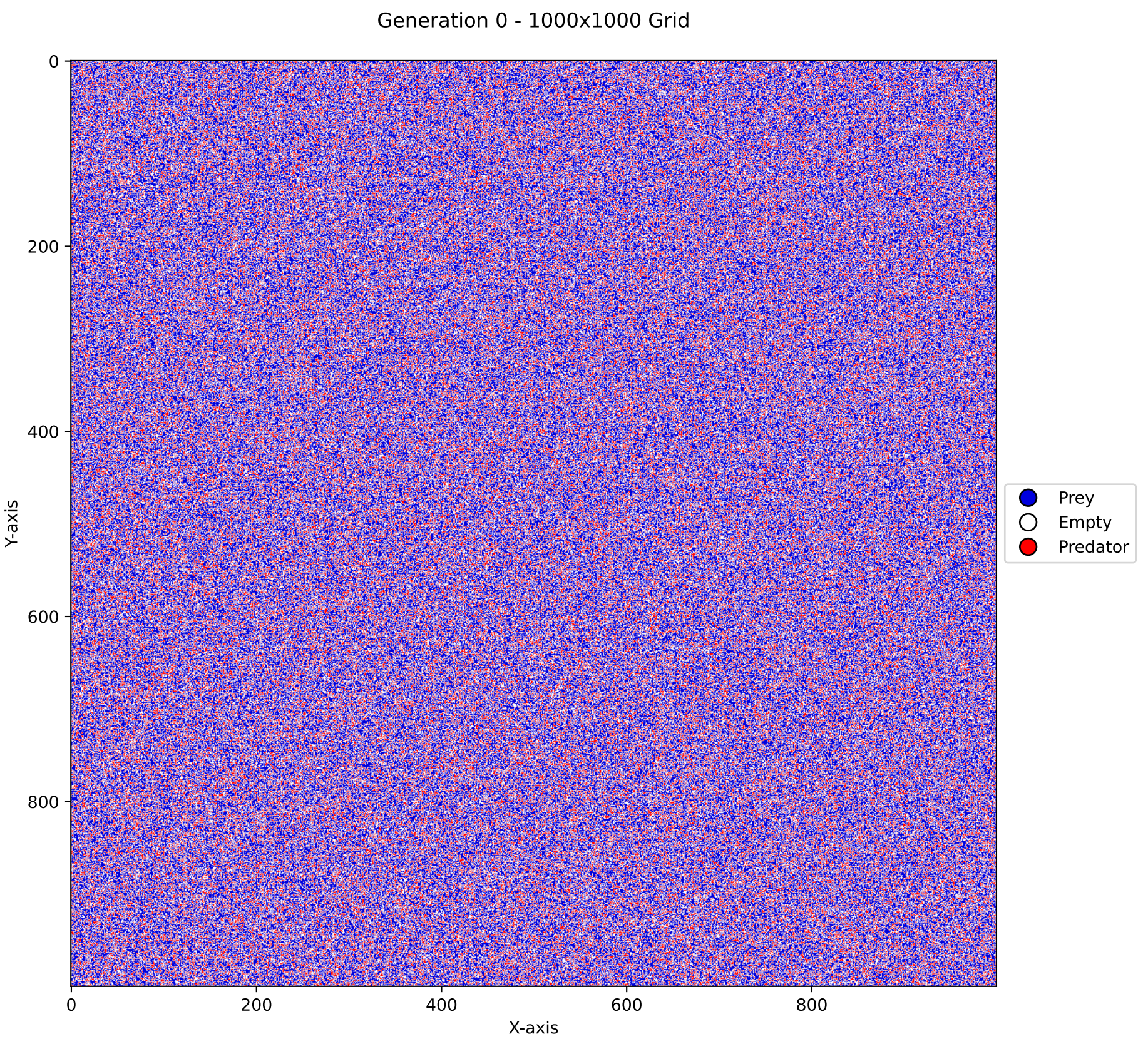
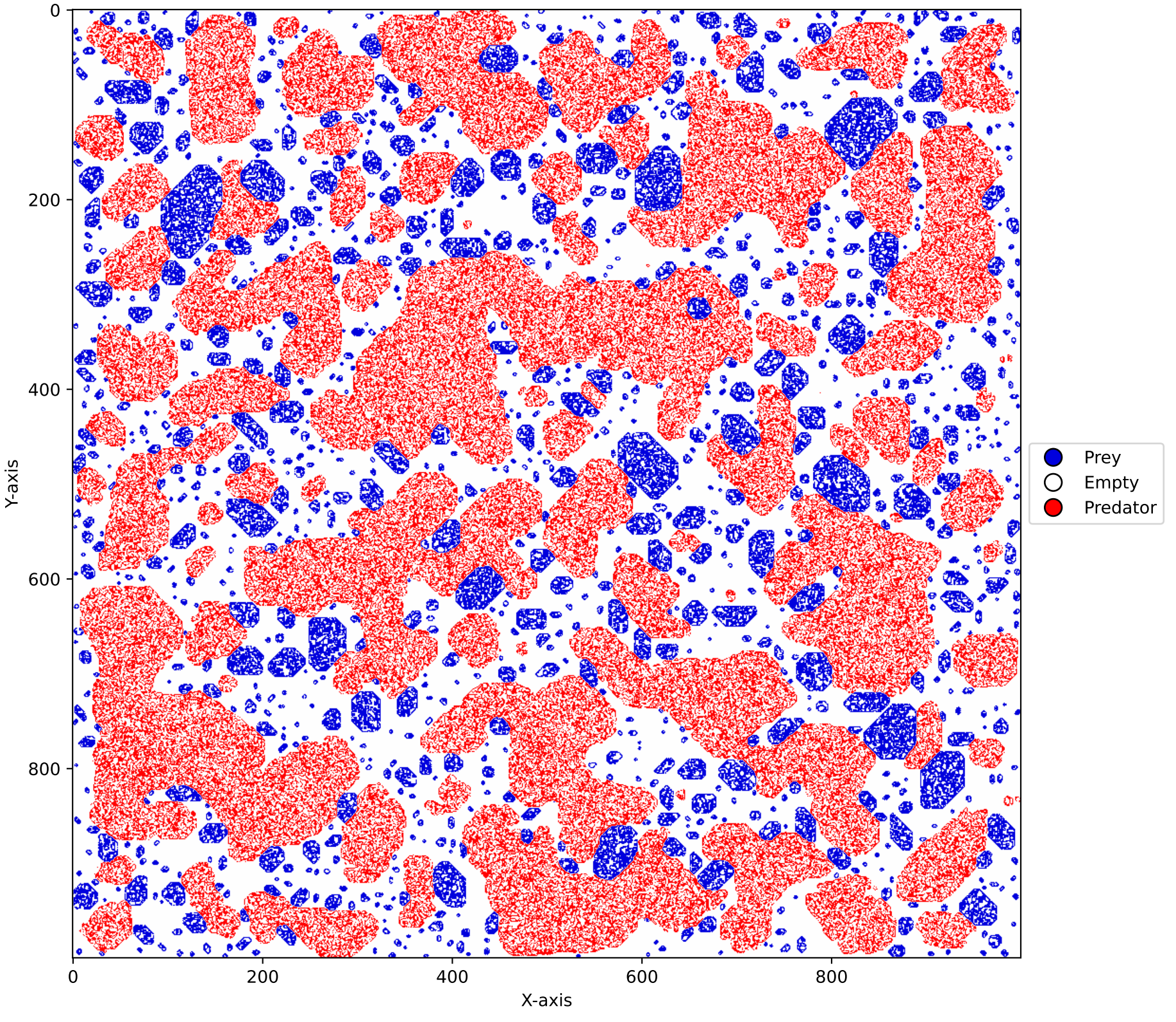
