Massive Battle: Coordinated Movement of Autonomous Agents

Alberto Boccardo, Rosario De Chiara and Vittorio Scarano
ISISLab - Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Università degli Studi di Salerno
{dechiara|vitsca}@dia.unisa.it

Abstract

The simulation of groups of characters moving in a virtual world is a topic that has been investigated since to 1980s. A widespread approach to this problem is the boid model. In the boid model agents, named boids, simulates the flight of a flock of birds. The keystone of the system is the concept of behavior: for each boid in the flock, a simple geometric calculation based on the positions of a limited number of boids, suggests an acceleration along a certain direction. Mixing more behaviors permits to create more complex behaviors like the boids following each others or avoiding obstacles. Starting from this, we present a system, Massive Battle, that, by extending basic behaviors, reproduces the complex movements of platoons of soldiers marching along a path and even simulates them engaging in a battle. The system is designed for online interactive simulations.

Keywords: Simulation, Autonomous Agents, Boid-model;

1 Introduction

The simulation of groups of characters moving in a virtual world is a topic that has been investigated since the 1980s. A widespread approach to this kind of simulations has been introduced in [4] and it takes inspiration from *particles system* [3]. In a particle system there is an emitter that generates a number of particles that move accordingly to a set of physics inspired parameters (e.g. initial velocity, gravity). The particle system approach is expanded with the purpose of simulating a group of more complex entities, dubbed *autonomous agents*, whose movements are related to social interactions among group members. A classical example of use of this approach is the simulation a flock of birds in the most natural possible way. Elements of this simulated flock are usually named *boids* (from *birdoid*) and got instilled a range of *behaviors* that induces some kind of *personality*. Actual implementations of such personalities can vary [14, 6], but the idea is the following: at every frame of the simulation, for every boid, for each behavior in the personality the system calculates a request to accelerate in a certain direction and the actual movement is the result of a weighted sum of all the contributions for a certain boid. The behaviors are, in the most of cases, simply geometric calculations that every boid makes, considering the nearest boids it is flying with: for example the behavior called *pursuit* just let the boid to pursuit a moving target (e.g. another boid). Boids react to their neighbors so they must be able to identify them by filtering nearby boids out of the whole population. The most simple way of do this filtering consists in a $O(n^2)$ proximity screening, and for this reason the efficiency of the implementation is yet to be considered an issue.

The number and the type of the behaviors allow to implement different personalities: a *leader* in the flock is a particular boid who has the knowledge of the path to follow. To differentiate a leader boid among the flock is pretty easy, while every boid in the flock has the behavior of following the leader, the leader has the behavior of following the path, ignoring other boids. In [1] is presented a study where such model is investigated and the simulated flock demonstrates the

capability of being able of following a path (e.g. toward food, along a migration route) even if the number of informed boids, that is leaders, is a very small proportion of the whole group.

In the following discussion we will refer to the *boid model* as the model presented in [4]. The boid model is designed for the aggregate motion of a simulated flock of boids as the result of the interactions of the relatively simple behaviors.

1.1 Previous works

In [8] is presented a technique to let a group of mechanical robots to navigate an environment with obstacles. The technique is based on potential functions: every point in the space has an assigned potential value that measures how much a robot is attracted by it. Robots will move around looking for the highest potential that usually is assigned to the goal to be reached. Obstacles will provide negative potential in order to repulse robots. The methodology is also extended by letting robots to coordinate in order to create a formation: every robots has a series of attachment sites that will attract other robots, in this way robots will just attach each others in various formation by defining suitable attachment sites.

The movement in an obstacles field is approached in [9], where it is presented a solution that plans the movement of a group of units without splitting it. The algorithm performs two phases: on the first phase it calculates the *backbone path* while in the second phase the actual group movement is performed. The backbone path is a function that describe a "corridor" amidst obstacles where the clearance at every point on the path is at least the radius of the enclosing circle of the units. The backbone path is the only area in the map where the units are allowed to walk during the second phase, where units actually walk along the path directed by a potential function. This solution clearly achieve the goal of letting the group to walk in an area avoiding splits.

