Samuel Fiedorowicz (se22m005)
Nguyen Ngoc Lam (se22m009)

Boids

Programmablauf

Es geht um einen Vogelschwarm mit einer fixen Anzahl an Vögeln. Die Vögel starten auf zufälligen Positionen innerhalb des definierten Feldes und haben eine zufällige Startgeschwindigkeit in einem definierten Rahmen.

Die Richtung und die Geschwindigkeit in der sich ein Vogel als Nächstes bewegen wird, berechnet sich aus drei Regeln:

- Der Vogel richtet sich zur gemeinsamen Mitte des Vogelschwarms aus
- Der Vogel passt seine Geschwindigkeit an die Durchschnittsgeschwindigkeit des Schwarms an
- Der Vogel darf das Feld nicht verlassen

Wobei das Gewicht jeder Regel (beispielsweise wie stark sich der Vogel zur Mitte des Vogelschwarms bewegt) einstellbar ist. Für jeden Vogel wird die neue Position mittels dieser 3 Regeln berechnet und dann die Bewegung durchgeführt.

```
foreach (Boid boid in Boids)
{
    boid.MoveTowardsGroup(Boids, factor: 0.0001f);
    boid.AdjustSpeedToGroup(Boids, factor: 0.01f);
    boid.AvoidCollisionWithWall(Width, Height, factor: 0.05f);
    boid.Move();
}
```

Abbildung 1: Serieller Programmablauf

Anmerkung: die Faktoren, die für die Anwendung der Regeln mitgegeben werden, sind die oben genannten Gewichte der Regeln.

Darstellung

Das Programm wurde mittels einer WPF-Anwendung visualisiert. Die WPF-Anwendung enthält eine Rendering-Loop, die bis zu 60-mal pro Sekunde ausgeführt wird. In dieser wird die nächste Bewegung jedes Vogels berechnet und dieser bewegt. In Abbildung 2 ist Visualisierung zu sehen. Unten in der Mitte werden die FPS angezeigt.

