# Boid Game-Playing through Randomised Movement

Lenka Pitonakova
contact@lenkaspace.net
University of Southampton 2012

**Abstract:** *The original boid flocking algorithm is extended by adding randomised movement to the flock members. This approach is a light-weight alternative to other 'follow the leader' techniques implemented in order to create a 'game-playing' behaviour during which a flock changes its movement direction as observed in real birds.*

**Keywords:** flocking, boids, random movement

## 1. Introduction

The boid flocking algorithm, first introduced by C. W. Reynolds (1987), represents a biologically inspired approach to collective motion animation where agent-level behaviour replaces more traditional computer animation techniques that often involve centralised or pre-scripted control. Reynolds argued that traditional computer animation made creation of realistically looking flocks very hard or nearly impossible to implement, whereas an alternative technique based on continuous individual-level calculations would remove obstacles such as difficulty of centralised motion planning.

There is however a trade off to be made when implementing boid flocking, such as the lack of precise group movement control (Reynolds, 1987) or slow algorithm speed when large flocks are used (Bourg & Seeman, 2004, pp. 57; Hartman & Benes, 2006; Silva et al, 2009) Nevertheless, boid-inspired flocks and herds of artificial animals were used in games like Grand Theft Auto and Pikim, as well as movies including The Lord of The Rings or The Lion King (Silva et al., 2009). Martínez et al. (2008) used boids as spatial clues in adventure games and Bajec et al. (2003) showed that boids with imprecise fuzzy logic could look realistically despite the lack of crisp mathematical calculations. More unusual implementations of boid-like collective behaviour include granular sound synthesis (Kim-Boyle, 2007) and a special case of particle swarm optimization (Cui & Shi, 2009).

This paper presents implementation of the original Reynolds' model and extends it by allowing agents to move randomly in order to create more life-like behaviour.

## 2. The Original Model

The original model was implemented in a 650x650 pixel 2D 'torus' arena where the view field and the position of an agent wrapped around the world during motion. The size of each boid was 20x20 pixels. One simulation update loop was executed each 1/60 seconds and each simulation run lasted 600 seconds. Every run started with boids randomly rotated and having random speed between the minimum (0.4 pixels per update loop) and maximum (2.0) allowed. The boid movement was simulated as continuous (Reynolds, 1987; Bourg & Seeman, 2004, pp. 16-19), where the centre of its body was moved by a floating-point value every time step.

The rotation of a boid could change by a maximum of around 32 degrees per second and the speed could change at any rate, given that it remained within the <0.4; 2.0> range. The velocity change was calculated using the following three components as described by Reynolds

- Separation: steer away from the neighbours
- Alignment: try to match velocity with neighbours
- Cohesion: steer towards perceived average position of neighbours

Formal calculations are given in Bajec et al. (2007) and Hartman & Benes (2006). This paper follows Bourg & Seeman's (2004, pp. 53-73) vectorised implementation.

The velocity change is calculated based on the nearest agent neighbours. Reynolds defined this neighbourhood as a sphere with backward occlusion and the impact of a neighbour within the sphere was weighted based on its distance. Other authors used different shapes of the view field, for instance a full sphere (Bajec et al., 2003; Silva et al., 2009) or a very narrow forward-pointing cone (Bourg & Seeman, 2004, pp. 65), without implementing the neighbour distance adjustment. The radius of the neighbourhood ranges in different applications, from 3 (Silva et al, 2009) to 7 agent body lengths (Bajec et al, 2003). The model implemented for this paper uses the original Reynolds' wide view field translated into 2D with radius of 5 body lengths (Figure 1).

Once the individual velocity change requests are calculated, many authors (Bourg & Seeman, 2004, pp. 68-73; Hartman & Benes, 2006; Martínez et al., 2008) simply multiply them with fixed weights and add them together, cutting the resulting force to fit within a velocity change range. Reynolds himself suggest that this produces behaviour that works "pretty well", although a request accumulator is needed when resolution of conflicts is important, for example when two steering requests point to opposite directions and therefore cancel each other as it could be the case during obstacle avoidance. Reynolds therefore implemented an accumulator by