A different solution to the problem of group splitting while traversing an obstacles filled area is presented in [10]. The paper offers two results: firstly an algorithm to control the steering behavior of groups based on a boundary value problem (BVP) path planner, then a strategy to handle the group formation-keeping problem effective to use any desirable formation shape. An interesting implementation detail of the techniques is that because the BVP path planner is based on the equation of Laplace, it is suitable to be efficiently implemented on GPUs and multi-core CPUs. The idea on which the path planning is based is a *group map* that surrounds the group during its movements on the map. The group map keeps track of the obstacles as repulsive areas within it, while the formation positions are attractive areas.

A commercial application that worths to be cited here is Massive [12], whose name is an acronym for Multiple Agent Simulation System in Virtual Environment. Massive allows to design, simulate and render complex scenes containing up-to millions of agents. The agents simulated personalities can other an high degree of realism. The system is not designed to render scenes in real-time.

This paper describes a system capable of animating autonomous agents with the purpose of reconstructing interactive scenes from a battlefield showing a number platoons fighting each others. Each platoon presents different soldier topology deploying different kinds of weapons. In this scenario we also liked to introduce a "courage" factor that let the user to tweak the ability of the soldier to stick to its purpose of killing the enemy. We mutated the idea of expanding previous models in order to reach an higher degree of complexity of the behaviors. We will use the boid model as the foundation on which to build more complex behaviors, the initial idea of simulating a flock of boids will be expanded to simulate a platoon of soldiers obeying to commands imparted by a leader. Soldiers not only will be able to march along a path but will also be capable of engaging a fight with enemy platoons.

The contribution of our paper are two successive steps of the process of expanding the boid model toward a more complex model. On the first step we show how by using basic behaviors we are able to convey to a group of agents the capability of marching like a platoon, executing directional commands. On the second step we extend the capabilities of executing orders to provide soldiers the capability of engaging with enemiy platoons, if any in sight. In Figure 1 is shown how

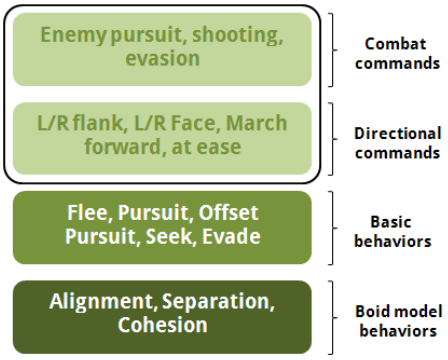

Figure 1: A vision of the presented technique.

the whole system is built: on the lower level there is the boid model, that offers basic behaviors to simulate a flock of boids; on the upper (second) level there are slightly more complex behaviors that are similar to boid model's behaviors, but offers different functionalities; both these level are implemented in [11] which is the library we used in our prototype. The upper levels in Figure 1, surrounded by a rectangle, represents the successive extension of functionalities offered by our system: the first one is the capabilities of marching as a platoon that is obtained by overlapping the effects of two of more basic behaviors; the other one is the combat capability, that is offered by slightly modifying the marching capability.

2 Coordinated movement

The target of animating units belonging to platoons in a realistic way has been inspired by real platoons parade in which soldier execute a certain amount of activities in a coordinated way. Soldiers usually are trained since first days of their carrier to march by coordinating just by looking at a small amount of their neighbors and, no matters of this limited coordination effort, they are able to follow orders fed them by the *commander*. A fundamental role is played by the *leader*, a special unit that in charge of to coordinate troops by the mean of specific directional commands.

In the following we will use the terms "soldier" and "unit" as synonyms. We will use the term "platoon" to mean an amount of soldiers within our system; this is an abuse and does not have any relation to real platoon, also considering the varying number of components in simulated platoons.

The Massive Battle system handles 4 different types of information that made up the scene that the user intends to simulate:

The map: the whole simulation will be carried out on a bidimensional map;

Obstacles: are static objects units have to outflank;

Leader: it is the commander of a platoon, it feeds the unit with directional commands;

Units: units made up a platoon and execute directional commands by just exploiting a local vision of the platoon.

All these different types of information are collected in a script that describes the map, the obstacles placed onto it, the number and the type of the platoon and for each platoon, it contains a description of the unit characteristics.

