The 10th Conference on Mathematics in Engineering: Theory and Applications
Novi Sad, December 12-14th 2025

EDUCATIONAL VISUALIZATION OF THE BOIDS ALGORITHM IN UNREAL ENGINE'S NIAGARA SYSTEM[1]

Lana Dujmović[2] , Ana Perišić[2] and Nebojsa Ralević[2]

https://doi.org/10.24867/META.2025.07 Professional article

Abstract. Graphical representations of mathematical concepts often tend to appear too abstract and disconnected from intuitive understanding. To bring these principles closer to students, this work explores the simulation of flocking behavior based on the Boids model, implemented through the Niagara Particle System in Unreal Engine. By utilizing procedural functions to define local interaction rules, such as alignment, cohesion, and separation, the study demonstrates how mathematical ideas underlying motion dynamics can be visualized in real time. The research is envisioned as a foundation for future applications within courses such as Simulation in Animation, Engineering Animation and Other Media, and Design of 3D Space and Environments, highlighting the value of visual and interactive approaches in understanding algorithmic and physical systems.

AMS Mathematics Subject Classification (2020): 00A72, 37M05

Key words and phrases: simulation, Unreal Engine, Niagara Particle System, applied mathematics, boids algorithm

1. Introduction

The Boids algorithm (short for "bird-oid objects") represents a groundbreaking concept in computer graphics and Artificial Life. Developed in 1986 by Craig Reynolds, Boids introduced a distributed behavioral model for simulating collective motion observed in flocks of birds, schools of fish, or herds of animals. Its primary goal was to achieve visually convincing and computationally efficient procedural animation, eliminating the need for laborious manual scripting of individual object paths. The significance of the model lies in its demonstration of emergent behavior, complex, synchronized group motion arising solely from simple local interaction rules applied to autonomous agents [1][2]. Modern game engines such as Unreal Engine 5 (UE5) enable

[1]This research is partially supported by the Science Fund of the Republic of Serbia, #GRANT no. 7632, the project "Mathematical Methods in Image Processing under Uncertainty" MaMIPU and by the project "Socio-technological aspects of improving the teaching process in the English language in fundamental disciplines"

[2]Department of Fundamental Sciences, Faculty of Technical Sciences, University of Novi Sad, e-mail: lanad@uns.ac.rs, anaperisic@uns.ac.rs, nralevic@uns.ac.rs

both visual and procedural, implementation of such systems through the Niagara particle framework providing the capability for real-time simulation of complex behaviors within virtual environments This paper explores the implementation of emergent behavior within a modern VFX context by integrating the Boids model and the Niagara particle system in Unreal Engine [5]. Focusing on the visualization of mathematical and procedural principles, it illustrates applied mathematics through creative simulation, with potential uses in games, interactive media, and intelligent animation systems related to courses such as *Simulation in Animation*, *Engineering Animation and Other Media* and *Design of 3D Space and Environments*.

2. Theoretical Framework and current implementations

For further understanding of this paper, it will be necessary to define the problems that will be analysed. A brief overview of all important concepts will be given in this chapter, as well as some of the recent implementations and visualisations.

2.1. Mathematical Model of the Boids System

In his seminal paper "Flocks, Herds, and Schools: A Distributed Behavioral Model" (1987), Craig Reynolds defined three simple behavioral rules that govern how each agent, boid, moves through space. These rules form the foundation of emergent group behavior, showing how coordinated structures arise solely from local interactions, without any central control or global plan [1].

Mathematically, the position and velocity of the i-th boid at time t are represented as:

$$\vec{x}_i(t), \quad \vec{v}_i(t)$$

The evolution of the system is governed by a coupled set of differential equations:

$$\frac{d\vec{x}_i(t)}{dt} = \vec{v}_i(t), \qquad \frac{d\vec{v}_i(t)}{dt} = \mathbf{F}_i^{\text{sep}} + \mathbf{F}_i^{\text{ali}} + \mathbf{F}_i^{\text{coh}} \tag{2.1}$$

where the three components represent local behavioral rules:

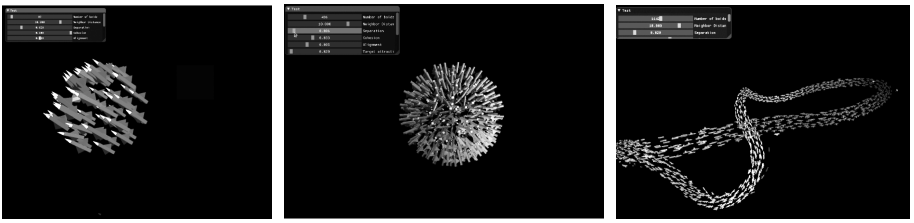
Separation (F_i^{sep}) - Each boid tends to avoid getting too close to its neighbors. Reynolds describes this as the avoidance of collisions with nearby flockmates. The rule ensures that individuals maintain a minimal distance, creating a natural spread within the flock and preventing overlaps [2].

