: https://www.staff.science.uu.nl/~gerae101/UU_crowd_simulation_publications_framework.html
• Goal
  – Optimize C++ code

• Performance criterion
  – Relative difference in total running time of the whole simulation during 60s (600 frames) in city environment with 25K agents
Prizes

1. Arduino starter kit
2. Arduino starter kit
3. Arduino starter kit
Getting started

• Sign EULA
  – Improvements may be integrated in UUCS
  – IP goes to University so that education and research is secured
    – ucrowds.com/eula

• After signing, you will get access to
  – UUCS library and demo projects
    • https://git.science.uu.nl/UUCS/explicit-corridor-map-framework

• To compile the project
  – Follow the instructions listed in README.md
  – You can get some help
Technical support

• Compilation
  – Geert-Jan Giezeman
  – g.j.giezeman@uu.nl
  – BBG 5.77
  – Please send him an e-mail first

• Weekly visit hour
  – Monday 10.00 - 11.00
  – UtrechtInc, Padualaan 8, Office W125
  – Contact Yiran Zhao
    • yiran@ucrowds.com
Roland Geraerts  
R.J.Geraerts@uu.nl  
uu.nl/staff/RJGeraerts  
BBG 4.07  
06 28 80 49 01