SHORTCUTS TO KAOS: COMPLEX BEHAVIOUR FROM SIMPLE DYNAMICS

Gary Ushaw & Gordon Bell, VIS Interactive, Dunfermline, Scotland.
Gary@vis-plc.com, gordon@vis-plc.com

Abstract

A game currently under development at VIS Interactive utilises some of the tenets of chaos theory and non-linear dynamics in its AI modules to take full advantage of the vastly increased processing power available to the programmer of next generation consoles such as Playstation2. This paper concentrates on how apparently complex behaviour of agents can be attained from disarmingly simple dynamics, while drawing parallels to the field of chaos theory. One of the basic tenets of chaos theory is that a mathematically simple set of dynamics can result in an endlessly complex and non-repeating behaviour; this bears a similarity to work in such AI fields as flocking and A-life.

Introduction

The arrival of the next generation of home console hardware presents the programmer of game AI with a new and unprecedented set of challenges. To date, the limited processing power available on game systems has been utilised primarily for graphics processing, collision routines and physics calculations, leaving only a small percentage of CPU time for artificial intelligence. If Sony's marketing department is to be believed, however, the "Emotion Engine" of Playstation2 will allow the implementation of character interaction which can pass the Turing test three times before breakfast and still leave room for more polygon manipulation than you could shake a pre-rendered stick at. In reality, it is likely that the processing power required for geometry transforms, graphical effects, animation, collision, etc, will expand to fill the newly available CPU time, leaving a similar percentage of processing resource available for AI as is currently the case. However, such a scenario still provides us with a vast increase in the amount of CPU power which the AI modules can tap into. A possible area of research which may be of use to the AI programmer with this extra CPU resource at his fingertips is chaos theory. This paper outlines how the increased processing power of a next generation console has been utilised on a game currently under development at VIS Interactive which will be referred to throughout this paper by its codename, "Kaos".

A discussion of chaos theory and its possible relevance to AI-programming for games is presented. A brief description of the game, Kaos, is provided, followed by an examination of how the various elements of the AI module are being implemented. The final section of the paper discusses possible areas of game AI which could benefit from this chaotic approach in the future.

Chaos Theory

Chaos theory is a relatively new branch of mathematics which studies the behaviour of a class of non-linear systems. Providing a rigorous definition of *chaos* is a can of worms which is best left unopened, however there a number of features which a chaotic system will tend to exhibit.

- Non-periodic but bounded. A chaotic system is not periodic, in that it will never repeat a particular variable state. However, the system is bounded (if the initial values fall within the "basin of attraction"), which means that the variables of the system remain within certain limits as the system evolves.
- Sensitive dependence on initial conditions. A very small change to a chaotic system will radically change the state of that system over time, compared to how the system would have evolved without that change. This is most popularly, and not entirely apocryphally, illustrated by the "butterfly effect", whereby it is said that a butterfly flapping its wings in the Brazilian rainforest can cause a

hurricane to strike Hong Kong harbour [1].

- Long term prediction is impossible. This is a direct result of the sensitive dependence on initial conditions described above. An error, however small, in measuring a chaotic system means that the model of that system used for prediction has marginally different initial conditions to the actual system. Hence the model and the system will diverge over time.
- Expansion and contraction of phase space.

A good populist account of the history and theory of the study of chaotic systems can be found in [2]. Further seminal works on the subject are found in [3], [4], [5] and [6]. The related field of fractals was most famously presented in [7].

Chaos Theory In Game AI

As game environments become more complex and more extensive, the behaviour which the player expects to see within that environment will also become more complex. An area of research which may be of use is that of chaos theory and non-linear dynamics. A number of aspects of a chaotic system would seem to be of interest to us in our quest to provide a deeper and more varied gaming experience.

Non-linear dynamics and chaos theory have shown that very complex behaviour can result from a simple set of equations. An example of this is the Lorenz system[8], which consists of three simple first order differential equations; iteratively solving these equations over time results in an output which never repeats itself, yet never leaves a bounded region of possible states. The non-repetitive yet bounded nature of such systems is clearly of interest to a game AI programmer seeking to portray a constantly interesting yet consistent character. An infinitesimal change to the seed values provided to such a system will produce a large change to the output after very few iterations, to the extent that the instantaneous state of the system will be on a completely different part of the attractor (an attractor can be thought of as the set of all possible states of the system). This is "sensitive dependence on initial conditions". This also would seem to be of interest to game AI programmers, as it describes a system capable of responding to the subtlest of changes in the player's interaction with the NPC.

