AE6102 : Parallel Computing and Scientific Visualization

Final Project Report

Game of Flies :

a particle-system simulator in python

Members:

Tanyut Sharma (190100128)
Vinit Doke (190260018)
Mihir Agre (190260030)

Abstract

Time evolution of a particle-system based on two interaction rulesets : Boids and clusters, is simulated via numba CUDA, numba parallel, and numba serial implementations in python. Visualisation is done via VisPy-based 2D and 3D widgets embedded inside a PyQt5-based GUI for interactivity. A pathway to pre-compute the solution (bypassing the GUI) and visualise later, using either a live VisPy widget or rendering to a video is provided. Parallelization of binning (both 2D and 3D) for reducing time-complexity to $O(n)$, support for multiple interactive-species of particles, and periodic boundary conditions are also implemented.

Contents

List of Figures

List of Tables

1 Tools Used

Library/Tool	Utilization/Purpose
conda	Python Environment and Package Manager
GitHub and Git	Collaboration and Version Control
imageio	Video Stitching
matplotlib	Rudimentary Visualisations
numba	CUDA Programming and CPU Parallel Optimizations
numpy	Initialisation Random Number Generation
PyQt5	User Interface
Python Standard Libraries	os, pathlib, time, argparse
scalene	Profiling (CPU and GPU)
tqdm	Loading Bar
vispy	Primary 3D/2D Visualization
VSCode/ PyCharm	IDEs

Table 1: Tools and Libraries Used

2 Outline

In our project we intended to simulate complex emergent behaviour of a large number of point particles interacting with each other via simple rules. We added real-time and post sim visualisation with visPy, alongside real-time UI to update simulation parameters.
We began with implementing a naive $O(n^2)$ serial algorithm on the CPU via *numba* for computing particle-particle interactions. It could 'reasonably' handle around 1000 particles simultaneously in real-time. While writing this, we tried to make our code as flexible as possible to allow for more experimentation and more powerful algorithms.
This was then parallelized easily through *numba* itself and provided around a 3x performance boost on our laptops.
We then managed to write the parallel code for binning the particles to get $O(n)$ performance. We then found that *numba.cuda* does not implement the scan algorithm, and the inbuilt reduction function was extremely slow, so we had to implement those algorithms ourselves.
So far the particle interactions we were calculating were fairly simple longitudinal forces, but now we wanted to support 'Boids' as well. These require each particle to have awareness of its neighbours' average position and velocity. We eventually managed to simulate Boids alongside the simpler particles we were dealing with before.
Throughout these weeks while dealing with writing parallel code in *numba.cuda*, we encountered bugs that we had never seen the equivalent of in serial CPU coding. Thorough debugging and ensuring compatibility across multiple git branches took up the majority of our time, more than writing the code in the first place.

