Crowd Simulation

based on emergent behaviours

Carlos Pérez López

August, 2013

Master of Science,

Computer Animation and Visual Effects

Contents

Abstract

Abstract is the last thing you write, it should be concise. Spend time to write it correctly, it has to be perfect. Google how to write an abstract, it is not just summary!

Acknowledgements

To my parents, who paid the bills.

Chapter 1

Introduction

Currently, the presence of crowds in visual pieces, let they be films or video games, has acquired a lot of relevance. Scenes with a crowded train station, big streets full of pedestrians, tons of animals in a flock, troops of robots, hordes of zombies or armies of warriors can be found very often in modern films and games.

In the past, not many films could afford to employ this kind of resources in their scenes, since the only way of having more characters was actually including more characters in the cast. The use of extras highly increases the cost of a production, besides the requirement of a strict coordination and proper training in some situations. One of the reasons of the boom of big masses in films was the arrival of new technologies in computer graphics and artificial intelligence. Alongside with this, the generation and simulation of virtual crowds became a reality, saving money and time and making the use of crowds affordable to big and small studios.

Other of the reasons of the popularity of big groups of individuals is the strength they give to a shot. It is not only the matter of fact that crowds are visually stunning and their magnificence captures the attention of the audience, but also they are powerful tools to tell and enrich a story. At this point of time, it could be hard if not impossible imagine the saga of the Lord of the Rings without those epic battles of tens of thousands of warriors, or imagine the breathtaking apocalyptic scenes without those milliards of zombies wandering around.

Closely tied to the idea of crowd simulation, the concept of behaviour appears. This is what gives live to the crowd and make the spectator perceive the group nature of it. No matter the shape of the individuals, an army of brave warriors in a battlefield will find enemies and will show aggressive movements; on the other hand, on a ballroom gentlemen and ladies will dance graciously.

(a) *Crowd in a train station*

(b) *Troops of droids (Star Wars)*

(c) *Horde of zombies (World War Z)*

(d) *A Battlefield (Lord of the Rings))*

(e) *A Ballroom*

(f) *One vs Many (Matrix)*

Figure 1.1: *Crowd scenes*

The aim of this project is to propose an approach which gives the flexibility to simulate any sort of crowd needed for a scene. Taking base on how the real world works, this method is based on the principle than the group behaviour is determined by the specific behaviours of every individual. And here is where one of the main

ideas for this project appears: Emergent Behaviour.

Leonard Kleinrock, professor of computer science in UCLA, states that emergent behaviour is unanticipated behaviour shown by a system (Kleinrock, 2011). Once a system is designed and defined by certain rules or mathematical equations, it may configure itself in a way that could not be anticipated. The interaction of a large number of simple individual things is very hard to predict; the complexity does not reside in the individuals, but on the way they are interconnected and they interact to each other. Professor Kleinrock proposes this example: we might know how a bunch of children behave when they are alone, but once you put them together in a group, you will observe behaviours that will surprise you.

Subsequently, how a real crowd behaves is something hard to predict, and how realistic it is depends directly on how realistic each individual is.

This thesis is structured as follows:

- **Chapter 2: Related Work.** It explains the previous approaches in this field, the similarities and differences to the proposed method, as well as the advantages and drawbacks they present.
- **Chapter 3: Technical Background.** Mathematical and physical concepts, algorithms and optimized routines, and object orientation features for the build of the crowd engine or how scripting may enhance the flexibility and scalability of the system.
- **Chapter 4: Agent Based Model.** This is where the current approach is explained into details. The main aims of the method are discussed, how all of this was faced, designed, implemented and tested, and a pipeline where this methodology might fit in a real production situation is presented.
- **Chapter 5: Crowd Engine.**
- **Chapter 6: Applications and results.** Some results of different tests will be shown in this chapter. A set of individual behaviours will be presented and the emergent behaviour observed will be explained and discussed.

- **Chapter 7: Application design and implementation.**
- **Chapter 8: Conclusion.** A final concluding chapter will summarize the whole approach, mentioning the main advantages and drawbacks, as well as presenting possible lines for future work.

Chapter 2

Related work

Generating crowds is a problem that has frequently been faced in the field of computer graphics and artificial intelligence. Numerous solutions have been proposed, following different approaches and applying different ideas and concepts. Crowd motions can be created by planning or simulation, and even some researches propose hybrid approaches.

2.1 Motion Planning for Crowd

Opposite to the philosophy of this thesis, a substantial sum of research establishes that a crowd is not only a group of individuals and involves problems that should be handled at the group level, making use of pre-planned techniques. Frequently, strategies such as virtual force fields, navigation fields, motion planning, navigation graphs, etc. are employed to drive the movement of the whole mass, forgetting by all means the individual nature of the crowd. S. Musse et al. state that a motion planning for a group walking together requires more information than an individual motion planning (Musse & Thalmann, 2001). In addition, this research claims for the need of model behaviours at the level of groups and crowds in order to acquire the beauty of synchronization, homogeneity and unity.

Figure 2.1: *Motion Planning for Crowd using Navigation Fields (Patil et al., 2011)*

Extending the control vertically, in (Musse et al., 2005) a hierarchical model for real time simulation of virtual human crowds is proposed allowing different control features at levels of crowd, groups or individuals. This is an example of hybrid approach where it is possible to increase the complexity of crowd-group-individual behaviours according to the problem to be simulated.

By means of these techniques, it was visually proven that very convincing and realistic results can be produced. Nevertheless, the identity and the decision capacity of each of the individuals is partially if not completely lost.

2.2 Crowd Motion Simulation

Purely simulation strategies, which is the case of this thesis, discard any pre-planned decision and the final result is entirely based on the global consequences of local interactions of members of the population. This is known as Agent-Based Model or Individual-Based Model. This will be detailed in Chapter **??** and for further information check (Reynolds, 1999).