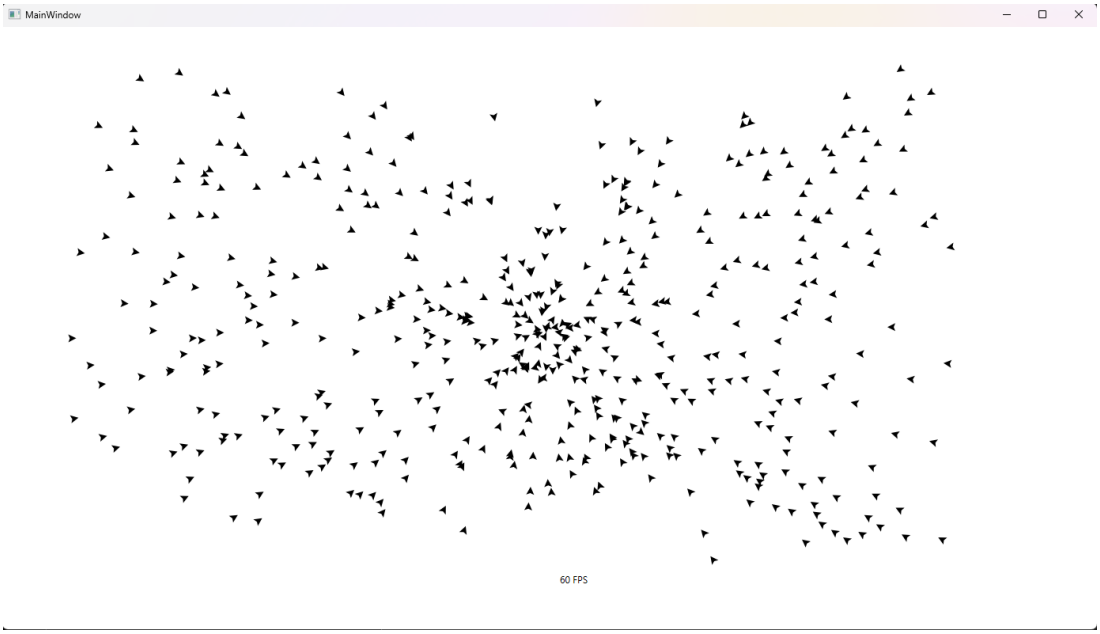

Abbildung 2: Visualisierung eines Vogelschwarms mit 500 Vögeln

Parallelisierungsansatz

Um das Rendering performanter zu gestalten, werden die Positionen der Vögel parallel berechnet. Die parallele Berechnung wird mit `Parallel.ForEach` durchgeführt. Durch diese Verbesserung soll es bei einer höheren Vogelanzahl möglich sein, mehr FPS zu erreichen als bei der seriellen Variante.

```
Parallel.ForEach(Boids, body: boid =>
                        {
                            boid.MoveTowardsGroup(Boids, factor: 0.0001f);
                            boid.AdjustSpeedToGroup(Boids, factor: 0.01f);
                            boid.AvoidCollisionWithWall(Width, Height, factor: 0.05f);
                            boid.Move();
                        });
```

Abbildung 3: Paralleler Programmablauf

Messansatz FPS

Hardware: i7 3770k @3.5Ghz, 4 Cores, 8 Threads, GTX 980ti

Um die Performance der parallelen mit der seriellen Lösung zu vergleichen, wurden die FPS bei einer höheren Vogelanzahl verglichen. In der unteren Tabelle sind die durchschnittlichen FPS über 15 Sekunden bei unterschiedlicher Schwarmgröße für die parallele und serielle Variante ersichtlich. Die ersten 5 FPS auf Grund des Warmups verworfen. Anzumerken ist, dass die maximalen FPS auf Grund der Rendering-Loop auf die Frequenz des Monitors limitiert ist.

Vogelscharmgröße	FPS (seriell)	FPS (parallel 4)	FPS (parallel 8)	Speedup (FPSp/FPSs)
500	102,77	120,55	120,66	1,17
1000	34,55	65,88	75,42	2,18
1500	17,66	36,22	43,11	2,44
2000	10,88	24,33	30	2,76

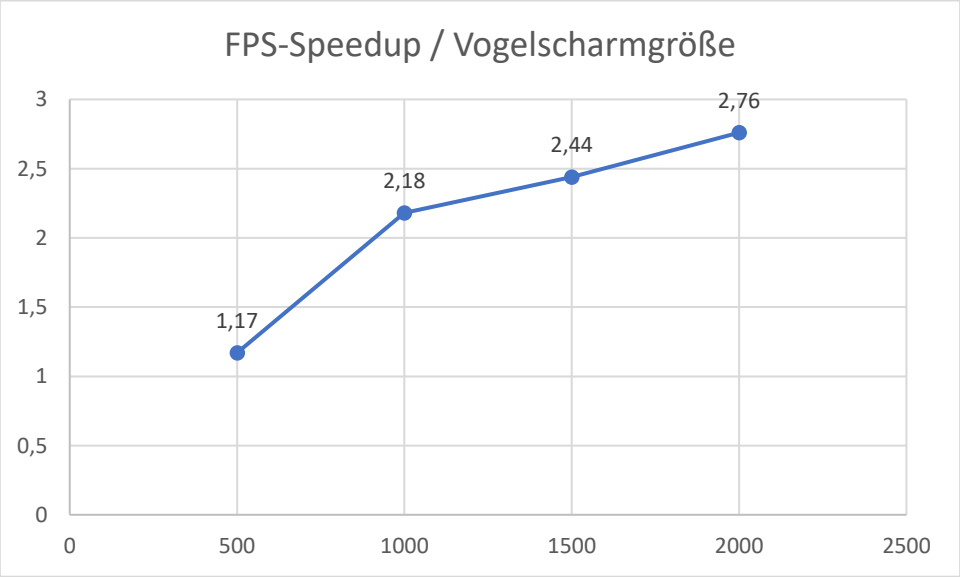

Abbildung 4: FPS Erhöhung (y-Achse) in Bezug zu Vogeschwarmgröße (x-Achse) bei paraleller statt serieller Berechnung

Messansatz Performance

Hier wurde mithilfe von Benchmark.Net ein Berechnungsdurchlauf für die verschiedenen Vögel anzahlen gemessen. Wir sehen, dass je mehr Vögel da sind, desto größer wird der Performance gewinn, jedoch steigt die Performance nicht ins unendliche, sondern nähert sich dem 5-Fachen an.