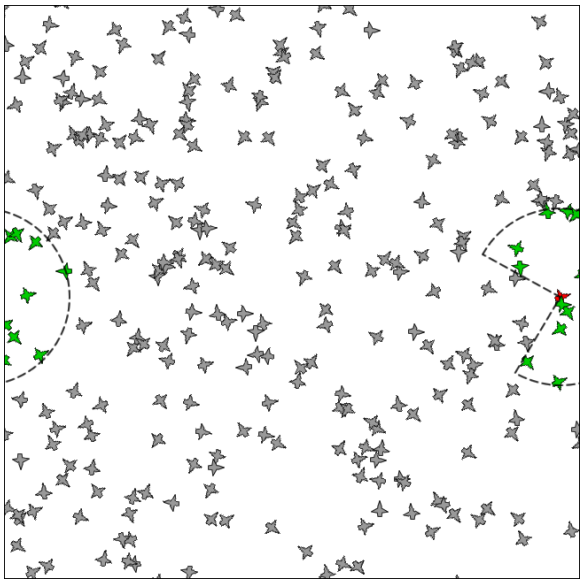

**Figure 1:** Field of vision of a boid in a torus world. The boid is marked in red and its visible neighbours are green.

calculating magnitudes of each request and adding them together until a specific threshold was reached, at which point the last velocity change request was trimmed to compensate for the excess magnitude and the rest of the requests were set to 0.

Figures 2 and 3 show that under conditions with no obstacles and no randomness, using the request accumulator made no difference in terms of how fast boids agreed on collective orientation and speed. While the common speed stabilised fairly quickly, standard deviation of rotation kept logarithmically decreasing for around 240 seconds, depending on the flock size. Furthermore, it took a smaller flock longer to agree on a common orientation, but the differences between individuals were smaller afterwards.

Using of the request accumulator also had no effect on occurrence of boid-to-boid collisions (Figure 4). The presented results are based on 200 runs and the used request weights were 0.2 for cohesion and separation and 0.6 for alignment.

Wave-like adjustments were observed during flock stabilisation, especially in runs with 50 boids when the request accumulator was used. This suggests that while using the accumulator did not affect the algorithm per-

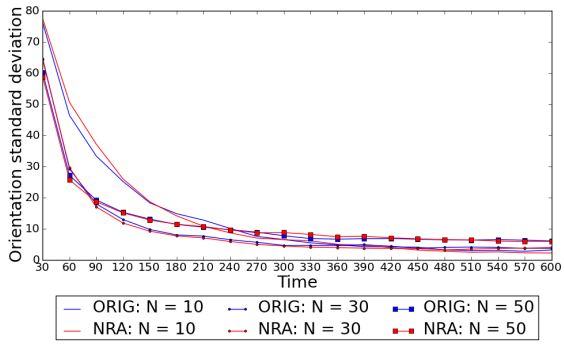


**Figure 2:** Average standard deviation of boid orientation across 200 runs, sampled during 30-second intervals using the Reynolds' original (ORIG) algorithm and the algorithm with no request accumulator (NRA) in flocks of size N.

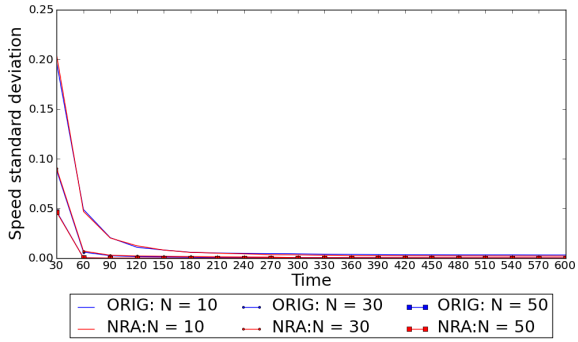


**Figure 3:** Average standard deviation of boid speed across 200 runs, sampled during 30-second intervals. See Figure 2 for the legend clarification.

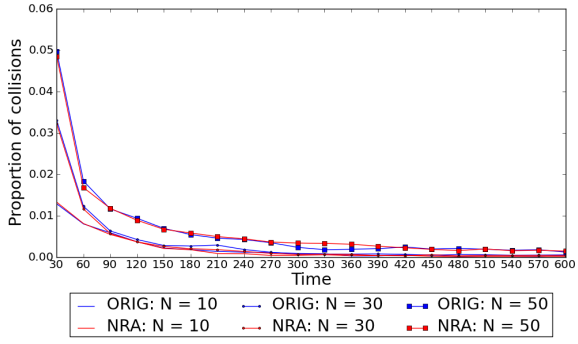


**Figure 4:** Proportion of boids colliding with another boid averaged across 200 runs and sampled during 30-second intervals. See Figure 2 for the legend clarification.

formance, believability of the animation was increased. However, this observation is to some extent a matter of subjective opinion.