The main idea this thesis is settled on is the simple principle stated by C. Reynolds, the pioneer of flocking behaviours, which claimed that very simple rules can arise emergent behaviours without involving any central coordination. Notice again the concept of emerge.

One of the earliest works in group behaviours was Craig Reynolds' flocking algorithm which was a distributed behaviour model for flocks, herds and schools (Reynolds, 1987). The individuals of his flocking system are called "Boids" and are subjected to three simple rules: cohesion, alignment and separation.

This classic research not only presents a flocking algorithm, but also marks a turning point proposing an individual virtual force model as the way to affect how each agent move. The approach presented in this thesis has adopted that model.

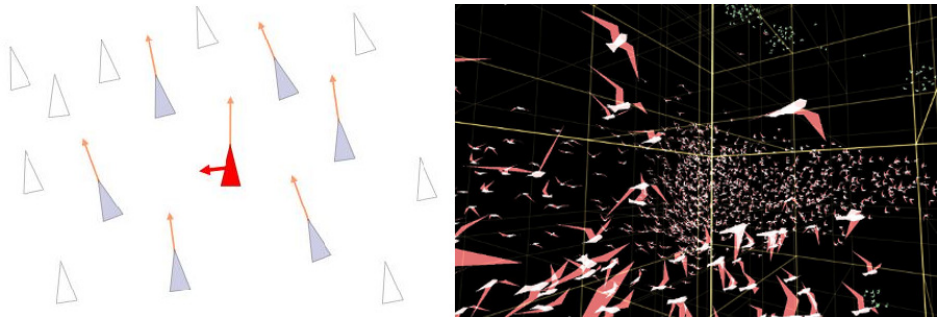


Figure 2.2: *Flocking System following the Virtual Force Model proposed by C. W. Reynolds (Reynolds, 1987)*

Starting from this robust and solid idea, tons of research paths can be taken in order to acquire and compare different approaches for crowd simulation. For instance, C. Wang & T. Li suggest an evolving crowd motion simulation (Wang & Li, 2006). In that research, the use of genetic algorithms is proposed to generate optimal virtual forces according to the given environment and desired movement behaviour.

Although Craig Reynolds presents a base-approach which models very accurately the way crowds behave in the real world, the main disadvantage is that configuring the forces in order to generate desired motion behaviours remains empirical. And again, we are witnesses of a characteristic inherent to emergent behaviours.

2.3 MASSIVE Software

MASSIVE stands for Multiple Agent Simulation System in Virtual Environment and is a software developed by Stephen Regelous in Weta Digital, as a request from Peter Jackson to recreate those epic battle scenes that Tolkien described in the books of the Lord of the Rings. Massive has contributed to the creation of many awarding visual effects, particularly the sequences; and due to this, it has been developed into a complete product and has been licensed by many other visual effects houses.

Figure 2.3: *Award winning battle scene from 'The Lord of the Rings: Return of the King'*

Massive introduces a very interesting model approach which consists in treating each agent as a combination of a body and a brain. The body defines the physical characteristic of the agent and the brain is a fuzzy logic network which controls the actions of the agent such as following an arbitrary terrain, avoiding obstacles or interacting with other agents. This thesis has adopted this natural design combined with the flexibility that scripting languages provide.

Each action is associated to a pre-recorded animation, rather obtained from motion captured session or hand-animated, and will be blended between them in order to achieve the movement of the character. Apart from Artificial Intelligence features, it includes other abilities such as Rigid Body Dynamics (RBD), cloth simulation or GPU rendering.

Chapter 3

Technical Background

The background needed for doing research on engineering or any technical discipline is extremely important, and frequently is what marks the difference between a solid, consistent and robust study from a weak one. Many mathematical and physical concepts (specially physical) are critical to understand analytically the ideas, reasoning and logical models presented in this survey.

This chapter intends to describe formally the main ground concepts where this thesis is settled on. Although this may only be a brief glimpse of all the ideas applied directly or indirectly, and a much finer conceptual background might be developed, this should be enough for following the approach.

The bibliography resource consulted to write this section was the chapter A Maths and Physics Primer, in the book Programming Game AI by Example (Buckland, 2005).

3.1 Physics

In any research that involves modelling the real world, physical rules will acquire a fundamental role, particularly the ones concerned with motion. Next, the key concepts to understand the virtual force model will be presented.

- **Time.** This is a concept that everybody has in mind. Physically speaking, time is a dimension in which events can be ordered from the past through the present into the future. It is a continuous scalar quantity with no direction measured in seconds. Time in computer simulations and computer games might be measured in seconds, like in the real world, or in *virtual seconds* or *ticks*. This will be discussed more in depth in Chapter 7: Application Design and Implementation.

- **Mass.** It is a scalar quantity measured in grams, and it is the measure of an amount of something. This property is directly linked to how fast bodies change of state. For example, if we imagine two people with the same properties except mass, the one with higher mass will require more time to change from standing to running.

- **Strength.** It is the scalar quantity which defines the physical power a person or animal has. Notice that this property is inherent to an individual and does not have direction, if it had, we would be talking about *force*, concept that will be introduced later. And again, this is related with how fast bodies change of state. The bigger strength, the faster change.

- **Position.** These are the location coordinates of a specific point related to an origin. This is not as simple as it might seem, because bodies have a volume, so which the exact position is, is a controversial discussion. Normally it is used the centre of mass, but other points can be used depending on the approach. In order to calculate the movement, we need to know the rate of change of the position, both the magnitude and the direction.