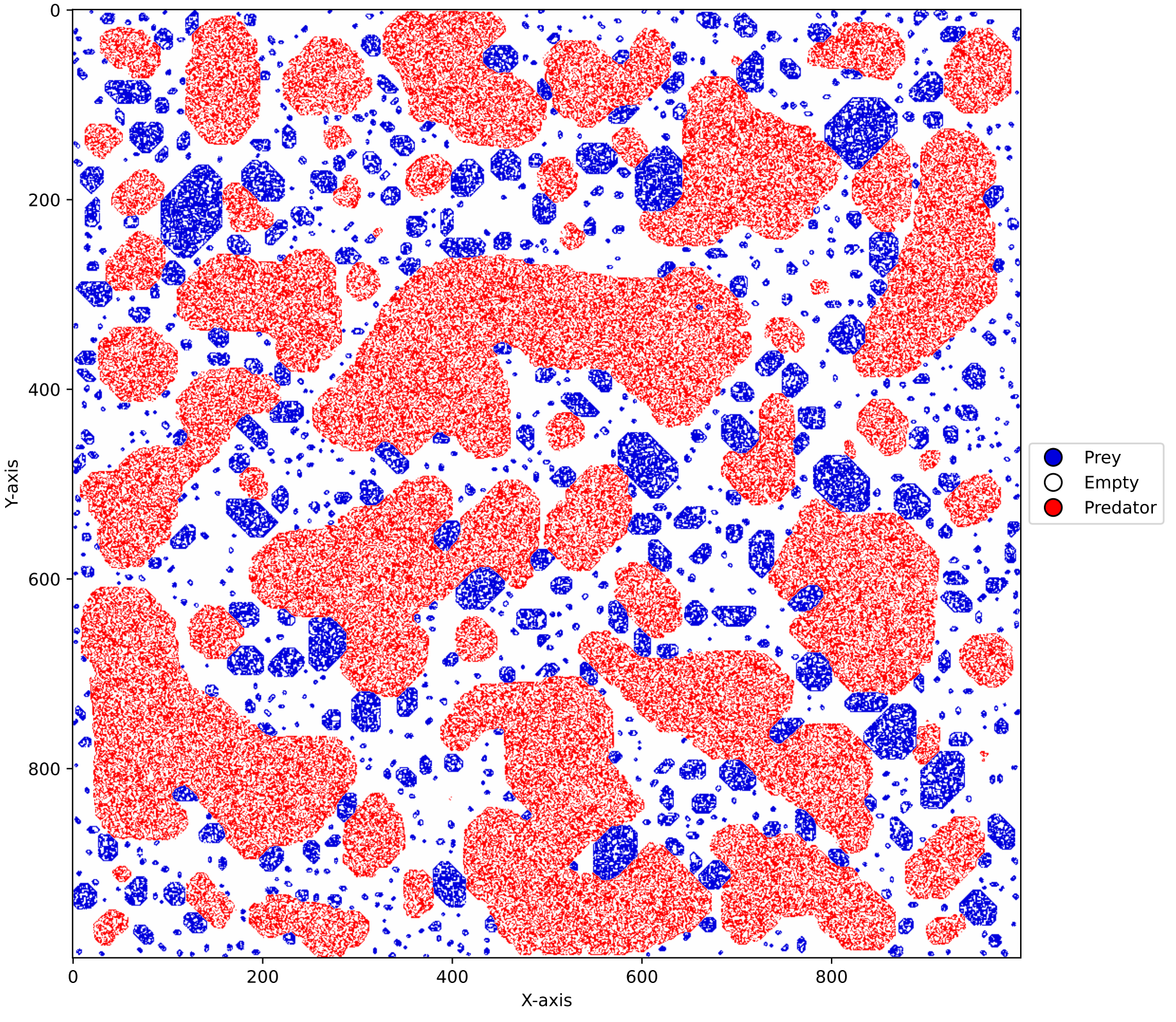