2.1 Basic behaviors

The whole system architecture is shown in Figure 2: on the upper level there is the script containing a path made by checkpoints, to reach each checkpoint, the leader will impart directional commands

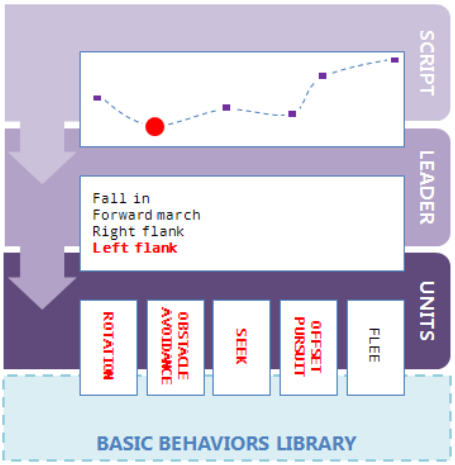

Figure 2: The information exchange between the three level of command.

to the troops, every directional command will be translated in a set of basic behaviors implemented by a library of basic behaviors. The example in Figure 2 shows what happens on the checkpoint shown in boldface, the directional command that will be imparted by the leader and the basic behaviors that will be taken into account by units.

We will shortly summarize a set of basic behaviors the system uses to translate high level directional commands received by the leader.

Seek this behavior lets the unit (agent) to move toward a point; The Flee behavior is the opposite of the Seek behavior: the unit runs away from a certain point. In Figure 3.(a) the effects of the Seek and the Flee behaviors are shown.

Pursuit it is similar to the Seek behavior but the fact that it applies to a moving point (e.g. another agent). This behavior takes into account a prediction of the movement of the point to follow. The Evade lets an agent to run away from a moving point. In Figure 3.(b) the Evasion and Pursuit behaviors are compared. A slightly different behavior is the Offset pursuit behavior that lets a unit to pursuit another unit keeping a certain distance (offset) between them.

Obstacle Avoidance this is a fundamental behavior in which the obstacles present in the scene are took into account. In Figure 3.(c) is shown how among 3 obstacles (A, B, C) just the nearest one (B) is taken into account (red arrow).

Flocking this behavior is the one directly inspired by looking at real bird flying in formation. Even if it can be considered as a basic behavior, in the boids-model it is the result of three behaviors that are applied simultaneously Separation, Cohesion and Alignment. Separation lets a boid to stay away from its neighborhood and Cohesion is the opposite of this behavior. Alignment lets the boid to align its fly direction to the average of neighborhood boids directions.

2.2 The leader

The leader mimics the role of the commander of a real platoon of soldiers. The commander usually marches besides the platoon and feeds the platoon with commands. Following a list of the implemented commands [2]:

Fall in: soldiers get together and form a platoon;

Forward march: soldiers start marching in the direction they are facing;

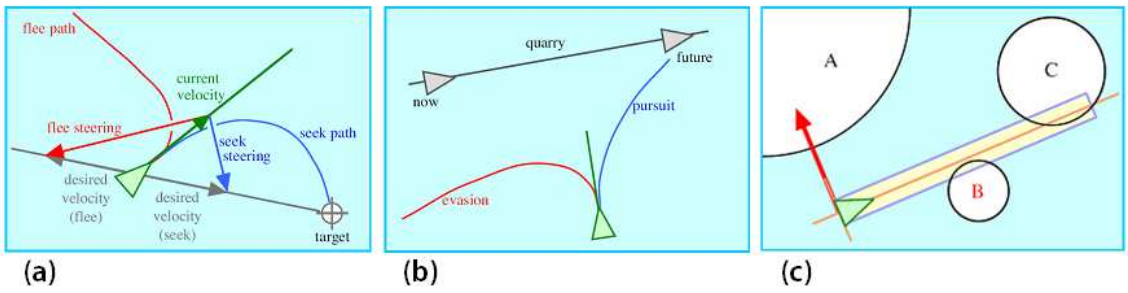

Figure 3: Different basic behaviors: (a) Flee and Seek; (B) Evasion and Pursuit; (C) Obstascle avoidance (Figure from [5]).