## 3. The Game-Playing Model

Velocity of a stabilised flock using the original model did not change, making the animation look rather non-interesting. In contrast, real birds often 'play games', i.e. repeatedly change direction of their collective flight (Hartman & Benes, 2006). The change does not happen simultaneously but it is rather spread through the flock like a shock wave (Reynolds 1987). Schools of fish exhibit similar behaviour (Couzin, 2005).

It has been argued that simply setting a global force in all boids to achieve collective velocity change creates unrealistic motion (Reynolds, 1987; Hartman & Benes, 2006). Instead, a boid can be designated as a "leader" and change the flock direction progressively. However, only boids on the edge of the flock are often allowed to become leaders and a leader often needs a separate function to calculate a velocity that would translate it away from the flock (Bourg & Seeman, 2004, pp. 76-79; Hartman & Benes, 2006).

This paper explores an alternative where a random velocity change vector is added to any boid at any time in order to achieve the game-playing behaviour. In the accumulator implementation, random movement is added as the last request but has a weight of 1.0, i.e. fills in whatever accumulator space is left. Rather than having a 'leader' state, each boid randomly enters a 'random movement' state approximately each 25 seconds and remains in this state for around 5 seconds. A random steering vector is picked when a boid starts the random movement and is always added to the resultant velocity until the boid resumes normal behaviour. This method assures that a) the randomly-moving agent can move the flock in any direction since it does not only try to steer away from the flock b) the movement is smooth as the same randomised vector is added during the randomised motion and c) although any agent can enter the random movement state, other forces like alignment or cohesion become stronger if another flock member is already moving randomly, providing a continuous transition of an individual from normal to random movement and back and removing the need for explicit agreement on who the leader is.

A flock with added random movement could stabilise itself but also change its direction as randomness became more important than the other forces.

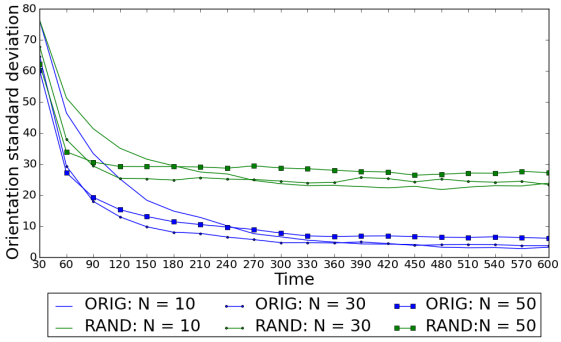


**Figure 5:** Average standard deviation of boid orientation across 200 runs, sampled during 30-second intervals using the Reynolds' original (ORIG) algorithm and randomised movement (RAND) in flocks of size N.

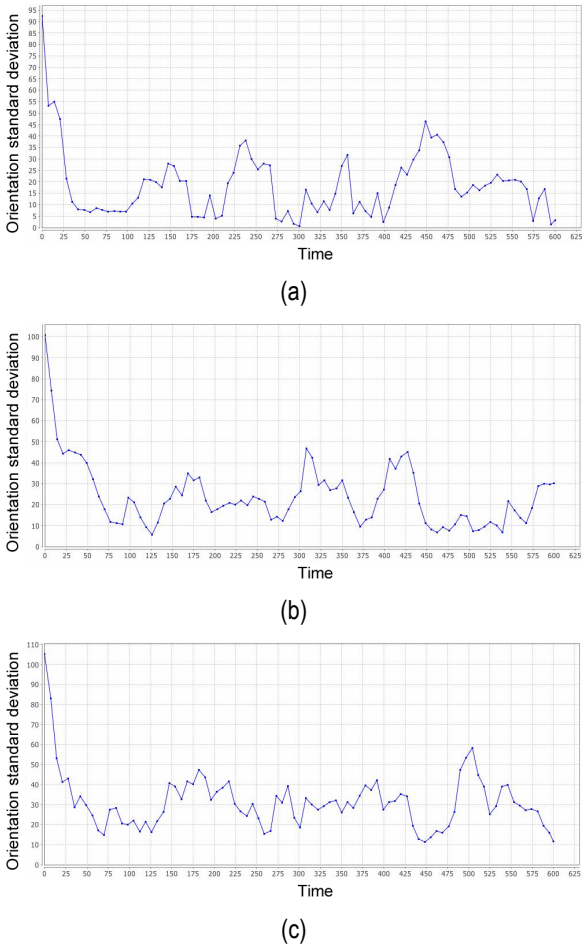


**Figure 6:** Standard deviation of boid orientation in exemplary runs, sampled during 1-second intervals using randomised movement in flocks of size 10 (a), 30 (b) and 50 (c).