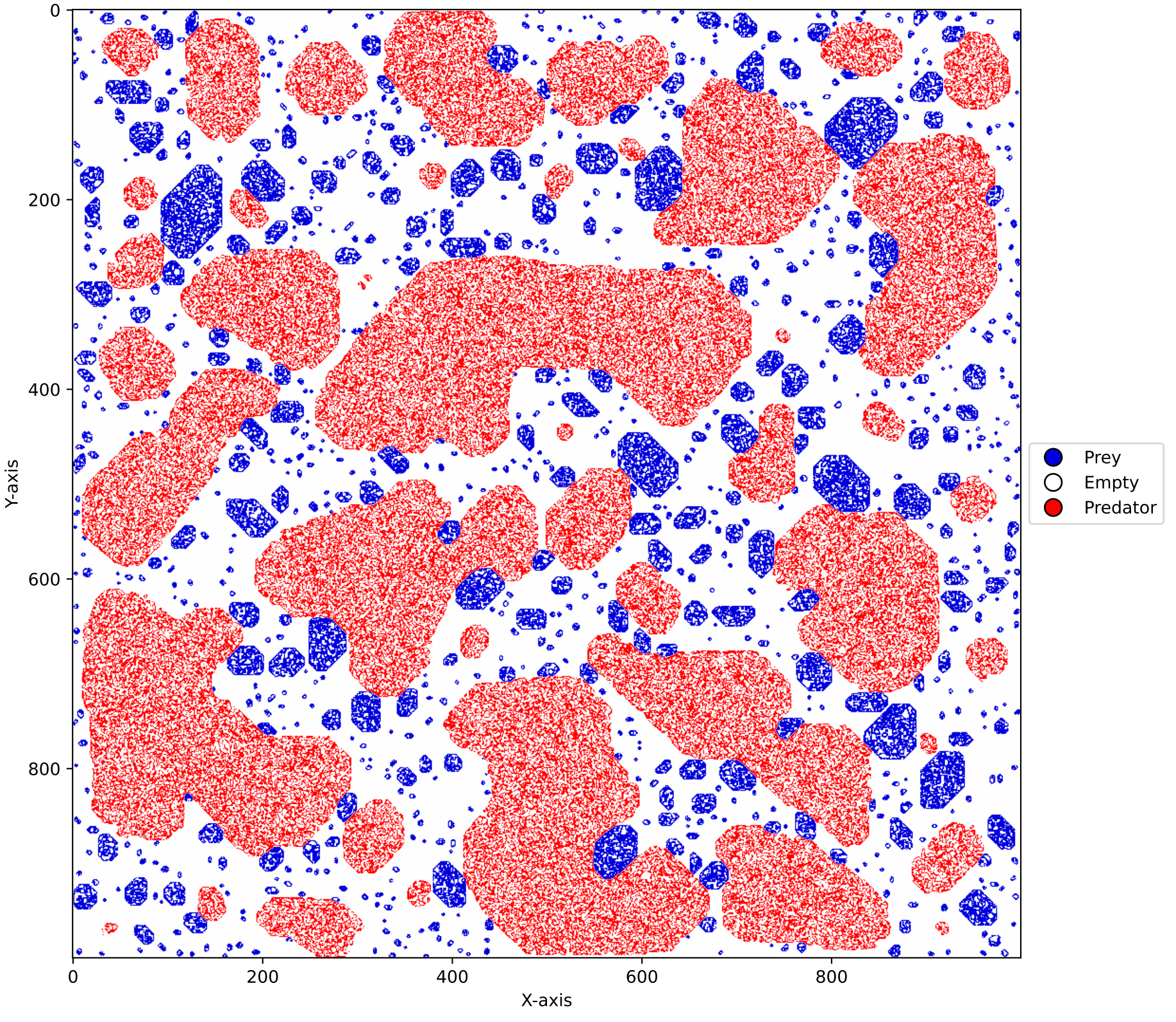
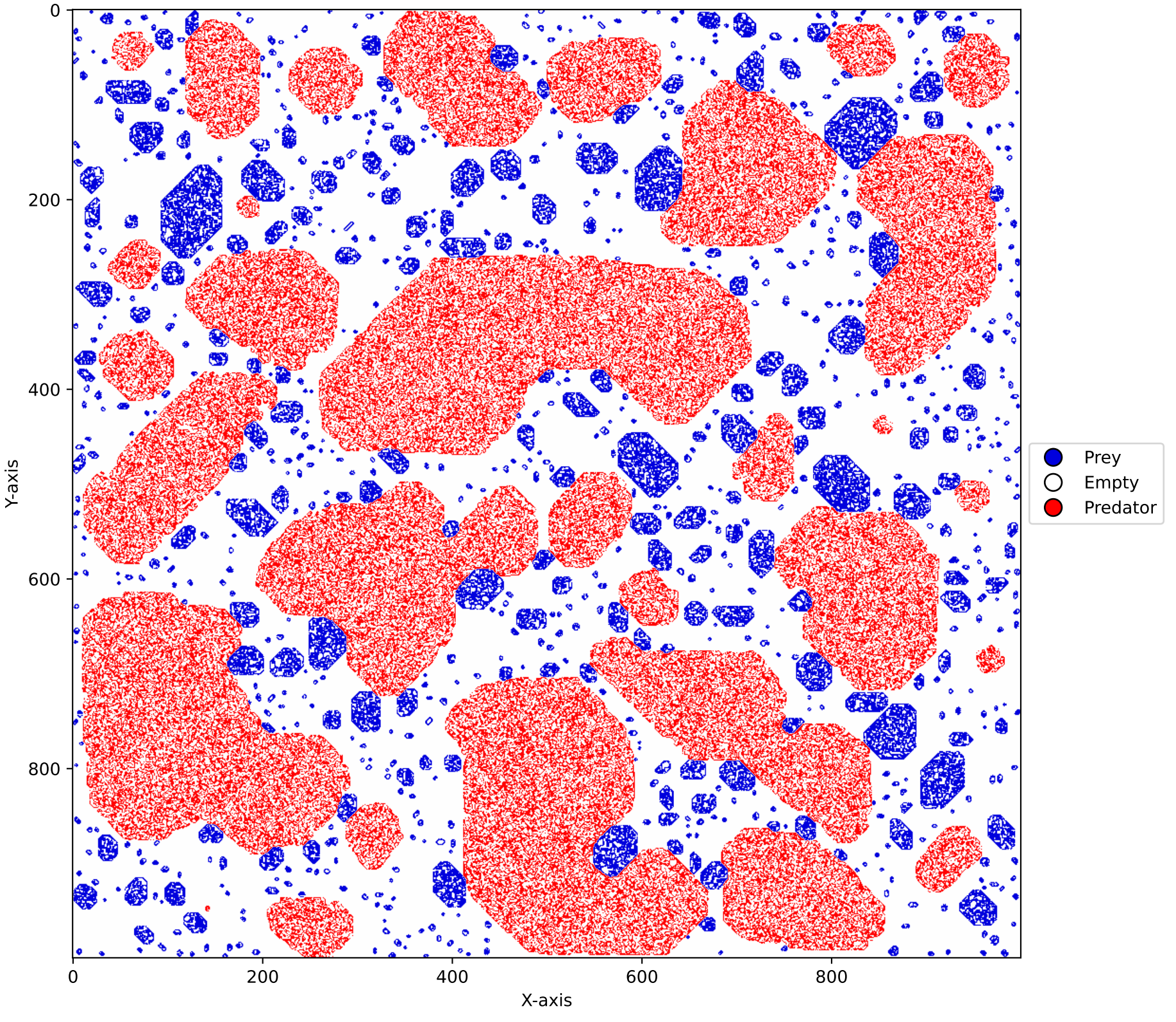
