Parallelized Boid Simulation

Milestone Report

Josiah Miggiani, Ryan Huang

URL: https://josiemigg.github.io/parallel-project.html

WORK SCHEDULE:

(4/15 – 4/18)
- Finish milestone report (due Tuesday, 4/15 @ 11:59pm) [@ All]
- Begin sophisticated binning implementation [@ Ryan]
- Finalize benchmarking configuration [@ Josiah]

(4/19 – 4/21)
- Continue sophisticated implementation [@ Ryan]
- Assist sophisticated implementation [@ Josiah]

(4/22 – 4/24)
- Finalize sophisticated implementation [@ All]
- Evaluate sequential, basic OpenMP approaches [@ Josiah]

(4/25 – 4/27)
- Evaluate sophisticated shared memory approach [@ Josiah]
- Begin final report [@ Ryan]
- Begin final poster [@ All]

(4/28 – 4/29)
- Finish final report and submit (due Monday, 4/28 @ 11:59pm) [@ All]
- Present at poster session (Tuesday, 4/29 @ 1 – 4pm or 5:30 – 8:30pm) [@ All]

SUMMARY OF WORK:

Up to this point, we've been working to adapt the existing sequential boids simulation to be benchmark-ready. Firstly, we've added a simulation bounding box, which limits boid movement within a cube surrounding the origin. This is accomplished by gently applying a "push force" to boids that stray outside the boundaries so that they are smoothly redirected back inwards. The bounding box prevents boids from flying to infinity, outside of the view of the camera observer. Secondly, we implemented deterministic boid system behavior, or controllably-random evolution enabled by random number generators and seeds. For any input space (e.g. boids=5000, bounding box width=100.0f, simulation time=2000 ticks, etc.), the results will be identical from one run to the next. This accelerates our testing process through improving ease of generating tests. Care was taken to introduce and ensure thread-safety, such as assigning a unique RNG and real distribution to each boid. We also had to change the simulation to take uniform time steps, or ticks, rather than using GLUT_ELAPSED_TIME as the base code did. This was the last piece to ensure truly deterministic simulation, as GLUT_ELAPSED_TIME is a register-callback based counter that measures wall-clock time, and is not well suited for evaluating simulations. Lastly, we implemented non-visual simulation (and toggle functionality)

and initialization/computation time reporting, similar to asst3. For the graphical simulation, we have a framerate detector that measures the duration of performing each system evolution update.

With the simulation nicely situated for benchmarking, we introduced a very basic OpenMP implementation for initial parallelization testing. We parallelize across boids during system evolution, and detect minor speedup in frame rate in the graphics simulation. Some preliminary results we've gathered can be viewed below.

GOALS AND DELIVERABLES:

Currently we are on track with our goals and schedule, and we believe that we will be able to fulfill the goals that we outlined in our initial project proposal. We have begun thinking about how to go about improving the naive $O(N^2)$ algorithm and have plans for our sophisticated binning algorithm. While it's not entirely clear that we'll be able to reach our desired speedup of O(NlogN), we will certainly be able to improve upon the naive parallel algorithm and produce results demonstrating heightened performance. It would be nice to be able to directly demonstrate our simulation at our poster session, so we will need to verify the capabilities of our laptops. Up to this point, development and evaluation has been performed on desktops.

Our new list of goals:

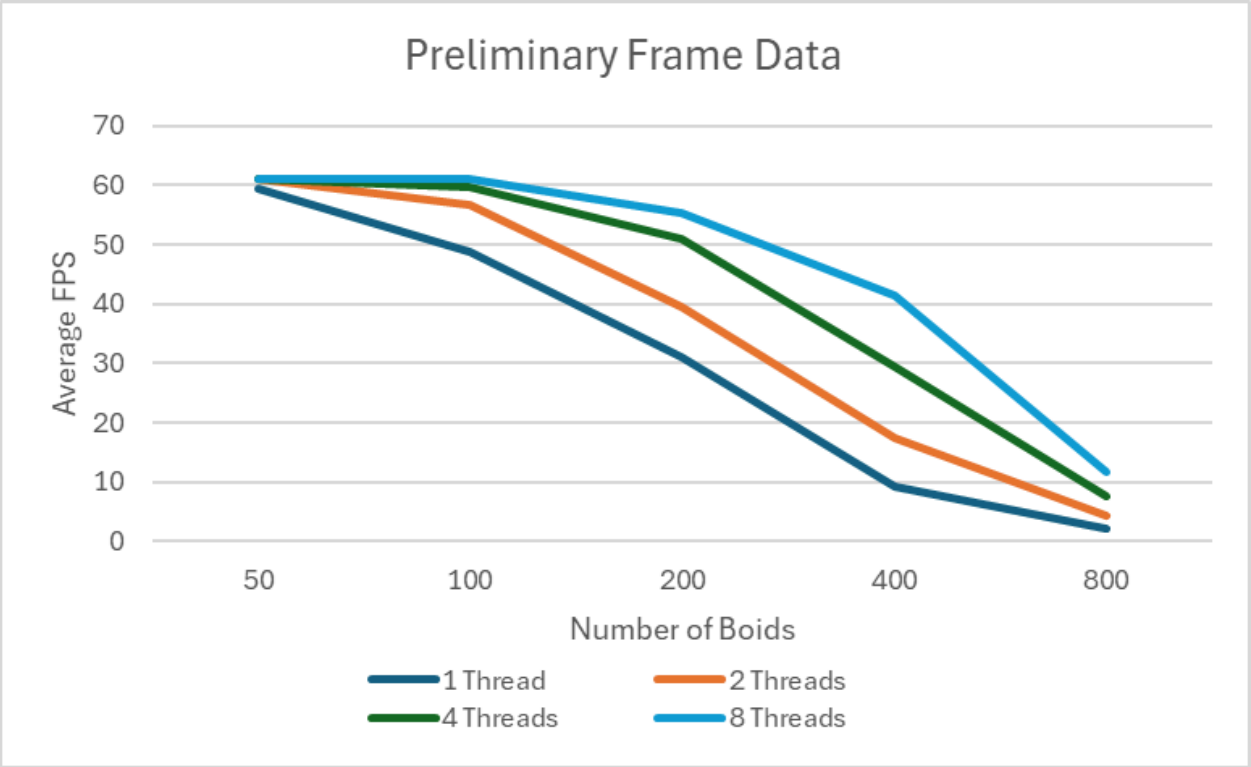
1. Achieve an algorithmic complexity of O(NlogN) during each update step
2. Time-constrained scaling: increase graphically simulated boid count by ~4x while maintaining visual framerate
    a. Attain similar results for graphics-less simulation. Achieve increased boid count within same (real) time allotment
3. Problem-constrained scaling: with fixed boid count and simulation time, increase maximum possible framerate by ~6x
    a. For graphics-less simulation, improve computation/overall speedup by a similar factor
4. Produce graphs and videos highlighting improvements in speedup/performance

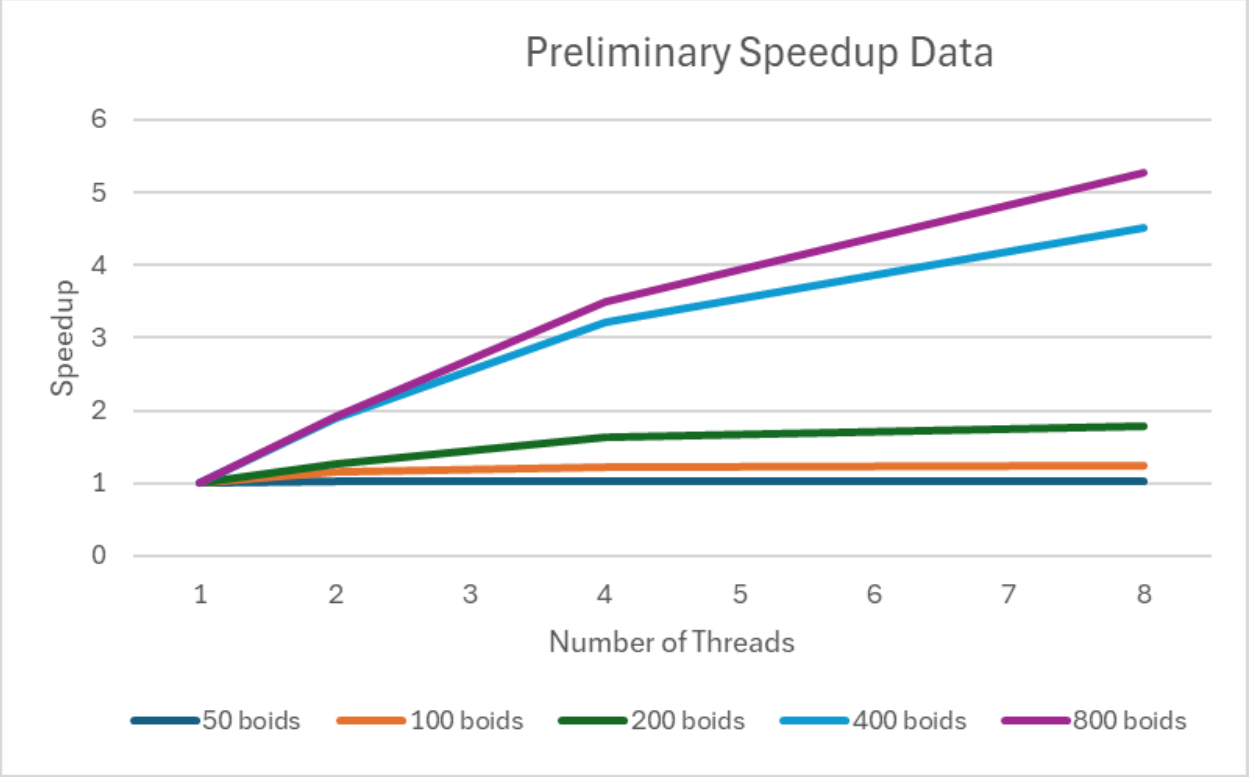
POSTER SESSION:

At our poster session, we will demonstrate our visual simulation directly on our laptops in realtime, complemented by our graphics-less simulation. We'll showcase our three implementations, being naive sequential, basic OpenMP, and sophisticated OpenMP (e.g. binning). We believe this will help visually emphasize the speedup that we achieve with parallelization techniques. If we find live simulations to be infeasible, we will present recorded animations. We will compile graph data to show the objective speedup that we achieve with our improvements. With this combination, we can show both the quantitative and qualitative improvements made to our program.

PRELIMINARY RESULTS:

We have some preliminary results with naive parallelization. We hope to achieve better speedup results with further improvements:

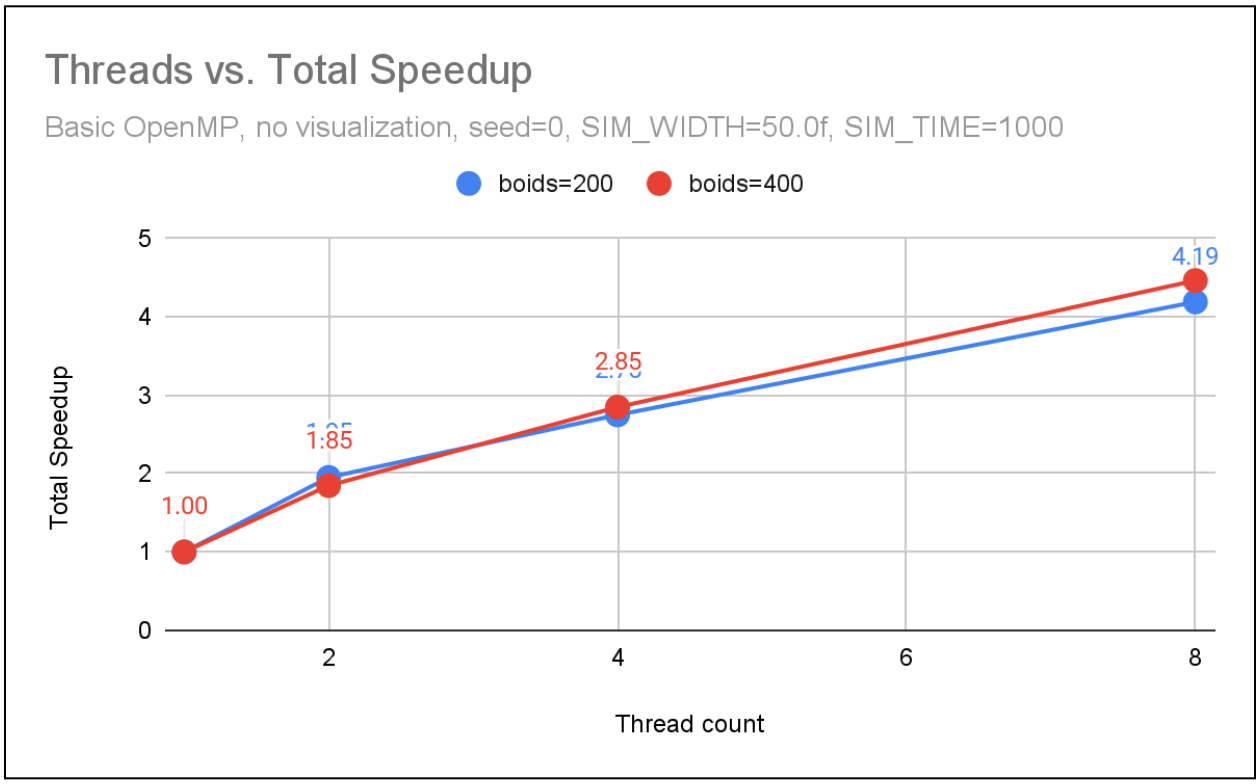




Note that this testing is rather preliminary. We still have a capped frame rate of 62.5 fps which affects our speedup. We also include the time taken to generate the visuals in our frame rate, so these are just starting sets of data to get a starting sense of how effective our parallelization is.

To illustrate, we've produced two .webms that showcase how increased thread count can enable more boids while maintaining framerate. Visit our website to view.

Here are results collected with graphics disabled:



REMAINING CONCERNS:
There isn't much that stands in the way of our final goal. The largest obstacle is implementing the binning algorithm for shared memory accesses when updating boids, but this is a matter of coding and doing the work.