- **Velocity.** This is the vector which defines the rate of change of distance over time. Its standard unit of measurement is m/s. Mathematically it can be expressed as follows:

$$v = \frac{\Delta x}{\Delta t} \tag{3.1}$$

- **Acceleration.** It is a vector which defines the rate of change of velocity over time. Acceleration is written as m/s^2, and expressed by:

$$a = \frac{\Delta v}{\Delta t} \tag{3.2}$$

- **Force.** This is the main element of the physical model followed by this thesis. According to Isaac Newton: "An impressed force is an action exerted upon a body in order to change its state, either of rest, or of uniform motion in a right line". Therefore, a force is a quality that can alter an object's speed or line of motion. It is measured in Newtons and represented as a vector, with both magnitude and direction

We know that to change the position of an object (to move it), it is needed to make its velocity greater than 0, and in order to achieve that, there has to exist an acceleration. Basically, what the object is experimenting is a change of state due to an acceleration produced by forces.

Newton's second law states the relatioship between an object's mass m, its acceleration a, and the applied force F by the equation:

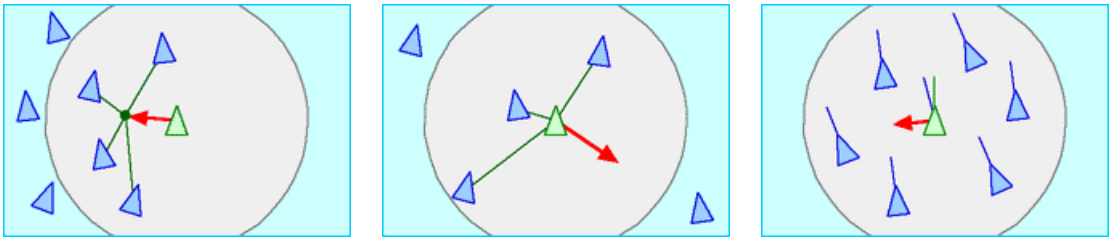
$$F = ma \tag{3.3}$$

Therefore, this gives us the key to perform a physically based simulation. According to all mentioned before, the motion an object experiments in a physically based virtual world can be calculated after synthesizing all the forces applied over it:

1. $a = \frac{F_{total}}{m}$
2. $v_{t+1} = v_t + a$
3. $p_{t+1} = p_t + v_{t+1}$

3.2 Reynold's Flocking Algorithm

Craig W. Reynolds proposed in 1987 a model to simulate natural group behaviours such as herds, flocks or schools (Reynolds, 1987). This is a very powerful mechanism to use in crowds, besides sharing the principles this thesis is based on. The method works by applying three simple rules to each individual of the flock, which Reynolds called *boids*, that make them move as a unit.



(a) *Cohesion Rule* **(b)** *Separation Rule* **(c)** *Alignment Rule*

Figure 3.1: *Reynolds' Model Rules*

Each boid knows a set of neighbours in the flock which influence its movement. Only the ones that are within a certain distance are considered neighbours, and the rest are ignored; thus, boids has local conscience of the flock. According to the virtual force model, those simple rules produce these three steering forces:

- **Cohesion.** This force makes the boid move towards the centre of mass of the neighbourhood so that they remain close to each other.

  Let p be the position and v the velocity of the current boid, p_i the position of the neighbour i and n the number of neighbours:

$$centreOfMass = \frac{\sum_{i=1}^{n} p_i}{n} \tag{3.4}$$

$$cohesionForce = normalize(centreOfMass - p)maxSpeed - v \tag{3.5}$$

- **Separation.** This force makes the boid move away from its neighbours, to avoid remaining too close.

  Let p be the position of the current boid, p_i the position of the neighbour i, d_i the distance to the neighbour i and n the number of neighbours:

$$separationForce = \sum_{i=1}^{n} \frac{normalize(p - p_i)}{d_i} \tag{3.6}$$

- **Alignment.** This attemps to keep the boid aligned with their neighbours.

  Let p be the position and v the velocity of the current boid, p_i the position and v_i the velocity of the neighbour i and n the number of neighbours:

$$averageHeading = \frac{\sum_{i=1}^{n} normalize(v_i)}{n} \tag{3.7}$$

$$alignmentForce = averageHeading - normalize(v) \tag{3.8}$$

3.3 Finite State Machine (FSM)

Finite State Machines, or FSM, have been the main instrument of choice to imbue an agent the illusion on intelligence. Some of the reasons are these:

- Quick and simple to code
- Easy to debug

- Little computational overhead
- They are intuitive
- They are flexible

Historically, a FSM is a rigidly formalized device used by mathematicians to solve problems, whose precursor might be considered the Turing Machine. The idea is to decompose an object's behaviour into easily manageable "chunks" or states. For instance, a light switch is a very simple FSM where *off* and *on* are the states. Transitions are made by the input of the fingers. By clicking the switch up it triggers the transition from off to on, and by clicking the switch down it triggers the transition from on to off. There is no action associated with the off state, but when it is on, the electricity is allowed to flow and light up the room.

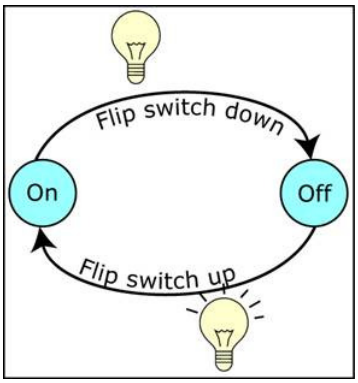


Figure 3.2: *Light Finite State Machine*

As mentioned above, one of the main advantages of FSM is that they are very intuitive. It is human nature to think about things as being in one state or another. Humans do not really work like FSM but sometimes it is useful to think our behaviour in this way. It is fairly easy to break down an agent's behaviour into a number of states with associated actions and to create rules to transit among them.

One feature that notably enhance the power of FSM and allow intercommunication among agents is the message passing support. Intelligent agents can send and receive information to each other, and act upon it. This is a model which reflects quite accurately how real interactions work. This messages are sent in the form of

packets of data to other agents, which might influence their behaviour triggering a transition to a different state. If an archer sends an “arrow” message to an enemy, this might respond changing his state from alive to dead.