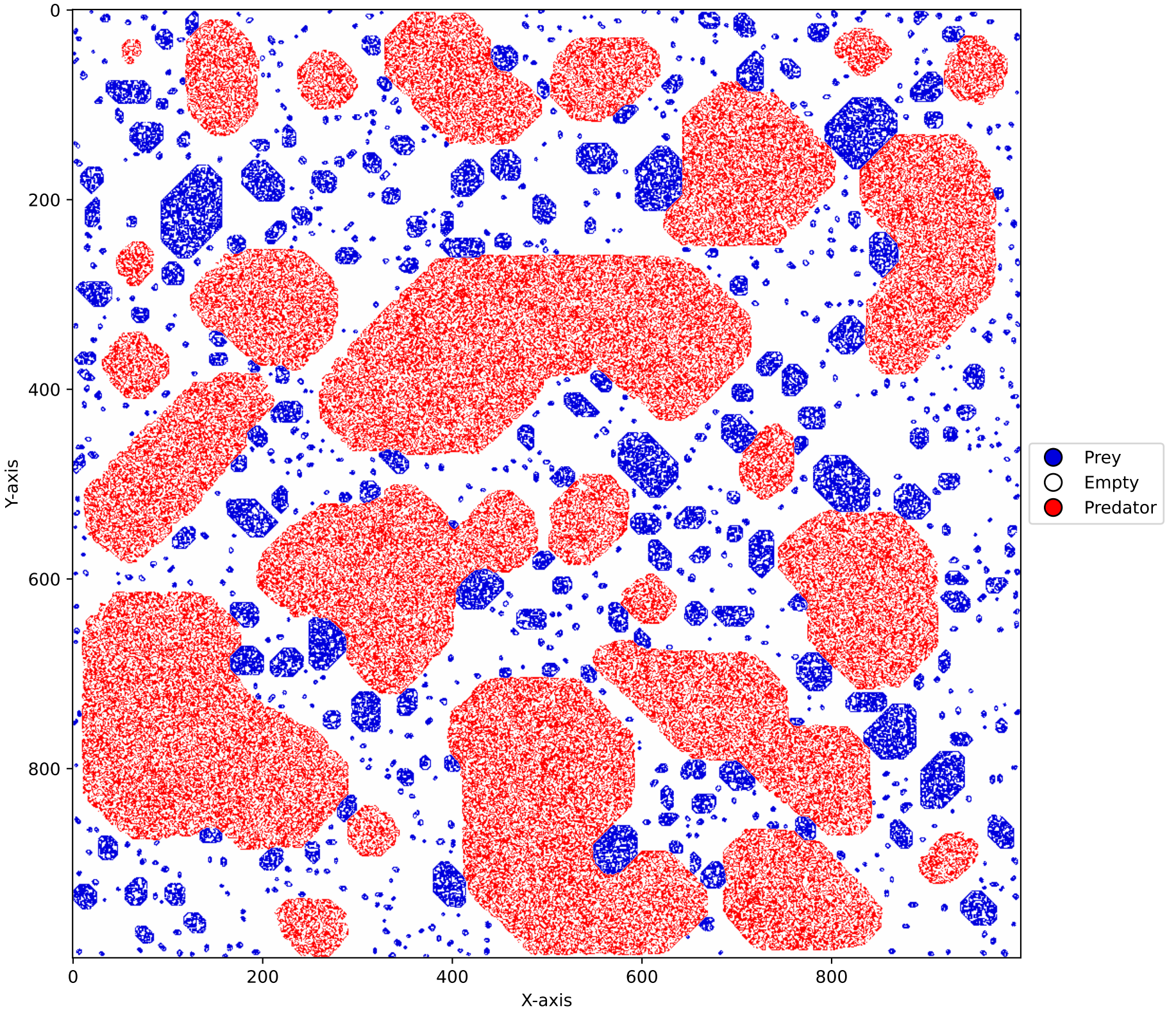