Furthermore, a large enough group sometimes split up and re-joined as multiple boids distant from each other entered the randomised state, providing more realistic movement than the original model. Similar flock splitting was observed by other authors who implemented their own 'follow the leader' rules (Bourg & Seeman, 2004, pp. 76-79; Hartman & Benes, 2006).

Figure 5 shows that the average standard deviation of orientation decreased slower than without random movement and that it did not fall below 25 degrees. It is however important to note that this is an effect on averaging over 200 runs. Recordings from individual runs (Figure 6) reveal that the common orientation was periodically agreed on and disturbed regardless of the flock size, although smaller flocks experienced on average lower deviations. A similar effect was observed for the standard deviation of speed (Figure 7).

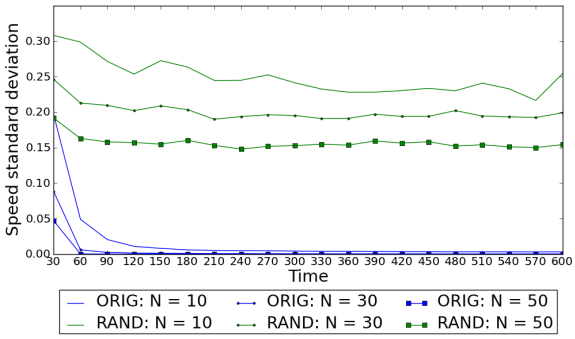


**Figure 7:** Average standard deviation of boid speed across 200 runs, sampled during 30-second intervals. See Figure 5 for the legend clarification.

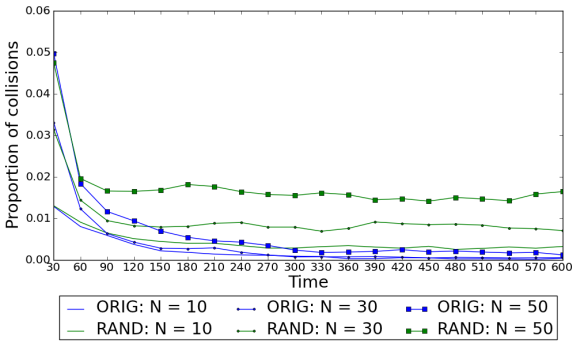


**Figure 8:** Proportion of boids colliding with another boid averaged across 200 runs and sampled during 30-second intervals. See Figure 5 for the legend clarification.

The average amount of collisions also increased when random movement was introduced and this effect was observed more strongly for larger flocks (Figure 8).

## 4. Discussion

An interesting property of the implemented original flocking algorithm is the fact that even though each run started with random boid velocity, the flock always stabilised pointing downwards, i.e. with 180 degree rotation, regardless of the flock size (Figure 9). It was found that this equilibrium point could be manipulated by adjusting weights of the individual velocity change request but that there was always only one attractor orientation value. Further investigation into the vector calculations that governed the flock movement is needed in order find out why this occurred.