Chapter 4

Agent-Based Model

After an introductory chapter and a background overview, this is where this document starts dealing with the particular approach studied in this thesis. The solution will be presented following a bottom-up explanation line, since the author believes that it will be clearer for the reader and is the way behaviours emerge. Thus, in this chapter the abstract model for an individual which belongs to a crowd, and which lives in the virtual world, is explained into details. This individual, which will be referred as *agent* from now on, is the element that will encapsulate the intelligence of the system. Therefore, this approach is based on individual intelligent agents rather than any global mechanism, from whose interactions a group behaviour will arise. As stated before, the crowd behaviour will have as much quality as the agent behaviours have.

Next Chapter will take a wider vision presenting the concept of crowd and the world which holds the complete simulation.

"An agent autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future" Buckland (2005)

Since the old Greece, some philosophers argued about a spiritual division between soul and body, other scientists consider the ability of thinking something that escapes

the bounds of the physical body. Without going such further, this thesis presents an abstract agent model divided in two parts: a body and a brain.

4.1 Agent Body

This part is the one which is materially in the world, and there is no intelligence in here. It is completely physically based and most of its attributes are immutable during a simulation. The body represents how big an agent is, or how fast it is, etc. This way, an agent might be big and slow or small and fast. These attributes might even limit the capacity of decision of the thinking partition of the agent. Let us imagine a situation where a prey's brain desperately keeps requesting for increasing the velocity to escape from a predator, but the body cannot perform a higher speed. Next, the list of the physical properties which define the body of an agent are presented.

4.1.1 Physical Properties

- **Mass.** The mass of an agent determines how much acceleration a force produces on it. Taking the assumption that agents have comparable densities, it is also employed to visualize their size
- **Strength.** This will influence in the *prime mover force*, which is the force an agent applies on itself to move in the space. Therefore, having agents with the same mass, the strong ones will have the ability of opposing to external forces easily or to produce more self-acceleration, changing their velocity faster. The strength might vary if, for example, the agent gets tired or receives any sort of damage.
- **Maximum Strength.** This is the amount of strength that an agent has when it is fully recovered.

- **Velocity.** This is the current velocity that an agent possesses and, contrary to the strength, this attribute has both magnitude and direction. This is the rate of change of the agent's location; thus, it is used to obtain the next position. It is calculated with the acceleration that the total force produce on the agent.
- **Maximum Speed.** This is the higher punctual speed that an agent can move in the world with. Do not confuse this with how fast an agent can change of physical state, which is related with forces and mass (acceleration). Consider the next example: a cheetah can change from 0 to 100 km/h in a matter of seconds (high acceleration); on the other hand, a high-speed train, which needs a lot of force (strength with direction) to move such a big mass, has much smaller acceleration, but it can travel at 200 km/h (high maximum speed).
- **Vision Radius.** This property determines which portion of the world the agent is aware of. This is mainly used to calculate the other agents that one agent can perceive, which will conform its neighbourhood. Therefore, an explorer may have a large vision radius, meanwhile a blind agent will have vision radius 0, and will need to receive information about the world by other means, such as somebody whispering at its ear (message passing).

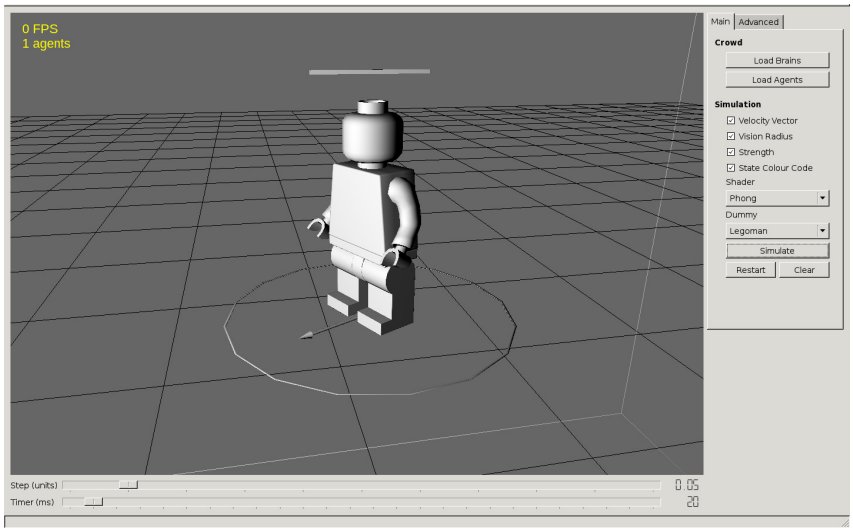


Figure 4.1: *One single agent with its physical properties*

At this point, a comment regarding to the development of this method might be pertinent. How an agent's body behaves in the virtual world is uniquely ruled by physical laws. The brain might say "move up" and apply a force towards the sky but if this force is smaller than the gravity, then the agent will descend. Hence, the piece of this system relative to the physical simulation or the physical properties of an agent is something static, a fixed and solid structure which works. That is why it is implemented in a compiled language. Nevertheless, somebody's way of thinking or acting, might be so diverse that it needs a flexibility that a compiled language cannot provide.[1]

4.2 Agent Brain

This is where resides the "intelligence" of the agent. This piece has the responsability of determining which motion an agent describes, how to interact with the world and how to interact with other agents.

The main mechanism used to model a brain is a FSM, although any other device might be used such as an artificial neural network, which is the closer to reality. This computational box receives a set of information both from the body and from the environment. This includes the physical properties, the current state of the agent, a table of own attributes, a list with its neighbours and their information and a list of the incoming messages. After processing, the brain will communicate to the body what to do by returning certain parameters.