Generation 0 - 1000x1000 Grid
0
200
400
600
800
Y-axis
0
200
400
600
800
X-axis
Prey
Empty
Predator

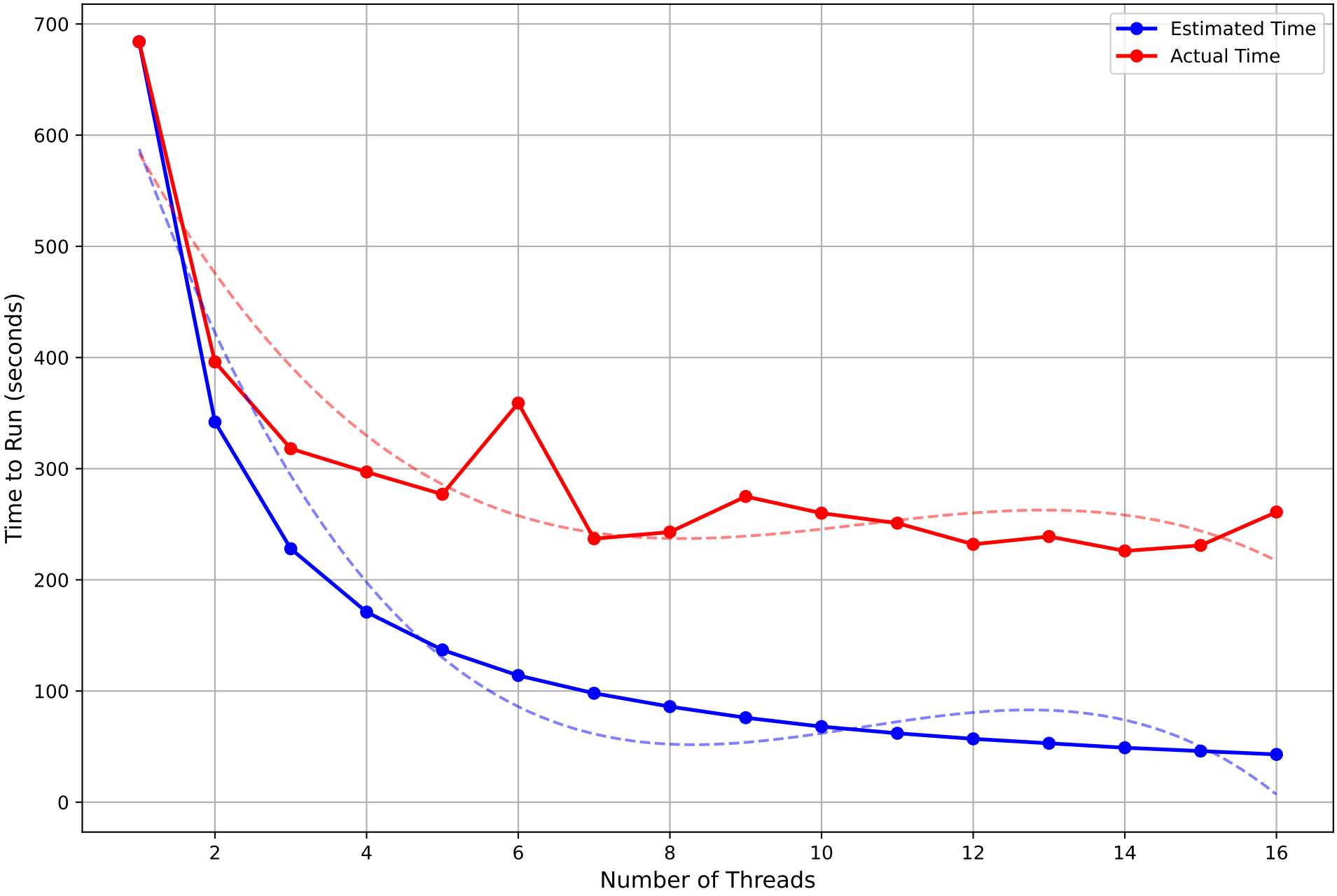

Generation 2000 - 1000x1000 Grid
0
200
400
600
800
Y-axis
0
200
400
600
800
X-axis
Prey
Empty
Predator