It is well-known within the AI community that complex and organic-looking flocking can be achieved from simple rule-sets, whereby individual agents react to their nearest neighbours in a pre-determined way. This approach is very similar to some of the methodologies utilised in the study of non-linear dynamics and chaos theory. In particular, when studying a chaotic sequence of data (eg in order to measure its Lyapunov exponents) a nearest neighbour approach is generally taken, after embedding the data as an attractor on a manifold of suitable dimension. As processing power becomes more freely available some of the more numerically intensive methods of carrying out such studies may become applicable to games.

Taking this further, chaotic systems are regularly studied in four, five, or more dimensions. This could be of use to a game where an emotion such as anger, fear or contentment is attributed to individual characters on a sliding scale. Each of these emotions could be treated as an extra dimension. Nearest neighbours in the three dimensions of the game world would have an effect on one another's contentment level; conversely, neighbours in the "contentment dimension" would have a greater effect on one another's movement. In the study of chaotic systems, nearest neighbours are defined as those occurring within a n-dimensional hyper-sphere around the point of interest. If each emotion to be represented is on a sliding scale, then each can be represented as an extra dimension of the hyper-sphere, with the centre occurring at the current position and emotional state of the agent being processed.

Project "Kaos": What is it?

Kaos is a game involving as many as three hundred autonomous agents on screen at once, set within an expansive urban environment. Each agent is an animated polygonal model of a human being. The environment is a working three-dimensional city in which agents go about their daily business until the player character interacts with them. Within the city there are literally thousands of agents, all completely autonomous, all under control of the AI module. This is not a strategy or resource management game such as Sim City or Populous, but a game where the player is in direct control of a specific protagonist, viewed from a third person perspective, as he navigates a game arena filled with characters which can be influenced on an individual basis. At first sight, any AI programmer of merit ought to balk at the thought of implementing what has just been described, especially given the constraints of time, programming resource and quality necessitated by a commercial product. However, the over-riding salient fact to be borne in mind from the above description is contained within the first four words: “Kaos is a game”. It is not a simulation, it is not a research project, it is not a fly-on-the-wall documentary or an interactive movie. If a piece of AI behaviour is convincing within the context of the game, then it is perfect and needs no further refinement or complication. As such, the underlying AI approach must be both simple and flexible, particularly given the large numbers of agents to be handled.

AI Implementation on Kaos

In the case of Kaos, the increased processing power available for use by the AI code is largely taken up by the sheer numbers of agents which need to be processed for this game. What follows is a description of the various parts of the AI code which have been implemented in order to provide this functionality. Particular note is taken of those aspects of the AI code which have included aspects of chaos theory, or which bear discussion in relation to it.

Two things have become apparent during this implementation. Firstly, to provide AI for a game of this nature, it has been advantageous to combine the best of all AI approaches, taking what is suitable from each in order to solve a particular problem. Secondly, we have repeatedly found that apparently complex behaviour can arise from disarmingly simple mechanics. This is a basic tenet of chaos theory, which the authors have unashamedly hijacked for the purposes of this paper. The converse of this statement is that a simple set of equations or rules can result in highly complex behaviours; the trick is to find that crucial rule set.

- Grid Squares

Kaos involves a large number of agents on screen at once, so an efficient collision routine was required. Fortunately it is set in an American city (and none of the characters have the power of flight) so a two-dimensional grid-based collision system was ideally suited to our purposes. Such an approach has the added advantage of providing us with access to a wealth of knowledge on AI implementation. The most obvious area in which the grid square approach provides us with a clear solution is the challenge of path-finding. The majority of literature available on path-finding utilises an environment based on grid squares, so an adapted form of A* was easily added to the project. A Line Of Sight algorithm is also ideally suited to the grid square paradigm, so we can quickly give our characters a sense of their surroundings. Line of sight can be achieved using the algorithms described by Bresenham, since we have knowledge of whether or not each square is solid, and a square equates to a pixel in Bresenham’s algorithm. This is considerably more efficient than resorting to some form of ray-tracing to check a line of sight. The grid squares provide further advantages for camera functionality, testing the locality of objects or characters, and display clipping.

Clearly there is little direct cross-over between the grid-square system and chaos theory. However, any code which relies on finding nearest neighbours will benefit greatly if the environment is sub-divided; in Kaos, the grid-squares provide this extra efficiency.

