# Report from running Inverse Kinematics using PSO algorithm

Experiment 1

Course: Research project - GPU algorithms
Coordinator: Krzysztof Kaczmarski

Eryk Dobrosz
Marcin Gackowski

June 27, 2019

# Description

Project tackles inverse kinematics problem and solves it using Particle Swarm Optimization algorithm implemented with CUDA. Main goals of the project are:

- real-time solving of inverse kinematics problem for arbitrarily large kinematic chains with many effectors
- solution choice that can be used for animation

## Output Files

- IK-diagnostics-positions.txt - contains positions of nodes each frame during test cases
- IK-diagnostics-degrees.txt - contains rotations of nodes each frame during test cases
- IK-diagnostics-distance.txt - contains aggregated distance of all effectors from their respective targets each frame during test cases
- IK-diagnostics-frames.txt - contains number of frames required to reach acceptable solution for each test case

# Report Goals

- Test implemented PSO algorithm and analyze gathered results in terms of performance and capability of real-time solving of inverse kinematics problem.
- Analyze potential for use in animations.
- Detect possible issues in current approach to find room for further improvements of used algorithm.

# Computational Method

Method used in this project is based on general-purpose Particle Swarm Optimization algorithm which is implemented in a way that takes advantage of task parallelization available on GPU.

The algorithm main parts are:

- **Simulation space** - set of values where each element corresponds to a single valid state of the input system. In this case it is a $n$-dimensional space, where $n$ is equal to degrees of freedom in kinematic chain. Each dimension represents rotation of a single node along one axis.
- **Particle set** - set of particles (points) moving through simulation space where position of each particle represents a valid state of the input system. Movement and velocity of each particle is based on:
  - inertia (keeping particle's previous velocity),
  - local extremum (exploring neighbourhood of best solution found by given particle)
  - global extremum (exploring neighbourhood of best global solution)
- **Fitness function** - $n$-argument function designed for a concrete problem where global extrema (usually minima equal to 0) are optimal solutions. Domain of this function is equal to the simulation space of PSO algorithm. In this project optimal solution is achieved when distance of all effectors to their respective targets is equal to 0.
- **Simulation** - process of updating particle positions and velocities either a certain number of times or to a moment when majority of particles converged to a single extremum. Due to inconsistent converging rates and requirement for real-time visualization of approaching the solution, this project has set number of iterations after which PSO simulation terminates.

Nature of the PSO is highly parallel with the only exception of calculating global best solution. Using GPU to execute this algorithm leads to significant performance increase, which allows simulation of much larger particle sets and therefore improves quality of found solution.

Each thread is responsible for simulation of a single particle and saving results to device memory. At the end of each simulation step, parallel reduction algorithms are used to find and update globally best result of fitness function. When simulation terminates, global best solution is returned and used to update state of rendered kinematic chain.

# Description of the results

Results were gathered through numerous executions of set test case. System configuration for testing was as follows:

- Kinematic chain with 21 degrees of freedom, that consists of:
  - origin node placed in (0, 0, 0)
  - 4 consecutive nodes creating a chain with all links of length equal to 1
  - 3 effectors linked to the last node in the chain with links of length equal to 1
- 16384 simulated particles
- 3 targets in set positions that can be reached by all effectors
- aggregated error threshold equal to 0.025 (sum of distances from effectors to their targets)

At the start of each test case system was reset to its default state, providing reproducible initial conditions for the algorithm.

Gathered results allow to verify used algorithm both in terms of performance and stability in approximation of optimal solution. Results for current iteration are as follows:

- frames to reach satisfactory solution:
  - average: 3.13
  - min: 1
  - max: 12
- frame-difference for each degree of freedom (angle in radians):
  - average: 2.04
  - min: 0.00
  - max: 6.28
- frame-difference for each node position (distance in OpenGL units):
  - average: 0.28
  - min: 0.00
  - max: 1.95

# Remarks

Results from test case show that PSO algorithm solves inverse kinematics problem with sufficient speed to allow using it in real-time animation, requiring less than 4 frames on average to approach satisfactory solution. However, rotation and position difference for each frame are unacceptably large and inconsistent, covering almost entire range of available values - $2\pi$ for rotation and 2 units for position difference. Most likely cause of these inconsistencies is uniform particle distribution through simulation space. This approach helps with searching for global extrema but also discards previously found solution, therefore it does not favour solutions that are locally available.

# Future works

- Modify algorithm to take advantage of previously found approximations.
- Improve consistency of calculated solutions to minimize position and rotation differences for each frame.
- Optimize data structures used by GPU to utilize memory coalescing.