Boids: learning vector arithmetic through animation

Neil A. Dodgson
Department of Engineering and Computer Science
Victoria University of Wellington
neil.dodgson@vuw.ac.nz

Joshua Scott
Department of Engineering and Computer Science
Victoria University of Wellington
joshua.scott@ecs.vuw.ac.nz

ABSTRACT

Boids is an excellent example of emergent behaviour. Coding some simple rules creates complex behaviour. The assignment consolidates students' learning of C++, OpenGL, GLM, and vector arithmetic. Students also learn about the careful balances that must be made to ensure that a simulation behaves in a realistic way.

CCS CONCEPTS

• **Computing methodologies** → *Physical simulation*; *Collision detection*;

KEYWORDS

education,animation,graphics,simulation

ACM Reference Format:
Neil A. Dodgson and Joshua Scott. 2018. Boids: learning vector arithmetic through animation. In *Proceedings of SIGGRAPH '18 Educator's Forum.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3215641.3215654

1 INTRODUCTION

The student implements the boids flocking algorithm, originally developed by Craig Reynolds [Reynolds 1987]. Each boid is controlled independently but the behaviours programmed into the algorithm mean that group behaviour emerges from the individual behaviours. The later stages of the assignment build extra features on top of Reynolds' basic algorithm. One advantage of this assignment is that there are numerous ways to interpret the implementation of various parts of the algorithm, giving students ability to explore ideas rather than slavishly following a recipe. The use of boids as a programming assignment is not original. Others have used variations on this idea [Greening 2000; Iba 2008].

The assignment is designed to consolidate students' understanding of C++ and OpenGL programming, to give them practical training in implementing vector arithmetic using the GLM library, and to provide experience of emergent behaviour and how to tune parameters to get the desired behaviour.

2 MATERIALS

Students are given three lectures that cover motivation, the basic boids algorithm, several extensions including ray/sphere intersection, and all of the vector arithmetic. The lecture handout contains links to a range of online resources, including Craig Reynolds' archival website https://www.red3d.com/cwr/boids/, which contains several essays on various extensions to the algorithm.

Table 1: CGEMS metadata

Summary	Implement the boids algorithm, extending it to multiple flocks, predators, and obstacle avoidance.
Learning outcomes	how to handle vector algebra, how to implement force-velocity-position physics equations how to to implement appropriate data structures in C++; to discover the checks and balances that must be built in to make a simulation effective.
Classification	(1) Animation, (3) Fundamentals
Audience	CS2
Dependencies	basic C++ programming, first-year linear algebra
Strengths	relatively straightforward, students enjoy creating something that has emergent behaviour, the mathematics lead directly on to ray tracing
Weaknesses	the best students find insufficient stretch
Variants	there are many ways to add features to the boids algorithm
Assessment	a set of binary achieved/unachieved criteria plus some discretionary marks for the quality of the programming, specific numeric penalties are applied for certain common bugs, students are marked in person where they must demonstrate their software working and answer questions

A C++ framework is provided, which handles the basic drawing and has a single example boid object. The framework is cross-platform and self-contained. It has been tested on Linux, Windows and Mac OS. Students are provided with supporting documentation to allow them to get the framework running on any one of these three platforms. This provides substantial flexibility for the students over a framework that runs only on the University's specific architecture (in our case, Linux). The framework and structure provides most of the ground work, allowing students to get straight to the implementation of the primary concept of boids, meaning that we can safely set more complex problems without having to worry about students getting stuck on the basics.

In our university, students are supported by tutors who offer weekly tutorial sessions, help via an online forum and a weekly one-hour in-person help desk. We prefer assessment to be done in person, with the student demonstrating and explaining their work.

3 STRUCTURE

This assignment has three sections: core (basic algorithm), completion (multiple flocks and predators) and challenge (large obstacle avoidance). A C-grade student would be expected to complete the

SIGGRAPH '18 Educator's Forum, August 12-16, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5884-2/18/08.
https://doi.org/10.1145/3215641.3215654

core to a good standard. A B-student will complete core and completion. An A-student is expected to complete all three parts.

3.1 Core assignment

The core is to implement the basic algorithm, which is outlined in Reynolds paper [Reynolds 1987] and for which the mathematics is given in the lecture handout.