- Scripting Language

It was decided at an early stage that much of the functionality required by Kaos, in terms of both game-play scenarios and environmental colour, could best be supplied through the use of a scripting language. This tool provides the Game Designers with a great deal of control over the game and the environment while remaining extremely flexible. It is also a simple matter to extend functionality by adding new scripting commands. An initial set of simple scripting commands were provided, such as "Send_To", "If_Arrived_At", "Is_Object_Damaged", "Start_Timer", etc. This initial command set allowed the game designers to script complex behaviour and to experiment with this behaviour without having to resort to Software support. Once a satisfactory piece of behaviour has been settled on then, if necessary, a new scripting command can be added to implement the behaviour more efficiently. It's important to note that the scripted behaviour must work in conjunction with the AI code, so that things like collisions, generic behaviours, etc are not left in the hands of the scripters. This is an example of a good and well-planned AI approach making everybody's job a lot easier.

Again there is virtually no input from chaos theory here, as a scripted piece of behaviour generally needs to be very specific for gameplay or scenario reasons. It would, however, be possible to provide script commands for introducing chaotic behaviour, or changing the nature of it, as specific game events occur.

- Map Properties: embedded AI

Information of use to non-player characters is embedded in the game environment, in the form of object properties, grid-square properties and area properties. This reduces the amount of information which needs to be attached to each agent, and relieves the amount of donkey-work which would otherwise be involved in scripting. For example, in order to decide whether an object can be picked up and thrown, an agent just needs to check the properties of that object; similarly, in order to find a target to throw the object at, the agent searches the vicinity for an object with suitable properties. Properties attached to grid squares define how agents react on those squares (for example a vehicle won't drive on a grid marked as sidewalk, unless specifically scripted to do so). Similarly, wider area properties can be attached which will affect different characters in different ways (so a character from a well-to-do district is unlikely to wander into a ghetto, whereas a group of inhabitants of the ghetto may cause some damage in Downtown).

Embedding a chaotic property can have an interesting effect here. The reaction of NPCs to this embedded property will change over time, as the property changes, but remain within a defined set of boundaries. For example, a broken sewer would repel NPCs, depending on how smelly it is. This "smelliness factor" is embedded in the map as a chaotic property; hence the lengths to which NPCs go in order to avoid the sewer varies over time, as if the smell is billowing out in a non-repetitive manner. This only requires storing the current state of the chaotic system used (typically three floats), and a piece of code to iterate these values (typically a handful of multiplications and additions).

- Living City

One of the biggest challenges presented by Kaos was to successfully imply a fully functional city. If the player just stands still and watches, a wealth of independent behaviour ought to be apparent: pedestrians walking down the street, stepping into a shop, waiting for a bus, crossing the road, vehicles stopping at traffic lights, giving way at junctions, etc. This has been achieved through a combination of the three elements described above. The grid square approach results in a straightforward way of deciding which parts of a map are available for pedestrians (ie sidewalks) and which are available for vehicles (ie roads). This information is embedded within the map, so that any agent can easily find out about the properties of its current grid square and of surrounding grid squares. Similarly properties relating to the occupancy of buildings (how many people are inside), whether a building can be entered (eg a bar), where to queue for a bus or cross a busy street are all embedded within the map objects and grids. Further complexity of living city functionality is provided by specifically scripted scenarios which may trigger after a particular set of player actions, or just randomly if the player is sufficiently close to notice.

The possibilities for including chaotic dynamics in the living city behaviour are enormous. As has been discussed, chaos provides a non-repeating source of raw data which is smooth and "organic". This is not a bad description of the sort of behaviour that is required for a convincing living city. If the player chooses to follow a NPC at random as it traverses the city, it is highly preferable that this character does not simply walk round and round a repeated route. Adding chaos to the NPC's actions and choices prevents this from happening.

- Crowd Dynamics.

The movement of crowds of people forms an important aspect of Kaos and it is here that the concept of achieving complex behaviour from simple dynamics comes into its own. The streets of the city represented in Kaos must contain, at times, a heaving throng of people, interacting with one another on an individual basis but also behaving as a whole in a convincing manner, replicating how large crowds of people have been observed to act.

In such a situation, although each crowd member is an autonomous agent, it must take cues for its behaviour from the agents immediately surrounding it. This methodology is well known within the AI community as a means to produce complex and organic flocking from a simple rule set. Each agent supplies a small repulsive force to its neighbours in order to prevent collisions. Coupled with this are larger less-localised forces providing attractions and repulsions to a crowd, resulting in an ebb and flow of the mass of the crowd. As an agent is pushed around by the surrounding forces, its target position is also pushed around, preventing it from stutteringly trying to achieve the same position, or relative position, time after time. This has resulted in a crowd which appears to jostle internally as individual members seem to push and shove in various directions. This is especially effective when a new element is integrated into the crowd, or when the crowd is moved into a more confined space.

