An Emergent Approach to Game Design – Development and Play

Penelope Sweetser

B. InfoTech (Hons), G.C.Ed.

A thesis submitted for the degree of Doctor of Philosophy

School of Information Technology and Electrical Engineering

The University of Queensland

June 17, 2005

Statement of Originality

The work presented in this thesis is, to the best of my knowledge and belief, original, except as acknowledged in the text, and the material has not been submitted, either in whole or in part, for a degree at this or any other university.

Penelope Sweetser

Abstract

Player enjoyment is the single-most important goal of games. Games that are not enjoyable are not bought or played. Within games, enjoyment of the gameplay hinges on the game world. However, game worlds are often static and highly scripted, which leads to restricted and shallow gameplay that can detract from player enjoyment. It is possible that player enjoyment could be improved by the creation of more flexible game worlds that give players more freedom and control. One way to create more flexible game worlds is through the use of an emergent approach to designing game worlds. This thesis investigates an emergent approach to designing game worlds, as well as the issues, considerations and implications for game players and developers.

The research reported in this thesis consisted of three main components. The first component involved conducting a focus group and questionnaire with players to identify the aspects of current game worlds that affect their enjoyment. The second component of the research involved investigating an emergent approach to designing game worlds, in which the Emergent Games Engine Technology (EmerGEnT) system was developed. The test-bed for the EmerGEnT system was a strategy game world that was developed using a 3D games engine, the Auran Jet. The EmerGEnT system consists of three main components: the environment, objects and agents. The third component of the research involved evaluating the EmerGEnT system against a set of criteria for player enjoyment in games, which allowed the system's role in facilitating player enjoyment to be defined.

In the player-centred studies, it was found that players are dissatisfied with the static, inconsistent and unrealistic elements of current games and that they desire more interactivity, realism and control. The development and testing of the EmerGEnT system showed that an emergent game world design, based on cellular automata, can facilitate emergent behaviour in a limited domain. The domain modelled by the EmerGEnT system was heat, fire, rain, fluid flow, pressure and explosions in a strategy game world. The EmerGEnT system displayed advantages relating to its ability to dynamically determine and accommodate the specific state of the game world due to the underlying properties of the cells, objects and agents. It also provided a model for emergent game worlds, which allowed more complexity than emergent objects alone. Finally, the evaluation of enjoyment revealed that incorporating an emergent game world (such as the EmerGEnT system) into a game could improve player enjoyment in terms of concentration, challenge, player skills, control and feedback by allowing more intuitive, consistent and emergent interactions with the game world.

The implications of this research are that cellular automata can facilitate emergence in games, at least in a limited domain. Also, emergence in games has the potential to enhance player enjoyment in areas where current game worlds are weak. Finally, the EmerGEnT system serves as a proof of concept of using emergence in games, provides a model for simulating environmental systems in games and was used to identify core issues and considerations for future development and research of emergent game worlds.

List of Publications

**Sweetser, P. & Wyeth, P. (in press) GameFlow: A Method for Evaluating Player Enjoyment in Games. *ACM Computers in Entertainment* 3 (3).

**Sweetser, P. & Wiles, J. (in press) Combining Influence Maps and Cellular Automata for Reactive Game Agents. *6th International Conference on Intelligent Data Engineering and Automated Learning*.

**Sweetser, P. & Wiles, J. (in press) Using Cellular Automata to Facilitate Emergence in Game Environments. 4th *International Conference on Entertainment Computing*.

**Sweetser, P. & Wiles, J. (2005) Scripting versus Emergence: Issues for Game Developers and Players in Game Environment *Design. International Journal of Intelligent Games and Simulations* 4 (1), pp. 1-9.

**Sweetser, P. & Johnson, D. (2004) Player-Centred Game Environments: Assessing Playing Opinions, Experiences and Issues. Entertainment Computing - ICEC 2004: Third International Conference, *Lecture Notes in Computer Science*, 3166, pp. 321-332.

Sweetser, P. (2004). How to Build Neural Networks for Games. In Rabin, S. *(Ed.), AI Game Programming Wisdom 2.* Hingham, MA: Charles River Media, Inc.

Sweetser, P. (2004). How to Build Evolutionary Algorithms for Games. In Rabin, S. *(Ed.), AI Game Programming Wisdom 2.* Hingham, MA: Charles River Media, Inc.

Sweetser, P. (2004). Strategic Decision-Making with Neural Networks and Influence Maps. To be published in Rabin, S. *(Ed.), AI Game Programming Wisdom 2*. Hingham, MA: Charles River Media, Inc.

Sweetser, P., Johnson, D., Sweetser, J., & Wiles, J. (2003) Creating Engaging Artificial Characters for Games. *Proceedings of the Second International Conference on Entertainment Computing*. Pittsburgh, PA: Carnegie Mellon University.