Outline specification. Create a single flock of boids. Each boid will exhibit the three standard boid behaviours: *avoidance*, steer to avoid crowding local flockmates; *alignment*, steer towards the average heading of local flockmates; *cohesion*, steer to move towards the average position of local flockmates. In addition, it will exhibit two further behaviours: *confinement*, keep the boids within a particular area, so that boids stay on the screen rather than fly into the distance; *speed limitation*, keep the boid's speed between a minimum and a maximum, to emulate a real bird that cannot fly slower or faster than certain speeds.

Comments. Students should start by getting a single boid to work with the latter two behaviours, then extend this to a small group, then to a flock of 100–300 boids exhibiting all five behaviours. We ask the students to provide controls (e.g., sliders on screen) that allow them to control the strengths of the avoidance, alignment, and cohesion behaviours, and the size of "local", which is the distance that a boid can sense the positions and headings of its local flockmates. We prefer students to use interactive controls, rather than have to recompile with new parameter settings, because this allows them to experiment easily with parameter tuning, giving them a good understanding of how the simulation responds to different weightings.

3.2 Completion assignment

The student "completes" the assignment by adding to the code to produce more than one flock and a predator. A good implementation of "completion" is sufficient to achieve a high B-grade.

Outline specification. Create two flocks of boids, distinguished by different colours. A boid will *avoid* all other local boids but it will *align* and *cohere* only with local boids in its own flock. Add a predator to the simulation. The predator frightens the boids and also tries to capture a boid.

Comments. Students need to change the boids' behaviour so that there is an extra force that drives them away from the predator. They need to think about various implementation issues here, which are left open for their interpretation: how far away should a boid be able to perceive a predator? what changes are needed to boid behaviour to get predator behaviour? how does a predator target its prey? what tracking algorithm does it use? does a predator fixate on a single boid and pursue it to the exclusion of all others? or does it switch attention between boids?

In our lectures, we show students video of real predator behaviour, showing that certain real predators cannot catch a bird in a flock because their inherent tracking mechanisms appear to keep switching between targets. We also tie the pursuit behaviour to ball games, both player-intercept in football and ball-catching in baseball. We want the students to think about what algorithms a human "implements" in their brain.

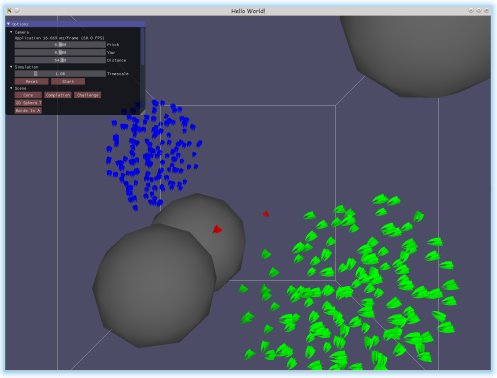

Figure 1: A successful challenge assignment, with two flocks (green and blue), two independent predators (red), and three obstacles.

3.3 Challenge assignment

The stretch challenge for the best students is to implement object avoidance.

Outline specification. Create other objects for the boids to avoid that are significantly larger than a single boid so the simple boid avoidance method will not work.

Comments. Students are expected to make spherical objects that the boids must avoid. They are taught ray/sphere intersection, which leads neatly into the ray tracing part of the course, where students have to reuse most of the vector arithmetic that they have learnt in the boids algorithm. To demonstrate this to the assessor, students are expected to show a simulation with at least three large objects, for example, three or more large spheres.

4 STUDENT PERFORMANCE

Students had three weeks to complete the assignment, which was worth 20% of their final mark. The 2017 class had 36 students. Of these, 21 achieved an A grade, with four getting 100%; nine received a B; two received a C; and four failed. The students who failed did not take advantage of the supporting help desks and tutorials, while struggling students who did use these achieved Cs or Bs.

5 TAKING THIS FURTHER

There are several ways to extend this assignment. Examples are: more complex objects to avoid, target following for the boids, and feeding or roosting behaviour.

REFERENCES

Tony Greening. 2000. Students seen flocking in programming assignments. In *Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSE conference on Innovation and Technology in Computer Science Education (ITiCSE '00)*. ACM, New York, NY, USA, 93–96. https://doi.org/10.1145/343048.343091

Wayne Iba. 2008. There's something about AI exercises. In *AAAI Spring Symposium: Using AI to Motivate Greater Participation in Computer Science*. 44–49.

Craig W Reynolds. 1987. Flocks, herds and schools: A distributed behavioral model. In *Computer Graphics (ACM SIGGRAPH)*, Vol. 21. 25–34.