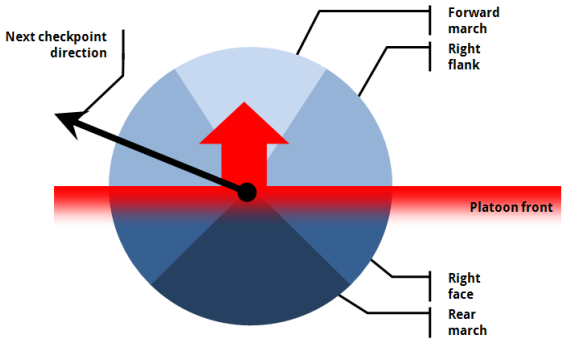

Figure 4: How the leader chooses the correct command to conduct the platoon to the next checkpoint (rightward commands shown). In this example the leader will choose to impart a Left Flank command to the platoon.

Rear march: the platoon performs a 180 degrees turn, while marching;

Right (Left) flank: the platoon performs a 90 degree pivot to the right (left), while marching.

Halt: soldiers stop the marching immediately;

Right (Left) face: a command given from a halt to turn 90 degrees to the right (left);

To march at ease: soldiers can march avoiding keeping in step because of the rough terrain.

The simulation consists of a certain number of platoons together with their paths to be followed. Paths, made by checkpoints, are followed automatically by every platoon. The platoon commander follows every checkpoint it finds along the path, one by one, feeding suitable commands to the troops. At every checkpoint the suitable command is simply chosen among the directional commands in the list above, by using the angle between the current direction and the direction of the next checkpoint. In Figure 4 is shown how this selection is implemented.

2.3 Assembling behaviors

To perform the directional commands described in previous section different units in the platoon must play different roles: this is exactly what happens in reality.

Fall in command. The fall in command is sent to every unit in the platoon. Every unit has an associated position within the platoon and reach this position by assuming an offset pursuit behavior. Once the unit reached its final position it notifies the leader, and whenever the fall in is completed, the commander will issue the next command.

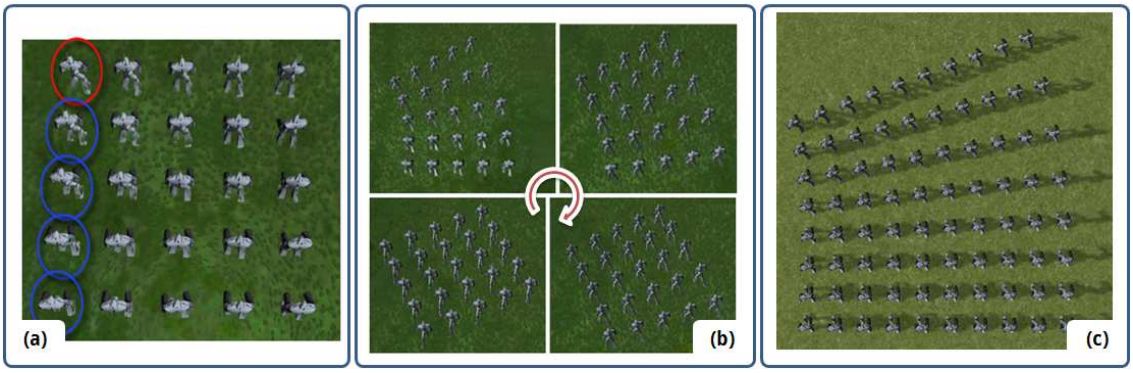

Figure 5: The left flanking (a) different roles played by soldiers: red circle the platoon pivot, blue circles the line pivot; (b) the result of a left flanking (from upper left, clockwise); (c) the same command execute by a larger platoon.

Left flank command. In Figure 5.(a) the three different roles played by soldiers to perform a left flank: the leftmost soldier on the front line is the *platoon pivot*, the leftmost soldiers on each line are the *line pivots* and the rest of non-pivot soldiers keep on marching by coordinating each others during the flanking. The differences between the roles are implemented by using a number of basic behaviors. The formation pivot performs a simple rotation by marching in place; line pivots will keep the rotation together with a seek behavior on the formation pivot; the other soldiers will perform a simple offset pursuit on the line pivot of the the line they belongs to. In Figure 5.(b) is shown, starting from the upper left figure and going clockwise, the effect of the left flanking of a 25 units platoon.