Generation 4000 - 1000x1000 Grid
0
200
400
600
800
Y-axis
0
200
400
600
800
X-axis
Prey
Empty
Predator


Generation 6000 - 1000x1000 Grid
0
200
400
600
800
Y-axis
0
200
400
600
800
X-axis
Prey
Empty
Predator


Generation 8000 - 1000x1000 Grid
0
200
400
600
800
Y-axis
0
200
400
600
800
X-axis
Prey
Empty
Predator


Generation 10000 - 1000x1000 Grid
0
200
400
600
800
Y-axis
0
200
400
600
800
X-axis
Prey
Empty
Predator


Estimated vs Actual Time to Run Predator-Prey Simulation
Estimated Time
Actual Time
Time to Run (seconds)
Number of Threads
0
100
200
300
400
500
600
700
2
4
6
8
10
12
14
16

Threaded Predator-Prey Simulation Report:

Visualization:

The grid was visualized using a custom color scheme:
  - Predators are represented by red.
  - Prey are represented by blue.
  - Empty cells are represented by white.

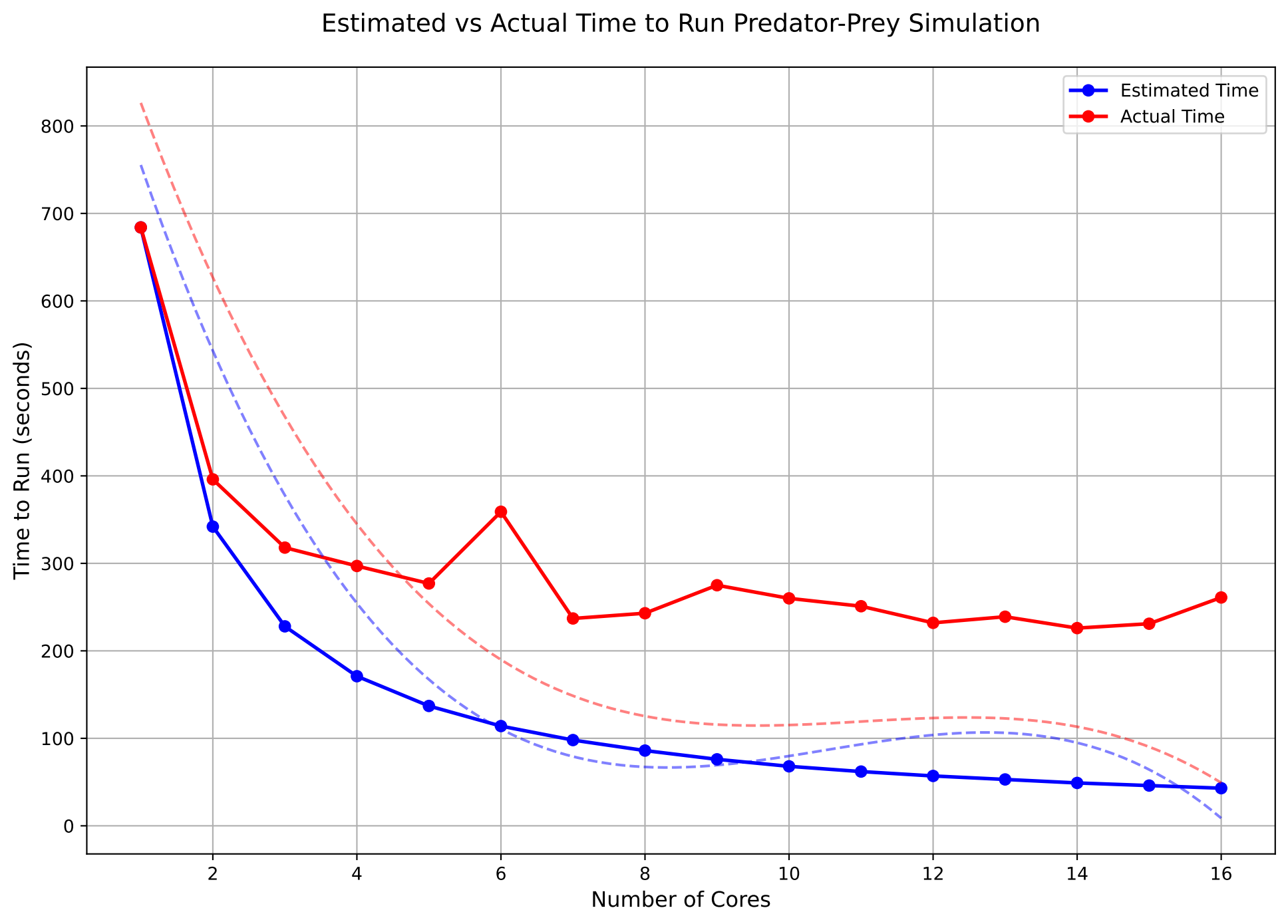
Hardware Used:

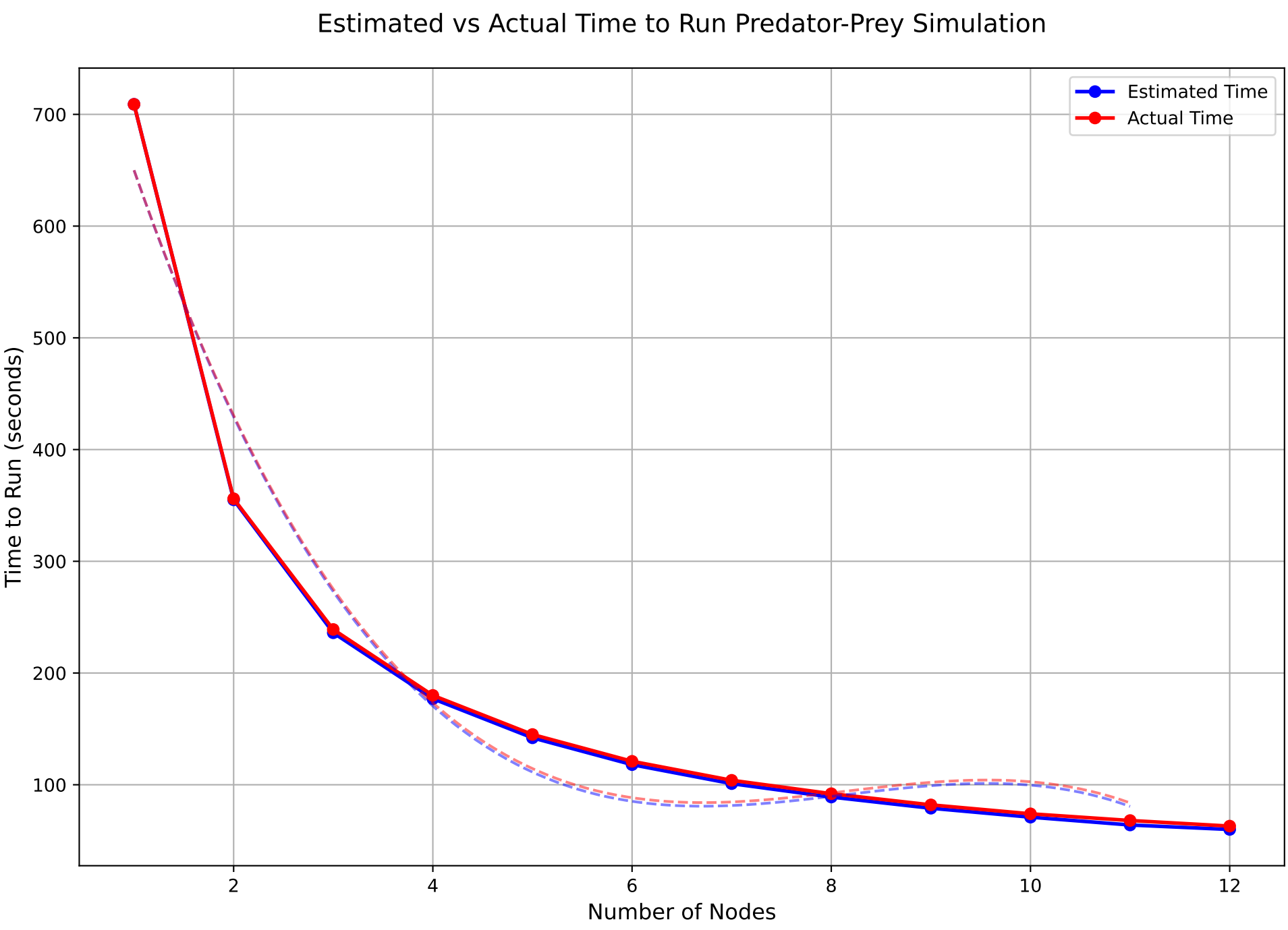
- CPU: <Intel i5 4-core>
- Memory: <16GB RAM>

Data:
---
1 thread: 684 seconds
2 threads: 396 seconds
3 threads: 318 seconds
4 threads: 297 seconds
5 threads: 277 seconds
6 threads: 359 seconds
7 threads: 237 seconds
8 threads: 243 seconds
9 threads: 275 seconds
10 threads: 260 seconds
11 threads: 251 seconds
12 threads: 232 seconds
13 threads: 239 seconds
14 threads: 226 seconds
15 threads: 231 seconds
16 threads: 261 seconds

Findings:

The results are as expected and similar to MPI-based simulations.
Running on a 4-core machine, the performance leveled off at around 4 threads, with diminishing returns beyond that point.


Estimated vs Actual Time to Run Predator-Prey Simulation
Estimated Time
Actual Time
Time to Run (seconds)
Number of Cores
0
100
200
300
400
500
600
700
800
2
4
6
8
10
12
14
16


Estimated vs Actual Time to Run Predator-Prey Simulation
Estimated Time
Actual Time
Time to Run (seconds)
Number of Nodes
700
600
500
400
300
200
100
2
4
6
8
10
12