- **Prime Mover Force.** This is the instruction the brain sends to the body in order to perform a specific displacement. This will be explained into details in the next chapter during the section of forces, but it is basically the force that the muscles should apply on the body to move it in a certain direction and with certain acceleration. The direction is a free choice, but the magnitude should be influenced by the current strength to acquire a realistic motion

[1] Although the design and implementation of the approach are treated in chapter 7, the author believes that it is convenient to mention some implementations details which are important to understand how much flexibility or scalability this approach offers.

- **Heading.** The brain can set the heading of the agent too; this is normally established by the velocity, but there are some kind of motions, such as lateral movements, where the heading does not point in the same direction than the velocity.
- **Strength.** According to this model, it is the brain who decides how much strength the actions of the agent consume or, by the contrary, how fast it recovers. So the new strength is returned.
- **State.** The brain will tell to the body the new state, which might be the old one, or a new one if some transition was triggered.
- **Messages.** A list with the messages to send to the agent's neighbours. This messages have certain type of information detailed later, and if the receiver knows how to react upon it, its behaviour may be affected

This is where the usability and beauty of the approach arises. The behaviours are script based; therefore, having all the gears of the system spinning properly, any new brain can be added at any time. For the behaviours proposed, it was chosen to use a FSM, as a simple and effective mechanism, but any other more sofisticated method could be used. Different techniques such as movement prediction, statistical study, evolving algorithms, etc. might be employed for developing new agent behaviours and observe the emergent configuration of the crowd.

4.2.1 Capacity of Decision

This is what, under a physical consistency, drives the simulation. The ability that an agent has to take decisions is the main input to produce a simulation. If we have in mind any simple video game in where we control a character to interact with a world, we are taking decisions, tons of them. They might be long term ones (reach a target) or short term ones (this way seems a shortcut). Thus, the idea is automate this task. An infinite amount of things might influence every decision we take in real life: in which state we feel, where we are, how tired we are, which is our physical shape, who is surrounding us, what we know about who is surrounding

us, etc. In order to take accurate decisions a large amount of information needs to be taken into account, besides all the different possibilities available; that could produce extremely complex FSM's or other artifacts very difficult to handle.

4.2.2 The Illusion of Intelligence

Intelligence is a very complex concept. It is thought as a mental capability that involves things such as the ability to reason, plan, solve problems, think abstractly, comprehend complex ideas, learn quickly or learn from experience.

The *Artificial Intelligence* (AI) is the science which studies and develops intelligent machines and software. The general problem of simulating (or creating) intelligence has been broken down into a number of specific sub-problems related to the tasks mentioned in the previous paragraph. There is no established unifying theory or paradigm that guides AI research. Researchers disagree about many issues.

As a general rule, it can be considered that the decisions an agent takes and behaviour it presents will tell us if it is intelligent or not. Building a simple behaviour is a matter of minutes, building a very intelligent behaviour might require several times the time this thesis was done in.

4.3 Interaction among agents. Message passing

The other important feature essential for an authentic emergent group behaviour is how the individuals are interconnected, as stated in (Kleinrock, 2011). This is acheived by passing messages, which allows communication among the agents.

A message is a package which contains information that includes:

- **Agent Identification.** This is used to recognize the sender of the message.
- **Label.** This is the string of text which contains the information inherent to the message. A message might have any type of label, meanwhile the behaviour

designer provides the mechanisms to handle it. It might be from an "attack" message to a "shall we dance?" message.

- **Position.** The position of the sender. Take into account that having the agent identification, we can check all the information about it looking into the neighbours. But it might happen, that the sender is out of our vision radius. Imagine that we hear a far shout, the only we know about it is what it says and where it comes from.
- **Strength** The strength of the sender. This is use for practical purposes. Knowing the strength of the sender allows to take richer decisions at the time of handle it. For instance, if an "attack" arrives to us, we know the strength associated to it; or if a strong warrior sends a "follow" message in a battlefield, it may be a wise choice to pay attention to him.

4.4 Overview of Possible Behaviours

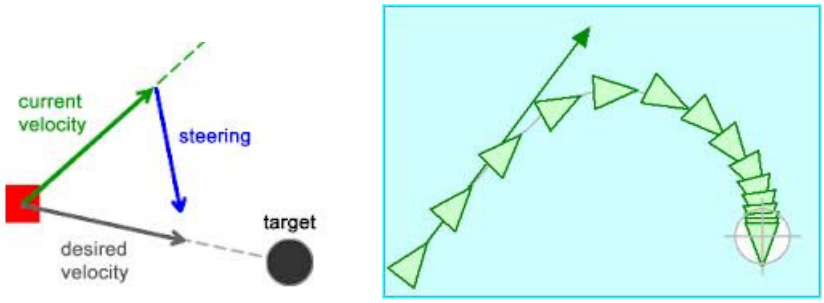
A behaviour is the range of actions and mannerisms made by organisms, systems, or artificial entities in conjunction with their environment, which includes the other systems or organisms around as well as the physical environment.

Many of some basic behaviours, which are thought as standard ones in the field of AI, were used directly or inderectly to develop some test behaviours for this thesis. The most classic one is the steering behaviour of the seek which directs the agent to a target position.

Let *pos* be the position, *velocity* the velocity, and *maxSpeed* the maximum speed of the agent; and *targetPos* the position of the target:

$$desiredVelocity = normalize(targetPos - pos)maxSpeed \tag{4.1}$$

$$seekForce = desiredVelocity - velocity \tag{4.2}$$



(a) *Calculating vector force* **(b)** *Shown behaviour*

Figure 4.2: *Seek Steering Behaviour*

This is one example among many other simple behaviours such as flee, arrive, pursuit, evade or wander (Buckland, 2005).

For this approach, more complex and varied behaviours which might involve the previous ones were developed. The method employed was basically logical reasoning and empirical testing, manipulating the motion by means of the prime mover force, besides heading, state, strength and all the other features that the brain supports.

