Report from running Inverse Kinematics using PSO algorithm

Experiment 2

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 changes to PSO algorithm concerning initial particle distribution.
- Detect system jittering in current approach to find room for further improvements in next iterations.

Computational Method

Project is using modified PSO algorithm used in first iteration to solve inverse kinematics problem, therefore method descriptions stated in previous report still apply.

Main changes concern particle distribution, which is now dependent on the current state of the system. Kinematic chain rotations are mapped to simulation space and every particle in particle set is initially placed in these exact coordinates, instead of being uniformly distributed across entire simulation space. This modification's main goal is to reduce system jittering, consequently improving project in terms of use in animations.

Description of the results

Results were gathered through numerous executions of set test case. System configuration for testing matches test case used in previous iteration:

- 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: 4.15
  - min: 2
  - max: 31
- frame-difference for each degree of freedom (angle in radians):
  - average: 0.16
  - min: 0.00
  - max: 1.31
- frame-difference for each node position (distance in OpenGL units):
  - average: 0.11
  - min: 0.00
  - max: 0.86

Remarks

Results from test case in this iteration show significant improvements in average rotation and position difference that present as follows when compared to previous iteration:

- 13 times smaller rotation difference on average
- 2.5 times smaller position difference on average
- 1 additional frame needed to reach satisfactory solution on average

Increase in average frame count to reach solution was expected since current iteration puts more emphasis on using locally available extrema to naturally interpolate system between initial and desired system configurations.

However, maximum values in rotation and position differences suggest that algorithm still occasionally jumps to different areas in simulation space, essentially discarding previously found approximation. This undesirable behaviour stems from random nature of PSO algorithm, meaning that further improvements upon distribution and simulation of particles will not eliminate the issue and can only minimize its probability. Instead, modifying currently used fitness function could ensure that local solutions would be favoured over solutions from different areas of simulation space, which could eliminate the issue completely.

Future works

- Redesign fitness function to favour locally available solutions.
- Implement collision system.
- Implement angle constraints.