Simulation of Swarm Behaviour

Abstract:

A flock of birds, a school of fish, a herd of animals or even human crowd at an airport terminal are both beautifully synchronized and intriguing to contemplate. They give a strong impression of an intentional, centralized control. Yet we know that these motions must merely be the aggregate result of the actions of individual members, each acting solely on the basis of its own local perception of the world. A large flock of small birds, for example, is made up of discrete birds, yet the overall motions seems fluid. It seems randomly arrayed and yet is magnificently synchronized. For these reasons, simulating swarm behaviours has been a topic of interest among computer animators, artificial intelligence experts and distributed systems engineers alike for quite some time.

Swarm behaviour, or swarming, is a collective behaviour exhibited by animals of similar size which aggregate together, perhaps milling about the same spot or perhaps moving *en-mass* or migrating in some direction. The term *flocking* is usually used to refer specifically to swarm behaviour in birds. It is an emergent behaviour arising from simple rules that are followed by individuals and does not involve any central coordination.[8]

Project Description:

In this project, we aspire to animate swarm motion by simulating swarm behaviour. We will use the term *swarm* of *boids* to represent a flock of birds or school of fish or herd of animals etc. We will assume a swarm motion as something that is not scripted nor centrally controlled, but *emerging* automatically from simply the aggregate result of interaction between individual bots.

Analogues to Particle Systems:

The simulated swarm behaviour is closely related to particle systems, which are collections of large numbers of individual particles, each having its own behaviour. Each particle has it's own state which may consist of colour, opacity, location, velocity, etc. In particle systems which simulate fluids like fog, smoke, clouds, etc. an individual particle's behaviour depends on the particles in its vicinity.[1]

Although swarm animation has some differences, or upgrades, from a particle system: The particles in the Reeve's original paper consisted of dot-like point particles. Whereas a boid may have its own geometry and its own local co-ordinate system. This might give rise to other complexities like orientation, pitch, yaw, roil of the individual boid which are meaningless in a point-particle. Another difference is that a swarm behaviour is several orders of magnitude complex than that of a typical particle system, a property which has spawned an entire branch in artificial intelligence to employ swarm behaviour to solve complex problems.[9]

Existing Literature:

There has been quite some work in swarm simulation since the 1980s. Craig Reynolds' paper on Flocks, Herds and Schools: A Distributed Behavioural Model[2] is one of the most cited works. It has spawned many implementations ever since. There has been research by zoologists in observing the nature to get the fine details of swarm behaviour in action which has been evolving since the origin of life on earth. There has been research in the AI community in exploiting swarm behaviour to solve NP-complete problems. There has been research in computer graphics community to simulate motion of a group of identical objects, because far more number of objects can be animated this way than its possible for any algorithm with *scripted* motion trajectories. Swarm behaviour has been used in movies to simulate fleet of robots, armies of Orcs, colonies of penguins, herds of bisons, and even crowds at the White House[10].

Tools like Blender have support for particle systems which can be helpful if needed. There are source codes available on the internet in C++, Java, WebGL, threeJS, Unity3D, and MS-XNA.[5] Currently we plan to implement the algorithm in C++ and use OpenGL, GLUT for rendering and window management.

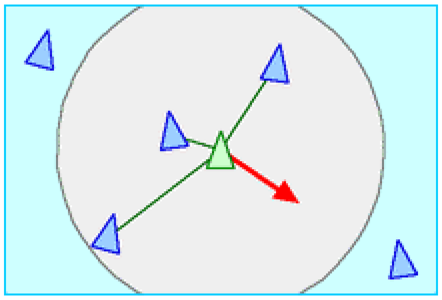
Goals of the Project:

We will focus on creating simple boids with enough mutual interaction capabilities so as to get them in motion in such a way that a swarm-like behaviour will emerge. That would be the first thing on our sequence of activities.

Due to the lack of scripted motion trajectories and very limited centralized control, a swarm of boids is a distributed system. The behavioural model is complicated enough that even for simplest plain boids, we will need to use the appropriate formal computational model and architecture. Our initial idea is as follows:

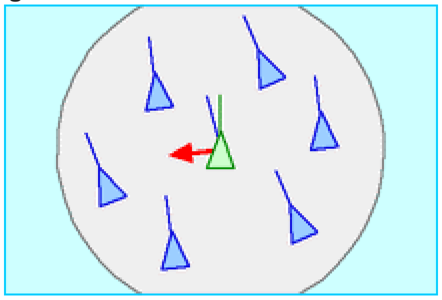
Each boid will have its state, its procedures and its process. A state might comprise of its location in the global coordinate system, its own local coordinate variables, its color, texture, velocity, opacity, age, etc. A boid's procedures may include the the methods or functions that apply rules of swarming behaviour. The three essential rules are separation, alignment and cohesion[4]:

Separation or Collision Avoidance:

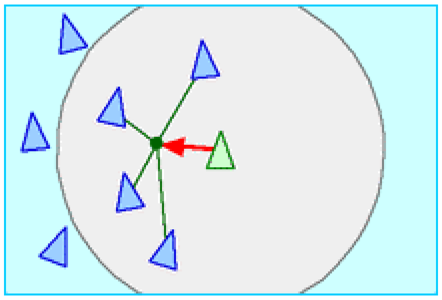


avoiding collisions with nearby boids

Alignment or Velocity Matching:


attempt to match velocity with nearby boids

Cohesion or Flock Centering:


attempt to stay close to nearby boids

The states and the procedures can be best encapsulated into a class, and it's convenient to spawn boids as instances of that class. So an object-oriented model seems like the approach to try. A process is the invocation of procedures of each object to update its states for displaying the next frame. We can run such processes serially on a CPU or in parallel on a GPU.

If time permits, in the next sequence of activities, we will try to implement one or more of the following improvisations:

1. Calculate each object's next states in parallel on the GPU so as to increase the number of boids that a machine can possibly simulate.
2. Add details to the boids. The details may include shape (geometry), texture, local coordinates and rotations with respect to those coordinates (pitch, roil and yaw), etc.
3. Adjust tessellation (number of triangles that form the mesh) of the boids dynamically depending on the depth of the boids. The boids that are far away will need fewer number of triangles in the mesh as compared to the boids that are nearer to the observer.
4. Create the environment in real-time using procedural generation. Perlin noise can be used to avoid any repetitive patterns in the environment.
5. Simulate obstacle avoidance in the boids, so that the swarm will avoid collision with any obstacle in its path while maintaining swarm-like behaviour.
6. Add advanced procedures in the boid to make the swarm-behaviour more natural. These may include one or more of seek, flee, pursuit, evasion, wander, path following, wall following, containment, flow field following, unaligned collision avoidance, flocking, leader following, etc.[2]

References:

1. http://www.red3d.com/cwr/papers/1987/SIGGRAPH87.pdf
2. http://www.red3d.com/cwr/steer/gdc99/
3. http://people.cs.clemson.edu/~dhouse/courses/2009/881/papers/reynolds00.pdf
4. http://www.red3d.com/cwr/boids/
5. http://www.kfish.org/boids/pseudocode.html
6. https://hal.archives-ouvertes.fr/file/index/docid/764996/filename/SPSO_descriptions.pdf
7. http://www.hvass-labs.org/people/magnus/publications/pedersen08simplifying.pdf
8. http://en.wikipedia.org/wiki/Swarm_behaviour
9. http://en.wikipedia.org/wiki/Swarm_intelligence
10. http://youtu.be/-jF5sAqBp4w
11. http://www.hvass-labs.org/people/magnus/publications/pedersen10good-pso.pdf