Alignment (F_i^{ali}) - A boid aligns its direction and velocity with those of its nearby flockmates (matching velocity with nearby flockmates). This produces synchronized, fluid motion characteristic of real flocks or schools moving cohesively in one direction [2].

Cohesion (F_i^{coh})- Each boid seeks to remain part of the group by moving toward the average position of its neighbors (tendency to move toward the average position of local flockmates). This keeps the flock spatially compact and prevents dispersion [2].

2.2. Implementation using C++ / OpenGL

This is a practical, low-level implementation of the Boids algorithm built with C++, OpenGL, and Dear ImGui, focusing on developing a high-performance and fully controllable flocking simulation from the ground up. The algorithm follows Reynolds's three behavioral rules (alignment, separation, and cohesion) where each agent calculates forces based on its neighbors and updates its position and velocity in every frame (Figure 1). The simulation integrates these rules through a weighted system of parameters (for example, α, β, γ coefficients), enabling interactive adjustment of flock dynamics in real time [3].



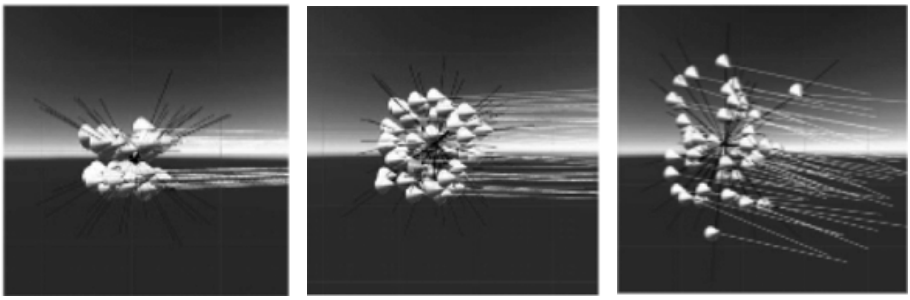
(a) (b) (c)

Figure 1: Visualization of Boids in OpenGL: (a) initialization, (b) flock formation, (c) emergent flow [3].

While the system produces smooth and realistic flocking behavior, boids are rendered as simple geometric shapes. It also requires solid programming skills in C++ and a basic understanding of the graphics pipeline. This makes the implementation useful for demonstrating the mathematical and computational principles behind emergent behavior and serves as a reference for higher-level frameworks like Unreal Engine's Niagara system, where similar concepts can be applied without low-level coding.

2.3. Implementation in Unity

In "Simulation of Boid Type Behaviours in Unity Environment" (2021), the authors implemented Reynolds's three behavioral rules, separation, alignment, and cohesion, directly within Unity's 3D environment using C# scripting and the built-in physics system [4]. Each boid was represented as a lightweight particle or simple 3D object that continuously updated its position and velocity based on nearby agents detected within a fixed perception radius (Figure 2). This neighborhood detection was optimized using Unity's spatial query functions to ensure real-time performance even for larger swarms. The visualization made use of Unity's Scene View and Game View for simultaneous inspection of both algorithmic behavior and rendered motion. The authors noted that the emergent flocking patterns, turning, grouping, and smooth avoidance, were "visually convincing and stable across various population densities."



(a) (b) (c)

Figure 2: Visualization of a BOID group in the Unity environment. The Reynolds forces (separation, alignment, and cohesion) were parameterized through a frequency coefficient. Subfigures (a), (b), and (c) illustrate the resulting variations in behavior as this coefficient is adjusted [4].

While the presented models are relatively simple and computationally efficient, it remains an open question how the same system would perform with complex, animated meshes, where per-agent updates could significantly increase processing cost.

3. Applications in UE 5.6

This chapter presents the visualization and simulation of the Boids algorithm, specifically the flocking behavior of birds and fish, developed in Unreal Engine 5.6 using the Niagara particle system (see Figures 3 and 4).

While the Unity implementation relied on object-based simulation, the Niagara framework treats each boid as an individual particle instance governed by user-defined behavior modules. These modules are constructed through node-based scripts, allowing procedural control of particle position, velocity, and interaction directly within the editor's visual environment. Each particle operates as an autonomous agent that continuously evaluates its surroundings according to Reynolds's three behavioral rules: separation, alignment, and cohesion. The system utilizes GPU-accelerated rendering, which supports higher population counts and smoother real-time performance compared to CPU-based approaches, offering an efficient platform for studying emergent motion.