Der theoretische Performancegewinn vom 8-Fachen durch die 8 Threads ist real nicht erreichbar, da das Problem nicht 100% parallelisierbar ist.

```
// * Summary *

BenchmarkDotNet=v0.13.5, OS=Windows 11 (10.0.22000.1696/21H2/SunValley)
Intel Core i7-3770K CPU 3.50GHz (Ivy Bridge), 1 CPU, 8 logical and 4 physical cores
.NET SDK=7.0.202
  [Host]     : .NET 7.0.4 (7.0.423.11508), X64 RyuJIT AVX
  Job-PSADJN : .NET 7.0.4 (7.0.423.11508), X64 RyuJIT AVX

IterationCount=5  LaunchCount=1  RunStrategy=Throughput
WarmupCount=3

|     Method | boidCount |            Mean |         Error |        StdDev |    Gen0 |  Allocated |
|----------- |---------- |----------------:|--------------:|--------------:|--------:|-----------:|
|     Serial |       100 |       235.84 us |     58.019 us |     15.067 us |  3.4180 |   14.24 KB |
| Parallel_4 |       100 |        92.67 us |     10.322 us |      2.681 us |  6.1035 |   24.75 KB |
| Parallel_8 |       100 |        81.07 us |      2.003 us |      0.520 us |  7.4463 |   29.62 KB |
|     Serial |      1000 |    19,472.30 us |    752.056 us |    116.381 us | 31.2500 |   133.8 KB |
| Parallel_4 |      1000 |     5,530.30 us |    409.747 us |    106.410 us | 31.2500 |  144.36 KB |
| Parallel_8 |      1000 |     4,834.65 us |    652.256 us |    169.389 us | 31.2500 |  149.41 KB |
|     Serial |      2000 |    77,095.60 us |  1,537.335 us |    399.241 us |       - |  267.08 KB |
| Parallel_4 |      2000 |    21,983.37 us |  6,657.718 us |  1,728.988 us | 62.5000 |  277.59 KB |
| Parallel_8 |      2000 |    18,136.14 us |  1,436.239 us |    372.987 us | 62.5000 |  282.73 KB |
|     Serial |      5000 |   483,687.82 us | 19,811.959 us |  5,145.103 us |       - |  715.19 KB |
| Parallel_4 |      5000 |   131,152.28 us |  7,092.103 us |  1,841.797 us |       - |  725.44 KB |
| Parallel_8 |      5000 |   109,153.90 us |  5,448.610 us |    843.178 us |       - |   730.6 KB |
|     Serial |     10000 | 1,889,525.36 us | 28,263.751 us |  7,340.006 us |       - | 1429.15 KB |
| Parallel_4 |     10000 |   511,563.24 us | 27,155.023 us |  7,052.073 us |       - | 1440.31 KB |
| Parallel_8 |     10000 |   428,300.83 us |  5,799.863 us |    897.535 us |       - | 1445.53 KB |
|     Serial |     20000 | 7,528,010.58 us | 94,459.569 us | 24,530.850 us |       - | 2857.05 KB |
| Parallel_4 |     20000 | 1,947,457.90 us | 71,727.196 us | 18,627.325 us |       - | 2871.15 KB |
| Parallel_8 |     20000 | 1,702,856.06 us |  5,263.833 us |  1,367.001 us |       - | 2874.74 KB |
```

Samuel Fiedorowicz (se22m005)
Nguyen Ngoc Lam (se22m009)