Sweetser, P. & Dennis, S. (2003). Facilitating Learning in a Real Time Strategy Computer Game. *Entertainment Computing: Technologies and Applications* (eds. Ryohei Nakatsu and Junichi Hoshino). Kluwer Academic Publishers, Boston.

Johnson, D., Gardner, J., Wiles, J., Sweetser, P. & Hollingsworth, K. (2003). The Inherent Appeal of Physically Controlled Peripherals. *Entertainment Computing: Technologies and Applications* (eds. Ryohei Nakatsu and Junichi Hoshino). Kluwer Academic Publishers, Boston.

*** Publications related specifically to this thesis.*

This thesis is dedicated to my partner, Peter Surawski.

Acknowledgements

I would like to acknowledge several sources for funding throughout my PhD. Firstly, the School of ITEE for a departmental scholarship for my first year, as well as ongoing tutoring and lecturing throughout my studies. Second, the UQ Graduate School for providing my UQGSS scholarship for two years. Also, I would like to acknowledge the Key Centre for Human Factors and Applied Cognitive Psychology and the Australasian CRC for Interaction Design for research work that supported my study.

I would like to thank my advisor, Janet Wiles, for teaching me about research and helping to bring the best out of my work. Also, I thank my associate advisor, Peta Wyeth, for her expertise in human-computer interaction and feedback on my thesis. Also, Daniel Johnson acted as an advisor on many occasions, giving me support with experimental design and statistical analysis. I also thank my sister, Jane Sweetser, for her help and advice on psychological data and analysis.

I would like to thank my friend and colleague, Penny Drennan, for her support, feedback and friendship throughout this period.

Finally, and most importantly, I would like to thank my family. Firstly, I thank my partner, Peter Surawski, for his ongoing support, understanding and encouragement throughout my study. I would also like to thank my parents, Bill and Gay Sweetser, for their support and encouragement throughout my study, as well as instilling me with the desire to reach for the stars and making the sacrifices that allowed me to do so.

Contents

LIST OF TABLES

LIST OF FIGURES

1 Introduction to Emergence in Game Worlds

The future of game development is towards more flexible, realistic and interactive game worlds. Games have become increasingly more realistic visually, with graphically lifelike and detailed characters, creatures and game worlds. However, the environments, objects and agents in these game worlds are often static, lifeless and afford limited interaction. There has been no research into how these game worlds affect player enjoyment and players are now seeking more realistic and interactive behaviour from these game elements. However, the current methods of game design are unable to accommodate this type of behaviour. Consequently, it is now necessary to assess the issues that players have in current game worlds and to search for a new approach to game design that will allow game worlds to accommodate the needs of the players, affording more flexible and interesting behaviour and gameplay.

1.1 Narrative versus Gameplay

Computer games can be broken down into two fundamental elements, the gameplay and the narrative. The question of free and open gameplay versus controlled, scripted narrative in games has aroused much debate by game developers and researchers in recent years. One side of the argument, often referred to as "ludology", claims that the enjoyment of games hinges on rules and gameplay and that games should not be designed to be stories (Juul, 2000). The other side, called "narratology" or "narrativism", believes that narrative and stories should be the foundation of games (Mateas, 2002). However, neither element constitutes a satisfying gaming experience alone. Without gameplay, games are simply stories that are created by the game developers. Similarly, with no narrative, games are virtual sandboxes without goals or motivation.

Narrative is the story that is told by the game, through cutscenes (movies), quests, characters, problems and the flow of the game. Narrative provides the "why" for the game, giving the player background and motivation to become involved with the game world and its inhabitants. Narrative also provides the "what" for the game, leading the player through the game, giving them specific quests, objectives, goals and problems to solve, making up the content of the game. The narrative determines who (i.e. game characters) the player will interact with, what problems they will face, and possibly in what order. The player assumes a role in the game's story, which the developer writes and the player acts out by going through a series of planned interactions. Narrative is scripted by the game developer and without it, the player has no reason or direction (i.e. why or what) and is just interacting in a sandbox world with no rules, goals or motivations.

Gameplay provides the "how" to the narrative's "what" and "why". The gameplay is how the player interacts in the game world, how they solve problems and how they play the game. Interactions in the game world are the foundation of the gameplay and the types of interactions depend on the game genre. In role-playing games, interactions include dialogue, using spells or abilities, collecting items, gaining experience and upgrading abilities. In real-time strategy games, interactions include building units and buildings, collecting resources, upgrading, attacking and defending. The gameplay is made up of how the player uses these basic interactions to solve problems, achieve goals and advance through the game.