In this way, we might have simple boids that essentially follow the rules of Reynolds' flocking algorithm, or more sophisticated ones. Droids that stand at a certain distance to the target to shot, warriors that send attacks to each others where the strong ones have more chances to reach the victory, dancers that employ forces softly to describe harmonious swings in couples, and more. The testing results will be exposed in further details in Chapter **??**: Applications and Results.

Chapter 5

Crowd Engine

5.1 Modelling the World

Once described the agent-model based, it is time to present where all the agents live with the superior form of a crowd. In order to model this, an entity called the *Crowd Engine* is presented. It is in charge of handling all the agents and take care of the consistence of the world. Although the agents and their interactions are what drive the simulation, this is the core of the system, where all the pieces meet together.

The Crowd Engine stores all the agents in the world and updates them in each tick. Besides its internal mechanism, it employs external components with specific responsabilities, which manage different aspects of the world.

5.2 Handling Large Amounts of Agents

One of the most important problems that a crowd simulation system may present, is the fact that it has to handle an enormous amount of agents. Routines such as neighbour finding or collision detecion will become a cartesian product of all the agents, which is equivalent to a computational complexity of $O(n^2)$. This means that if our world with 10 agents requires 20 ms per tick (50 FPS) in certain machine, if

it had 100 agents it would require 2000 ms (less than 1 FPS); and if it had 1000 agents, to update one single frame would take more than 3 minutes!

Thus, in this sort of circumstances a method called cell-space partitioning or, sometimes, bin-space partitioning is used. This is nothing more than indexing the space.

5.2.1 Cell Partition

With this method the space is divided into a number of cells or bins. Each cell keeps a list of all the entities it contains. This is updated every time an entity changes position. If an entity moves into a new cell, it is removed from its old cell's list and added to the current one. This way, instead of having to test every agent against every other, we can just determine which cells lie within an agent's neighbourhood and test against the agents contained in those cells.

As mentioned before, this is a way of indexing and ordering the agents in the world space. The weight of the process relies now in a search on an ordered structure (the cell partition),which is normally implemented as a very efficient algorithm over a b-tree.

The Cell Partition of this approach was inspired by the method proposed by Lee et al (Lee & Cho, 2012). They suggest a method that improves the efficiency of the search of the k-nearest neighbors in flocking behaviours.

It is used an heuristic based in this very simple statement: " two agents that are spatially close may share many common neighbours" (Lee, 2010). This thesis adopted this method, slightly modified by the author to find all the neighbours in a certain radius r (the vision radius).

The pseudo code of the algorithm is presented below. Notice that the Cell Partition developed for the approach is a 2D Cell Partition due to agents represent terrestrial creatures, but the design of the application allows scalate it at any time.

The routine $findAGentsInCells$ returns all the agents belonging to the cells

Algorithm 1 Efficient Neighbours Search Algorithm in radius

Require: list $agents$ that contains all the agents in the world
Ensure: update all the neighbours

```
1: checkedAgents = φ
2: for all agent1 in agents do
3:   if agent1 not in checkedAgents then
4:     r1 = agent1.getVisionRadius()
5:     agentsInCells = findAgentsInCells(agent1, r1)
6:     neighbours1 = findNeighboursInAgents(agent1, r1, agentsInCells)
7:     agent1.setNeighbours(neighbours1)
8:     checkedAgents.insert(agent1)
9:     for all agent2 in neighbours1 do
10:      if agent2 not in checkedAgents then
11:        r2 = agent2.getVisionRadius()
12:        if cellsUnion includes agent2 vision then
13:          neighbours2 = findNeighboursInAgents(agent2, r2, agentsInCells)
14:          agent2.setNeighboures(neighbours2)
15:          checkedAgents.insert(agent2)
16:        end if
17:      end if
18:    end for
19:  end if
20: end for
```

union which includes the vision radius of the agent, and $findNeighbours$ return the neighbours from those agents.

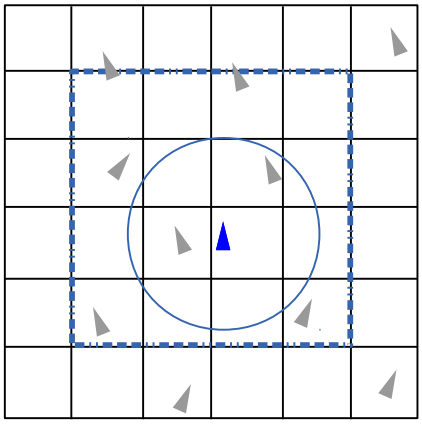


Figure 5.1: *$findAGentsInCells$ and $findNeighbours$ procedures*

According to the heuristic proposed in (Lee, 2010), it is likely to find the neighbours of a close agent in the same union of cells.

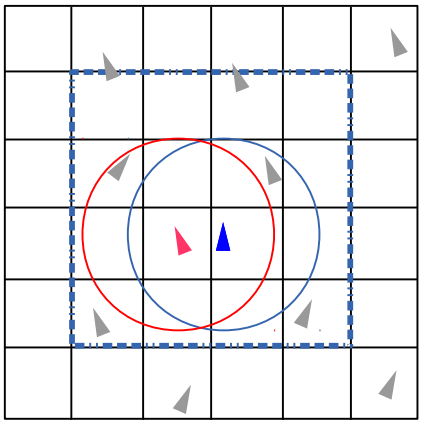


Figure 5.2: *heuristic for increasing the efficiency*

5.3 Virtual Force Model

As mentioned repeatedly along this document, the world follows a physical approach where forces take a fundamental role. The three main forces that will affect the

agents, imbuing an accelearion and therefore generating the motion are: the Prime Mover Force, the Gravity Force and the Friction Force.

5.3.1 Prime Mover Force

This is the only internal force. It comes from the agent and it could be comparable to the muscular force in order to move our body. More precisely, it is the brain who decides the value of this force and normally, the magnitude is influenced by current strength of the agent.