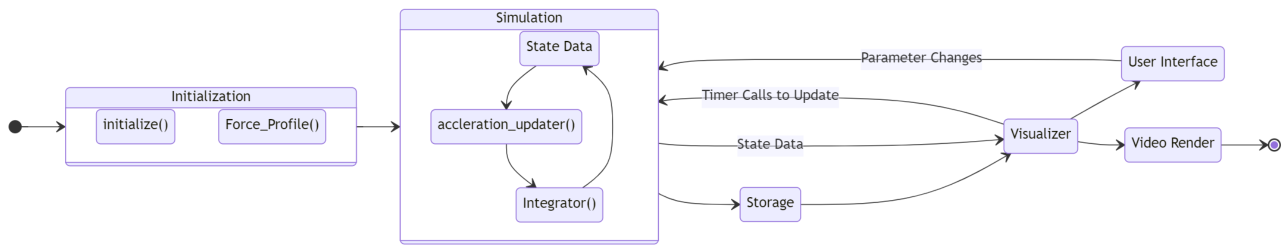

Figure 1: State Diagram of the Program

3 Showcase

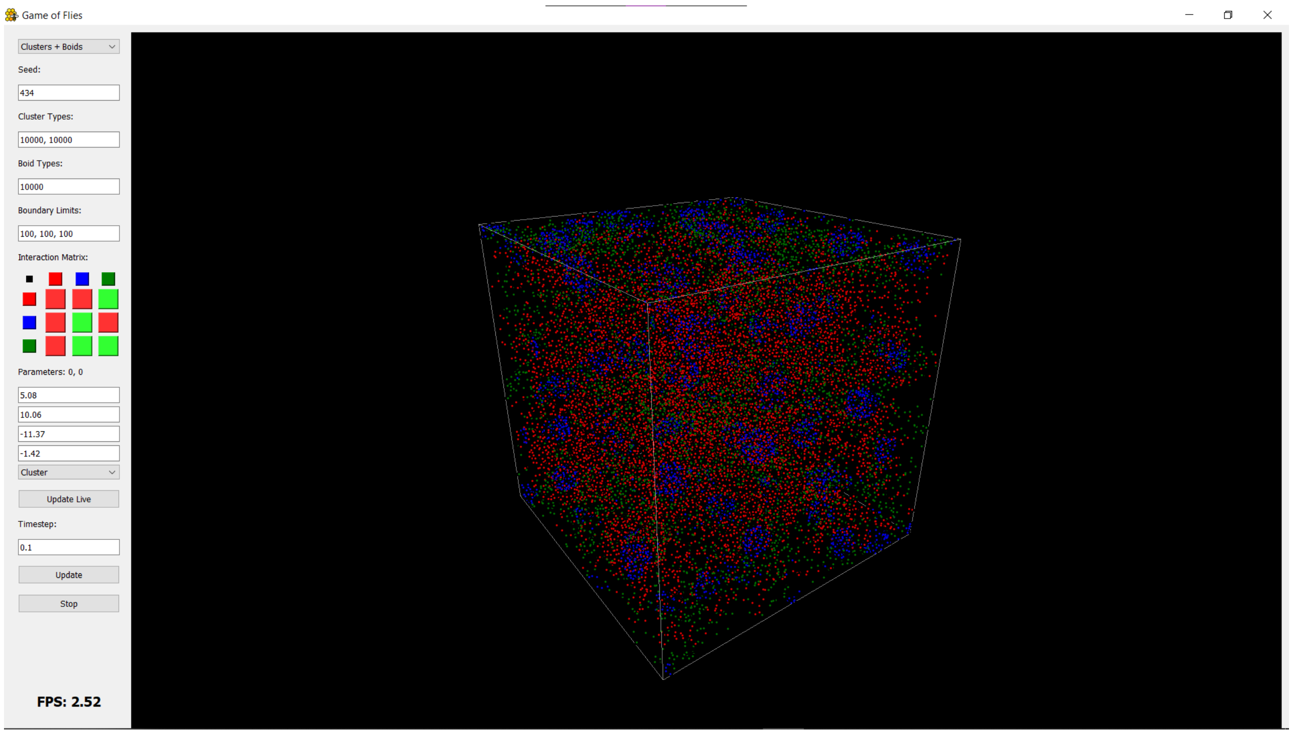

Figure 2: UI Snapshot

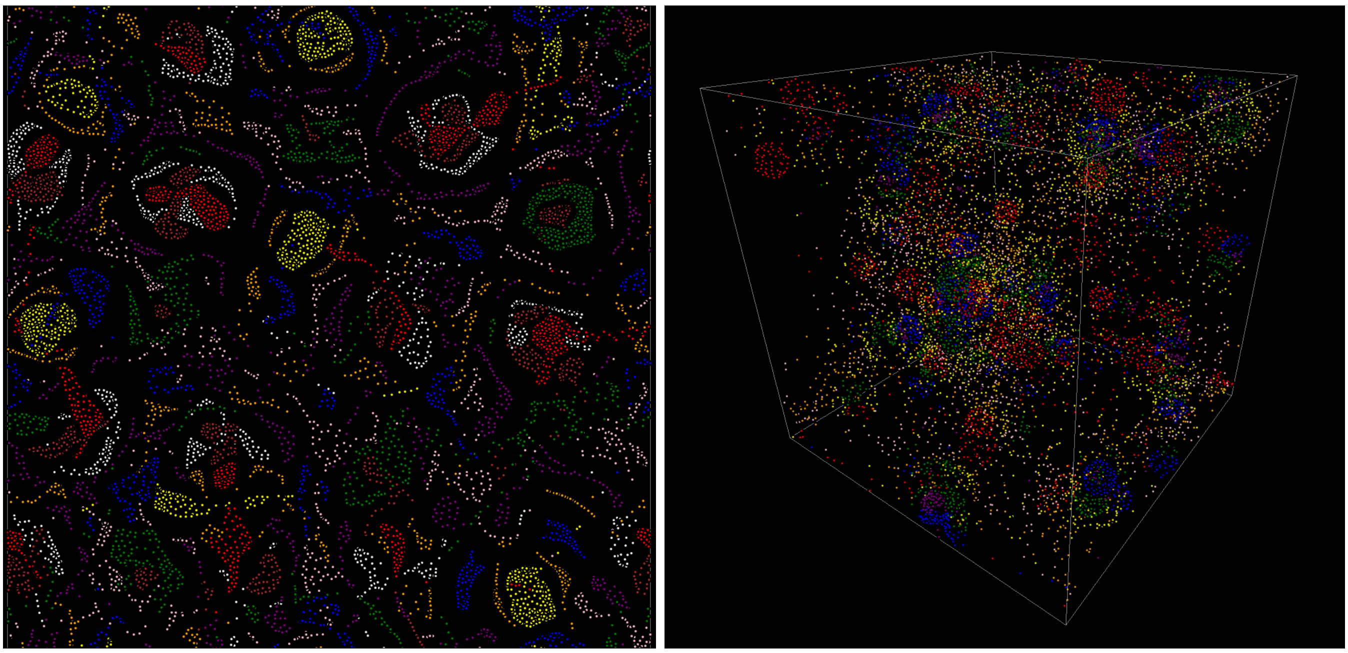

Figure 3: 2D and 3D Simulation Snapshots

Demo Videos

- Clusters 2D : 9000 particles https://youtu.be/mEeR2FnSDng
- Clusters and Boids 3D : 14000 particles https://youtu.be/gRxERbZKX5M
- UI Demo : https://youtu.be/PulBwRWKz0Q

4 Deliverables and Achievements

1. Types of inter-particle interactions implemented:
    (a) Clusters: Particles repel each other if too close, and may attract or repel at larger distances. Force function is piece-wise linear with distance.
    Complex structures and interactions appear as various number of particle types are increased.
    (b) Boids: Particles repel each other if too close, are attracted to the COM of their neighbours and their velocity rotates to align with the local average velocity.
    These form flocking behaviour similar to birds, hence the name "Bird-oids"
    (c) Mixing: Our code is written in a very general form and it is easily possible to mix any two force profiles or interactions by adding more particle types to the simulation. Mixing boids and clusters yields some fascinating results.
2. Numba jitted serial and parallel code for naive implementation