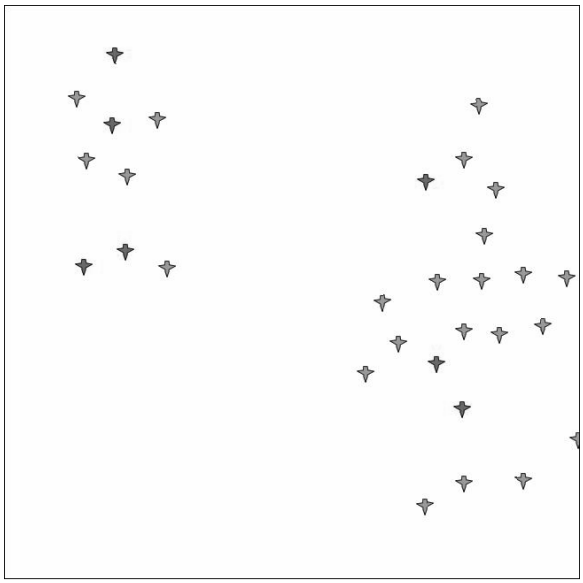

**Figure 9:** Example of finalised orientation in a 30-member flock.

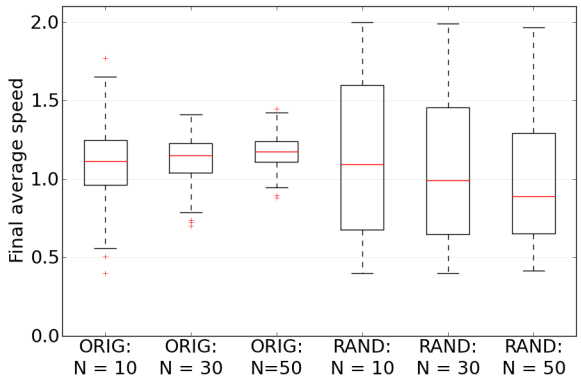


**Figure 10:** Box plots of average boid speed at the end of a run, based on 200 runs. See Figure 5 for legend clarification.

Likewise, the average final speed achieved throughout 200 runs with any flock size was around 1.2, i.e. the middle point between the minimum (0.4) and maximum (2.0) speed. However, unlike the case with orientation, the final speed varied and the variation was greater in smaller flocks (Figure 10).

As it was expected, addition of randomised movement caused the final orientation and speed to vary significantly between the runs. Similarly than with non-randomised motion, smaller variance was observed in larger flocks (Figure 10).

It was shown on Figure 8 that flocks with randomised movement experienced higher collision rates. It was possible for multiple boids that were far from each other to start pulling the flock into different directions, resulting in many conflicts between the sub flocks that formed. This effect was especially evident when observing 50-member flocks. A similar problem was noted when the request accumulator was not used, i.e. when the random movement vector was simply added to the other forces before restricting the final velocity change to the acceptable range (data not shown). This suggests that higher collision occurrence did not result from a conflict between the random velocity change and the other requests, in which case the request accumulator would have presumably delivered better results than simple adding of the forces. It is possible that better balance between the individual requests, smaller random velocity change vectors or a wider field of view would minimise the collisions.

Nevertheless, the addition of randomised movement did provide more realistically looking animation than when the original rules of separation, alignment and cohesion were followed. In contrast with other "follow the leader" implementations, randomised movement does not need calculation of additional vectors and does not make assumptions about the leader's position

relative to its neighbours. The flock or at least a part of it follows a boid who came up with the idea of changing its velocity first.

## Project code and executable

Full Java code for this project can be downloaded from http://lenkaspace.net/downloads/code/boidGamePlaying.zip

A runnable Java applet can be found on http://lenkaspace.net/previews/boidGamePlaying

## References


Bajec, I.L., Mraz, M. & Zimic, N., 2003. Boids with a fuzzy way of thinking. *Proceedings of ASC*, pp.58–62.

Bajec, I.L., Zimic, N. & Mraz, M., 2007. The computational beauty of flocking: boids revisited. *Mathematical and Computer Modelling of Dynamical Systems*, 13(4), pp.331–347.

Bourg, D. M., & Seeman, G., 2004. *AI for game developers* (1st edition). Sebastopol: O' Reilly Media.

Couzin I. D. et al. (2005) 'Effective leadership and decision-making in animal groups on the move'. *Nature,* 433:513-516

Cui, Z. & Shi, Z., 2009. Boid Particle Swarm Optimization. *International Journal of Innovative Computing and Applications*, 2(2), pp.77–85.

Hartman, C. & Benes, B., 2006. Autonomous boids. *Computer Animation and Virtual Worlds*, 17(3-4), pp.199–206.

Kim-Boyle, D., 2006. Spectral and Granular Spatialization with Boids. *Proceedings of the 2006 International Computer Music Conference*, pp.139–142.

Martínez, J. et al., 2008. Animal Flocks as Natural and Dynamic Spatial Clues in Adventure Video-games. *The International Journal of Virtual Reality*, 7(2), pp.73–80.

Reynolds, C.W., 1987. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4), pp.25–34.

Silva, A., Lages, W. & Chaimowicz, L., 2009. Boids that See : Using Self-Occlusion for Simulating Large Groups on GPUs. *ACM Computers in Entertainment*, 7 (December), Article 51.