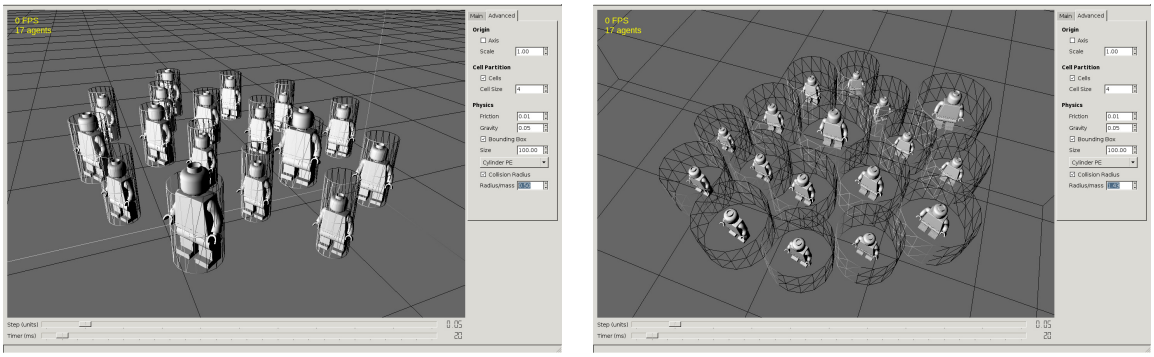
5.3.2 Gravity Force

5.3.3 Friction Force

5.4 Handling Messages

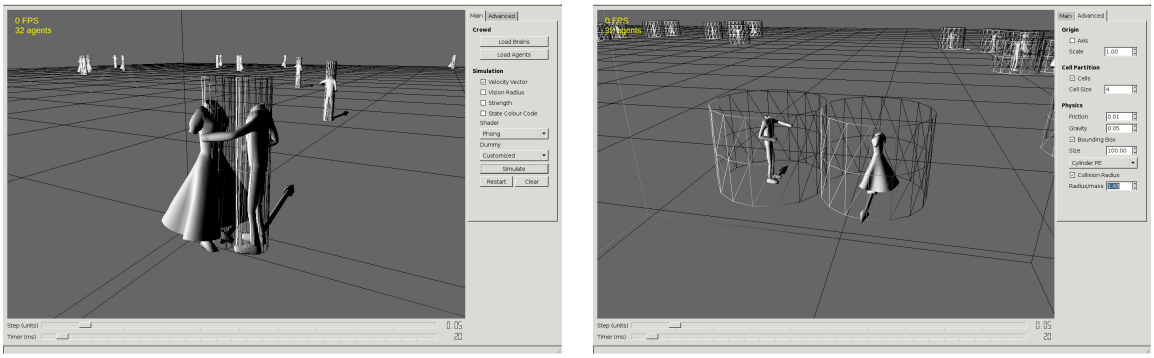
5.5 Collision Detection

5.5.1 Bounding Cylinder



(a) *Narrow Colision Radius* **(b)** *Wide Collision Radius*

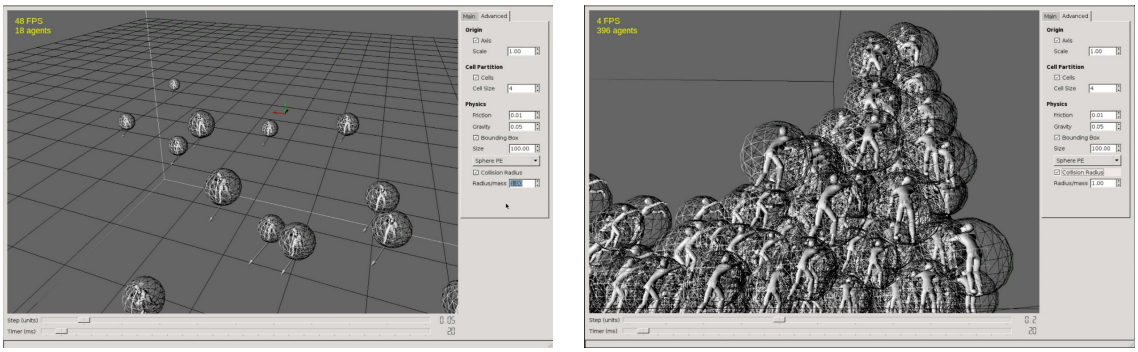
Figure 5.3: *Cylinder Collision Radius*



(a) *Romantic atmosphere* **(b)** *Problems in the relationship*

Figure 5.4: *Cylinder Physics Engine in Dancers*

5.5.2 Bounding Sphere



(a) *Marching Zombies* **(b)** *Mountaing of Zombies*

Figure 5.5: *Sphere Physics Engine in Zombies*

Chapter 6

Applications and Results

6.1 Possible real production pipeline

6.2 Test Behaviours

6.2.1 Crowds

6.2.2 Droid Wars

6.2.3 Zombie Apocalipse

6.2.4 A Battlefield

6.2.5 A Ballroom

6.2.6 One vs Many

6.2.7 Jumping Party

Chapter 7

Application Design and Implementation

7.1 Design

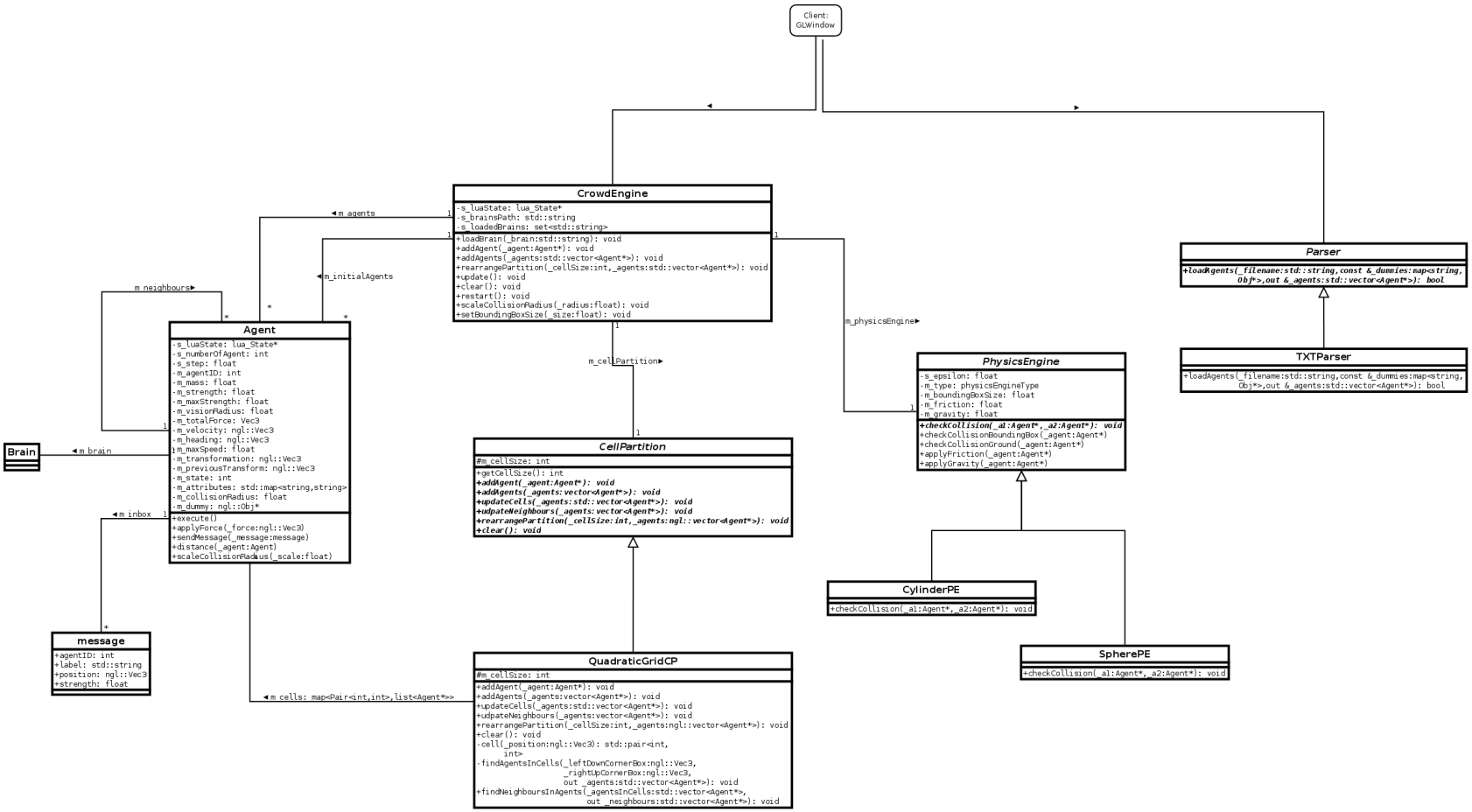


Figure 7.1: *Class Diagram of the Application*

7.2 Implementation

Chapter 8

Conclusion

Your conclusion should be structured like this. An introductory sentence then:

8.1 Summary

A summary of what has been achieved.

- Bullet points are good to clarify
- Bullet points are fun
- I ran out of ideas

8.2 Drawbacks

This is optional, some people put it in future work, I think it is better to have a separate section for it. Issues and bugs are different. Bugs are unexpected behaviour in the program, something not running as it should. Issues are more due to algorithm limitations. If the algorithm only works for meshes of less than 100k polygons, it is a limitation, not a bug. A program crashing if you move around in a particular

order is a bug. A render becoming ugly just on that particular spot in space, is a bug too.

8.3 Future work

What should be done in the future, what you would like to do.