Marching at ease. When an obstacle gets in sight of a unit it communicates this information to the leader that will impart the command of Marching at ease, this will let soldiers to keep moving toward the checkpoint without marching in formation. This high level behavior is obtained by assembling the effects of three basic behaviors: obstacle avoidance to circumvent obstacles, seek toward the checkpoint and flocking to not to get too far from others.

2.4 Combat

Once the platoon is able to walk along a path executing the directional commands imparted by the leader, is reasonable to model the situation in which two or more adversary platoons come insight. We have took into account this situation by firstly offer three different models of weapons: melee weapon model, mortar weapon model and rifle weapon model. Each platoon within the simulation has an identification that is used to discriminate among friend and foe platoons. While the leader is conducting the platoon along the checkpoints it always queries the system in order to verify if an enemy is in sight. Once an enemy is visible by the leader, the leader switches to *combat mode* and impart the same command to the soldiers.

Before describing the details of the combat mode we shortly summarize the differences running between the three weapons model available in the system. A *melee weapon* is a model that is used just in the clinch fighting, so to be effectively used, soldiers have to reach the enemy at a very short range. A *mortar weapon* simulates the behavior of a weapon shooting bullets with a high-arcing ballistic trajectories. The *rifle weapon model* produces bullets that runs horizontally to ground, 6.(a).

For each weapon model the system offers a wide range of parameters describing different aspects of how the weapon is actually simulated. The system not only simulates the trajectory of mortar-like shooting, but also allows the user to define the typology of bullets shot, by choosing the radius of the explosion induced by the impact of the bullet with the ground (see Figure6.(b)). The configuration of the weapons is pretty flexible, for example, by reducing the impact radius

of the mortar bullet is possible to simulate an arrow shot by a bow. For the rifle weapon model is possible to define the perforation capability, that is, the capability of trespassing through a number of bodies. In Figure 6.(c) is shown the effect a perforation value set to 2. A common trait of the three different models is the range of action the enemy must be within, in order to let the soldier to effectively use it.

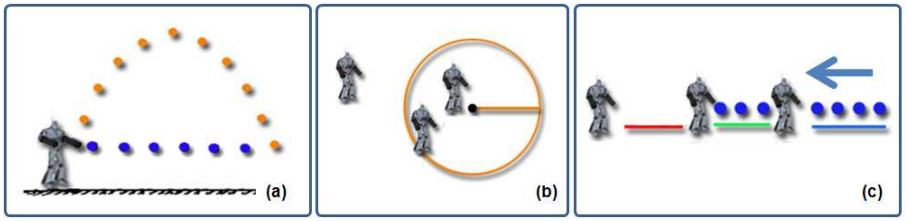

Figure 6: Simulated shooting characteristics (a) comparison between the mortar shooting model and rifle shooting model; (b) the explosion radius of the mortar bullet; (c) the perforation capability of rifle bullets.

When the leader is in combat mode it will keep track of the enemy platoon position and will feed it to the soldiers. Once the soldiers receive the command of switching to combat mode they will stop marching in formation and start marching at ease toward the checkpoint assigned by the leader, that is the position of the enemy. Each soldier will choose an enemy soldier it will keep walking toward until the enemy will gets in the weapon range. Once the weapon can be used to effectively hit the enemy, soldier will try shooting, and will keep doing so until the enemy dies, it will run away from the shooting range or will run out of bullets.

The described combat mode is implemented by using three additional behaviors depending on a factor called *courage*: the courage is used to decide to whether attack a soldier or not. Let consider 2 soldiers involved in a potential fight and their the courage factor, three cases can occur: if the courage of the attacking soldier is low it runs away from the enemy by assume an Evade behavior. If the enemy choose to evade, the soldier will pursuit it, by assuming a Pursuit behavior and shooting whenever the enemy is in range. If none of the soldiers evade, they will engage fighting, by assuming the Seek behavior.

The combat fighting is just a proof-of-concept of the idea of extending the marching behaviors by offering slightly different functionalities.