3. Basic binning algorithm (CUDA) extended for multiple particle types, multiple kinds of interactions, 3D and periodic boundaries
4. Postprocessing (frames saved as numpy arrays)
    (a) Visualize pre-computed simulation in vispy
    (b) Render pre-computed solution to a video
5. UI based real-time simulation

5 Logistics

Approx. Time	**Task**	**Contributors**
Post Midsem	Parameter Initialization	Mihir
Post Midsem	Integrator, Acceleration Update Logic	Tanyut
Post Midsem	Force Profile, Matplotlib Viz	Vinit
Pre Endsem	Object Oriented/ Numba Parallel	Vinit, Tanyut
Pre Endsem	Vispy 2D/3D Viz	Vinit
Pre Endsem	Scalene Profiling	Mihir
Pre Endsem	Video Stitching, Post-Compute Viz	Vinit
Pre Endsem	Binning (CUDA) + Boids : 2D	Tanyut
Post Endsem	Binning (CUDA) + Boids : 3D	Mihir
Post Endsem	PyQt5 GUI	Vinit

Table 2: Timeline and Contributions

6 Code Repository

Final Code in 'cuda_3D' branch of the following repository :
https://github.com/vinitdoke/Game_of_Flies
Direct Link :
https://github.com/vinitdoke/Game_of_Flies/tree/cuda_3D

7 References

1. Parallel scan/reduction algorithm : https://www.eecs.umich.edu/courses/eecs570/hw/parprefix.pdf
2. Boid basics: https://people.ece.cornell.edu/land/courses/ece4760/labs/s2021/Boids/Boids.html#Background-and-Introduction
3. Clusters by Jeffrey Ventrella: https://ventrella.com/Clusters/