- Become an astronaut
- Go out
- Make it work for real

Bibliography

M. Buckland (2005). *Programming Game AI by Example.* Wordware game developer's library. Wordware Pub.

L. Kleinrock (2011). 'What is emergent behaviour?'. Available from: http://curiosity.discovery.com/question/emergent-behavior [Accessed 10.08.2013].

J. M. Lee (2010). 'An efficient algorithm to find k-nearest neighbors in flocking behavior'. *Inf. Process. Lett.* **110**(14-15):576–579.

J. M. Lee & H.-K. Cho (2012). 'A simple heuristic to find efficiently k-nearest neighbors in flocking behaviors'. In *Proceedings of the 6th WSEAS international conference on Computer Engineering and Applications, and Proceedings of the 2012 American conference on Applied Mathematics*, AMERICAN-MATH'12/CEA'12, pp. 60–64, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS).

S. R. Musse & D. Thalmann (2001). 'Hierarchical Model for Real Time Simulation of Virtual Human Crowds'. *IEEE Transactions on Visualization and Computer Graphics* **7**(2):152–164.

S. R. Musse, et al. (2005). 'Groups and crowd simulation'. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA. ACM.

S. Patil, et al. (2011). 'Directing Crowd Simulations Using Navigation Fields.'. *IEEE Trans. Vis. Comput. Graph.* **17**(2):244–254.

C. W. Reynolds (1987). 'Flocks, herds and schools: A distributed behavioral model'.

In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pp. 25–34, New York, NY, USA. ACM.

C. W. Reynolds (1999). 'Individual-Based Models'. Available from: http://www.red3d.com/cwr/ibm.html [Accessed 10.08.2013].

C.-C. Wang & T.-Y. Li (2006). 'Evolving Crowd Motion Simulation with Genetic Algorithm'. In *Proc. 3rd Intl. Conf. on Autonomous Robots and Agents - ICARA'2006, Palmerston North, New Zealand*, pp. 443–448.

You can have different appendices (A. B. etc...) and sub sections too.