On top of this jostling dynamic is a dynamic with a longer term goal, such as moving the crowd along a street, or through a narrow gap. It has been found that increasing the amount of damping and lag on this higher-level dynamic in how it affects individual agents, can greatly change the over-all feel of the movement. Examples range from a lightly damped shoal of fish to a much more heavily damped crowd of people. This multi-layered approach to crowd dynamics, coupled with the underlying concept that complex behaviour can arise from a simple set of rules, if those rules are correctly tailored, bears a striking resemblance to the study of non-linear dynamics and chaos theory.

As an aside, and on a purely speculative level, it could be suggested that this crowd behaviour exhibits chaotic dynamics. One of the ways of defining a chaotic system is that it has both a positive and negative Lyapunov exponents. Loosely translated, this means that the elements of the system are subject to both expanding and contracting forces. The crowd members described above are subject to an expanding force (as they repel one another to avoid collisions), as well as a contracting force (as they try to move to a particular location or group around a target).

- Police Tactics

All of the above aspects of the AI code come together in the control of the police behaviour within the game map. The police characters in Kaos must function individually and as part of a larger group or over-all tactic. Individual police characters use scripting commands and embedded AI information in the same way as other NPCs within the game. The grid structure further provides a quick way for a police character to assess the current situation, checking how many of his fellow officers are in the immediate vicinity, how much of a threat is being presented, and how vulnerable local property and objects may be. This information is processed on an individual level, for a particular police character's immediate actions, and at a higher level to provide grouping together of police NPCs and teamwork. This once more demonstrates a two-tier approach to the AI functionality and is again implemented through a simple inter-linked rule set. This simple set of rules leads to a very diverse behaviour by the police characters, depending on the actions, both recent and long-term, of the player character and other NPCs.

No chaos theory has been used in the police code as such, since they are, by definition, an element of the game which is tightly governed by rules and discipline. However, since they are controlled by a completely stochastic system, and are in an environment with chaotic elements, their behaviour and responses are highly varied, due to the large number of situation combinations they can come into contact with in a chaotic environment.

The Future Of Chaos In Game AI

One element of non-linear dynamics that has not been mentioned so far in this paper is that of fractal geometry. Many display engines exist which are based on a fractal approach, for creating landscapes, trees, etc. There may, however, be some mileage in applying fractals to AI coding in games. One area of interest may be in populating an extensive environment with a range of character types and character traits. A fractal map could be used to define, for example, the "anger level" of each character as he is inserted into the environment. Similarly, fractal mapping could be used for embedded AI properties such as our "smelliness factor" on a sewer-world.

It is likely that many games are already in existence or development which include AI routines providing what could be described as chaotic behaviour, whether or not the developers think of it in those terms. Game AI coders are constantly striving for complex behaviour from simple rules.

Although some aspects of chaos theory have been incorporated in the game, Kaos, it has become apparent that there are many possibilities for future development in this area, both in games of a similar type to Kaos and in other genres. A number of these possibilities have been discussed throughout the paper, and many more will no doubt occur to the interested reader.

Conclusions

In planning and coding the AI functionality for Kaos, a number of parallels have been drawn with the tenets and analysis of chaos theory. In particular, it is apparent that there are similarities in the way complex behaviour can arise from simple equations. It seems likely that, as the requirement for ever more complex character interactions continues, coupled with the increased processing power presented by successive generations of gaming hardware, that the field of chaotic studies may provide insights and resources for the gaming AI community.

References

[1] E.Lorenz, The essence of chaos, UCL Press, 1993.

[2] J.Gleick, *Chaos: making a new science*, Cardinal, London, 1987.

[3] M.Feigenbaum, "Qualitative universality for a class of nonlinear transformations", *Jnl of Statistical Physics*, vol 19, pp 25-52, 1978

[4] T.Y.Li and J.Yorke, "Period three implies chaos", *American Mathematical Monthly*, vol 82, pp 985-92, 1975

[5] R.May, "Simple mathematical dynamics with very complicated dynamics", *Nature*, vol 261, pp 459-67, 1976

[6] D.Ruelle, "Strange Attractors", *Mathematical Intelligencer*, vol 2, pp126-37, 1980

[7] B.Mandelbrot, *The fractal geometry of nature*, Freeman, New York, 1977.

[8] E.N.Lorenz, "deterministic non periodic flow", *Jnl Atmospheric Science*, vol 20, pp 130-141, 1963.