3.1. Mathematical Vector-Based Forces in Niagara

In Unreal Engine's Niagara system, particle motion is defined through vector-based mathematical functions acting in three-dimensional space. These functions modify particle position and velocity via transformations that reflect physical or behavioral principles such as turbulence, drag, attraction, and rotation. Each force represents a vector field $\vec{F}(\vec{r}, t)$ influencing motion according to

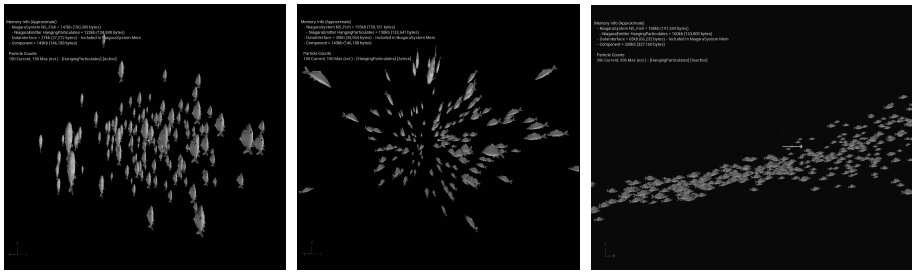
$$\frac{d\vec{v}}{dt} = \frac{\vec{F}(\vec{r}, t)}{m}. \tag{3.1}$$

By combining several operators, Niagara reproduces complex emergent behaviors from simple local interactions, aligning with Reynolds's flocking dynamics model.

(a) (b) (c)

Figure 3: Visualization of bird flocking behaviour (a) initialization, (b) flock formation, (c) emergent flow.

The *Curl Noise Force* adds spatially coherent turbulence by applying the curl of a 3D noise field. *Drag* simulates viscous damping proportional to velocity, ensuring smooth motion. *Point Attraction Force* drives particles toward a target, modeling cohesion or goal-seeking behavior. *Add Velocity* integrates multiple computed influences into a single motion update. *Vortex Force* induces rotational flow around an axis, and *Point Force* generalizes attraction and repulsion for obstacle avoidance or spatial structuring [5].



(a) (b) (c)

Figure 4: Visualization of fish shoaling behaviour (a) initialization, (b) shoaling formation, (c) emergent flow.

Together, these modules illustrate how Niagara's procedural system transforms mathematical vector operations into visually coherent simulations of collective motion, merging physics-based modeling with real-time graphics.

3.2. Discussion and observation

Compared to object-oriented implementations in Unity and low-level C++/OpenGL simulations, Niagara provides a higher degree of procedural

flexibility and visual realism. The node-based system abstracts the underlying mathematics while maintaining control over physical accuracy, enabling rapid experimentation with large agent populations. Its GPU-driven architecture supports complex emergent behavior without significant performance loss, demonstrating how mathematical concepts can be effectively visualized through modern VFX workflows. Future applications may extend toward interactive scenarios, such as Blueprint-driven character responses or virtual crowd visualization, where behavioral rules can dynamically adapt through fuzzy or AI-based control schemes [2].

4. Conclusion

Overall, the Niagara implementation confirms that Unreal Engine 5 provides a robust foundation for real-time visualization of emergent systems, bridging scientific simulation with cinematic-quality rendering. The resulting workflow illustrates how behavioral algorithms such as Boids can evolve from purely computational experiments into expressive, interactive, and visually rich simulations. The research is expected to support future work in courses such as Simulation in Animation, Engineering Animation and Other Media, and Design of 3D Space and Environments, by providing a framework for demonstrating motion simulations through the Niagara particle system. This system relies on vector algebra, local-space transformations, and particle lifecycle management, offering students a more intuitive and visual approach to understanding algorithmic and physical motion.

References

[1] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *Computer Graphics (SIGGRAPH '87 Conference Proceedings)*, vol. 21, no. 4, pp. 25–34, 1987.

[2] I. L. Bajec, M. Mraz, and N. Zimic, "The computational beauty of flocking: Boids revisited," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 13, no. 4, pp. 331–347, Aug. 2007.

[3] M. Grinberg, "Mastering Flock Simulation with Boids, C++, OpenGL, and ImGui," *Better Programming*, Medium, 2020. Available at: https://medium.com/better-programming/mastering-flock-simulation-with-boids-c-opengl-and-imgui-5a3ddd9cb958.

[4] T. Lypovyi and J. Montusiewicz, "Simulation of Boid Type Behaviours in Unity Environment," *International Journal of Computer Science and Engineering*, vol. 9, pp. 1–10, 2021.

[5] Epic Games, "Getting Started in Niagara Effects for Unreal Engine," *Unreal Engine Documentation*, Epic Games, 2024. Available at: `https://dev.epicgames.com/documentation/en-us/unreal-engine/getting-started-in-niagara-effects-for-unreal-engine`.