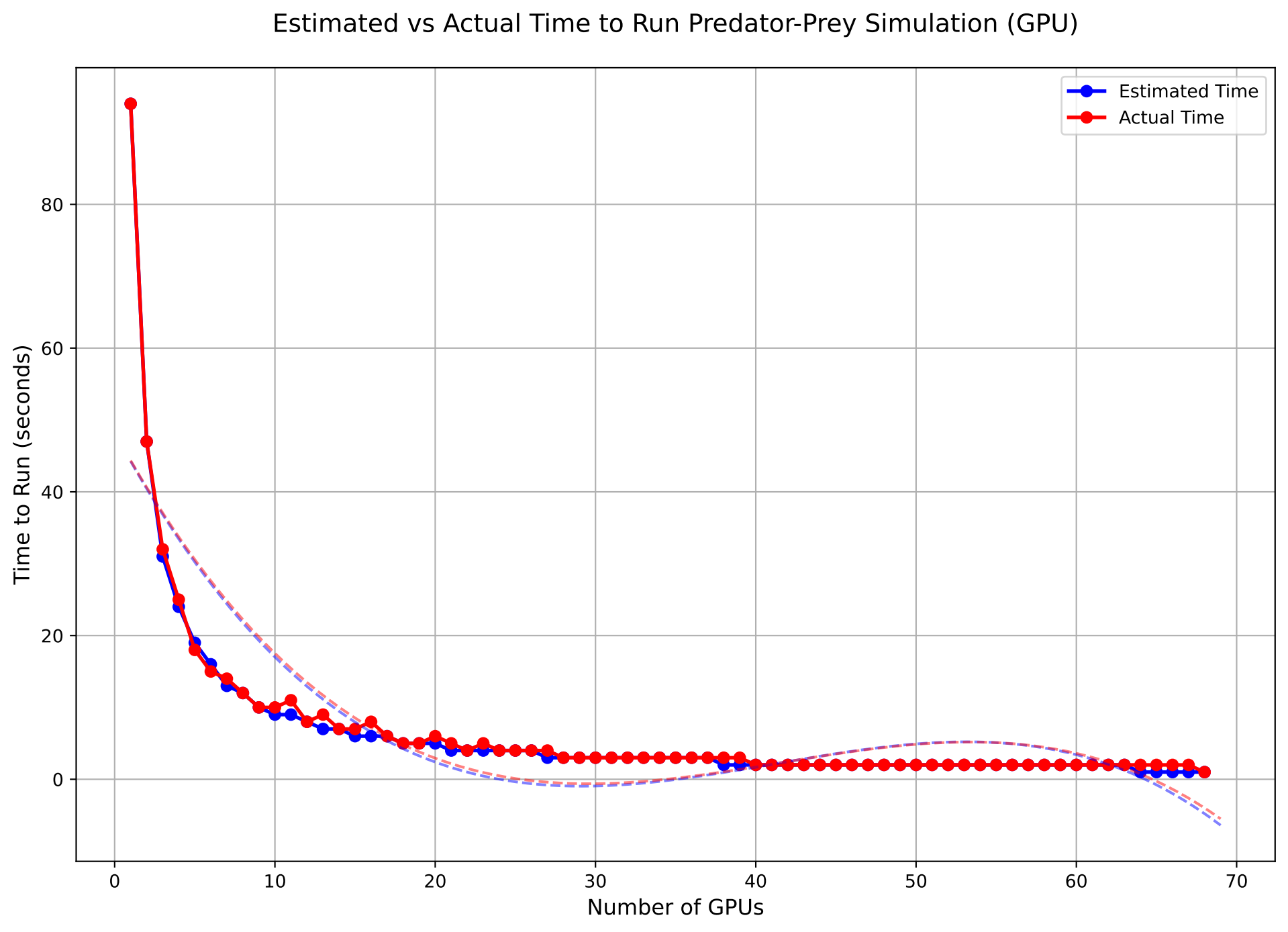
MPI Predator-Prey Simulation Report

Data
---
1 core: 879 seconds
2 cores: 611 seconds
3 cores: 407 seconds
4 cores: 311 seconds
5 cores: 249 seconds
6 cores: 209 seconds
7 cores: 179 seconds
8 cores: 155 seconds
9 cores: 139 seconds
10 cores: 125 seconds
11 cores: 113 seconds
12 cores: 103 seconds
13 cores: 95 seconds
14 cores: 90 seconds
15 cores: 86 seconds
16 cores: 81 seconds

1 node: 709 seconds
2 nodes: 356 seconds
3 nodes: 239 seconds
4 nodes: 180 seconds
5 nodes: 145 seconds
6 nodes: 121 seconds
7 nodes: 104 seconds
8 nodes: 92 seconds
9 nodes: 82 seconds
10 nodes: 74 seconds
11 nodes: 68 seconds

Findings:

The results are as expected and similar to the previous Monte Carlo Pi Simulation.
I actually was able to see improvements in runtime since the machines the code was run on actually had the capacity to increase runtime.
The nodes runtime was interesting because it stayed very consistent to the math, it was just slightly slower.


Estimated vs Actual Time to Run Predator-Prey Simulation (GPU)
Estimated Time
Actual Time
Time to Run (seconds)
Number of GPUs
0
20
40
60
80
0
10
20
30
40
50
60
70

CUDA Predator-Prey Simulation Report

Data:
---
1 GPU: 94 seconds
2 GPUs: 52 seconds
3 GPUs: 35 seconds
4 GPUs: 28 seconds
5 GPUs: 23 seconds
6 GPUs: 20 seconds
7 GPUs: 17 seconds
8 GPUs: 15 seconds
9 GPUs: 14 seconds
10 GPUs: 13 seconds
11 GPUs: 12 seconds
12 GPUs: 12 seconds
13 GPUs: 11 seconds
14 GPUs: 11 seconds
15 GPUs: 10 seconds
16 GPUs: 10 seconds
17 GPUs: under 10 seconds

Findings:

The GPU implementation showed significant performance improvements over CPU-based implementations. Even the first run was drastically shorter then every MPI run. It was also interesting that the actual runtime stayed close to the mathematical calculation. Overall GPU coding is much more preferable than MPI.