3 Conclusion

From a technological point of view we developed Massive Battle as a framework, written in C++, that is capable of interpreting a script file. The script file contains a full description of the initial setting of the parameters for each platoon. Once a file is loaded the simulation starts and the user can just sit and enjoy what happens on the screen. The system is implemented in a library that does not depend on the graphics engine used to represent the units. This widen up the scenarios of use: for instance, even if the library is designed for real time operations, it could be used to generate realistic movements of troops for high quality off-line rendering program.

Massive Battle is tested within an environment that simulates a battle among a number of soldiers grouped in platoons in a 3D scenario. Our system has been designed as a rapid prototype system to reconstruct famous historical battles. The basic simulation functionalities can be used to investigate different options, for example once the simulation is reproducing a real battle (e.g. the La Haye Sainte battle), the system allows the user to verify how the history could be different by just varying some of the settings (e.g. the heavy cavalry brigades attack earlier). We have also implemented a *Battle Editor* that provides all the necessary editing tools to modify the the morphology of the terrain and obstacles (e.g. rivers, mountains, buildings etc. . .). Together with the editing tools, the system also offers a set of tools to enhance the fruition of the simulation: it is possible to choose the speed of the simulation, to change the camera position and to stick the camera on a certain soldier.

Figure 7: Two screenshots from the application: (left) 6 platoons confrontating; (right) follow-up camera.

together with some validation performed by historical researchers.

The graphical engine we used to implement the visual part of the system is Ogre3D [13]. The basic behaviors library we used is presented in [11], but similar behaviors can be found also in the OpenSteer library [7]. It is worth mentioning some details about the performances of the system on a off-the-shelf PC: AMD Athlon 64 x2 4200+, 2GB of RAM, ATI X1900 with 512MB. The system was able to animate 3000 units on an interactive framerate of 25 fps (frames-per-second). This performances suggest an avenue for future researches, that is, implementing the behaviors on parallel architecture like GPUs or Multicore CPUs, in particular the spatial database used to query about positions of the agents.

References

[1] Couzin ID, Krause J, Franks NR, Levin SA. Nature. 2005 Feb 3;433(7025):513-6. Effective leadership and decision-making in animal groups on the move.

[2] USA Marine Corps Drill and Ceremonies Manual MCO P5060.20

[3] Reeves, W., T., "Particle Systems-A Technique for Modeling a Class of Fuzzy Objects", ACM Transactions on Graphics, V2–2, April 1983. and reprinted in Computer Graphics. V17–3, July 1983, (acm SIGGRAPH '83 Proceedings), pp. 359-376.

[4] Reynolds C. Flocks, herds and schools: a distributed behavioral model. In SIGGRAPH'87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, ACM, New York, NY, USA, 1987.

[5] Reynolds C. Steering behaviors for autonomous characters. In Game Developers Conference, Miller Freeman Game Group, San Francisco, CA, USA, 1999.

[6] Reynolds C. Big fast crowds on PS3. In Sandbox'06: Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames, ACM, New York, NY, USA, 2006.

[7] http://opensteer.sourceforge.net/

[8] Balch T, Hybinette M. Social potentials for scalable multirobot formations. In IEEE International Conference on Robotics and Automation (ICRA 2000), San Francisco, 2000.

[9] Kamphuis A., Overmars M. H. Motion planning for coherent groups of entities. In IEEE Int. Conf. on Robotics and Automation. IEEE Press, San Diego, CA, 2004.

[10] Silveira, R., Prestes, E., and Nedel, L. P. 2008. Managing coherent groups. Comput. Animat. Virtual Worlds 19, 3-4 (Sep. 2008), 295-305.

[11] Buckland M. Programming Game AI by Example. Wordware Publishing, 2005.

[12] Massive software `http://www.massivesoftware.com`. Accessed on May 2009.

[13] Pro OGRE 3D Programming, (Gregory Junker).

[14] R. De Chiara, U. Erra, M. Tatafiore and V. Scarano. Massive simulation using GPU of a distributed behavioral model of a flock with obstacle avoidance. Proceedings of Vision, Modeling, and Visualization 2004 (VMV 2004) (Stanford - California, USA, Nov 16 - 18, 2004). pp. 233-240.