Most games are either narrative-based or gameplay-based. Designing games that are narrative-based involves predefining a storyline that the player follows through the game. Game genres such as such as role-playing games (e.g. Elder Scroll III: Morrowind), action-adventure games (e.g. Tomb Raider) and the campaigns in real-time strategy games (e.g. Warcraft III) are narrative-based games. Designing games that depend on gameplay involves giving the player rules of play, options for actions and allowing them to develop their own strategies and game experiences. Genres that are based on gameplay include simulation games (e.g. SimCity) and real-time strategy games (e.g. Age of Mythology).

Interestingly, the divide between narrative and gameplay-based games brings two different approaches to game development. In narrative-based games, it is not only the narrative that is scripted, but also the gameplay. Conversely, in gameplay-based games, the gameplay is open but there is a lack of narrative, with only general goals for winning or success (e.g. kill the enemy). Therefore, in current games, either the gameplay and narrative are both scripted or both open (see Table 1.1). For narrative-based games, the result is heavily scripted games that successfully tell a story, providing depth and background, but limited freedom or control for the player. On the other hand, open games provide a sandbox world where the player has a great deal of freedom and control, but lacks motivation and goals, except for "experimentation". It seems that the optimal solution would be to have open gameplay within scripted narrative.

Table 1.1. Gameplay and narrative. The gameplay and narrative in most games are both scripted or both open

	Scripted Games	**Open Games**
Gameplay	Player plays a scripted role - no freedom or control	Player decides "how" to play - freedom in interactions and strategies
Narrative	Game has a scripted story - provides a "what" and "why"	Game is a sandbox - no goals or motivation
Genres	Role-playing, first-person shooter	Real-time strategy, simulation
Examples	Diablo, Wizardy, Might & Magic	The Sims, SimCity, Warcraft

1.2 Scripted Gameplay

The current approach to developing narrative-based games involves scripting a specific narrative flow, as well as specific interactions and behaviour for specific situations (Church, 2002). This scripted approach gives rise to problems for game developers and game players. The problems for game developers include effort in design, finding and fixing bugs, and difficulties in modifying and extending the system. For the players, problems include unintuitive and inconsistent interactions, a slow learning curve and an inability for players to freely express themselves.

In scripted systems, the game developer must design, implement and test specific game elements individually, as well as manually define possible courses of action through the game (Church, 2002; Smith, 2001). Scripting requires a great deal of effort and time in designing and testing, as each instance of a game element must be implemented and tested individually (Smith, 2002). Although scripted systems are relatively easy to create initially, they are harder to modify and scale poorly (Church, 2002). Changes and extensions to the system require revision of any aspect of the game that the change affects (Church, 2002). However, the benefit for game developers is that scripting the game world offers total structure and creative control for designers, empowering them to create a specific narrative or flow for the game.

The considerations for the players include the inconsistencies that occur in scripted worlds, which can break the player's immersion or suspension of disbelief that the game is real (Hecker, 2000). Inconsistencies also make learning how to interact in the game world more difficult. Players can only interact with game objects in prescripted ways, which means that objects behave less like real world objects and objects that appear similar can behave very differently (Smith, 2002). Additionally, the players only have a few prescripted interactions or courses of action to choose from (Smith, 2001), which limits player expression and creativity. The result can be an inconsistent, unrealistic, unintuitive and confusing game world, where the player has no freedom or control.

1.3 Open Gameplay

The current method of scripting game worlds prohibits players from experiencing open, free gameplay and the resulting control and agency. Additionally, scripting game worlds is difficult and time-consuming for game developers. However, there is currently no alternative design approach that can facilitate open gameplay in narrative-driven game worlds. The opportunity exists to develop a new method of designing game worlds that allows players to create their own interactions and strategies, as well as reducing the time and cost for game developers. These game worlds should be designed globally (not specifically), providing only rules and boundaries for player interactions, rather than explicit requirements. The question that arises is how can open, emergent game worlds that allow open gameplay within scripted narrative be made possible?

Some games have allowed more freedom and variation through property-based objects and rules for how the objects interact. For example, in the simulation game The Sims (Electronic Arts, 2000), intelligence is embedded into objects in the environment, called “Smart Terrain”. The objects broadcast properties to nearby agents to guide their behaviour (Woodcock, 2000). Similarly, the game objects in the first-person shooter game Half-Life 2 (Valve, 2004) uses named links between pieces of content called “symbolic links” (Walker, 2004) that define the properties of the objects and determine how they can be affected by players and other objects. Using this global design, the objects behave more realistically and are more interactive as they are encoded with types of behaviour and rules for interacting, rather than specific interactions in specific situations.

Another way that previous games have allowed more freedom and variation is through sheer size of the game world and number of possibilities. For example, the role-playing game The Elder Scrolls III: Morrowind (Bethesda, 2002) includes an enormous world with numerous characters, objects, events and quests. Although the game is heavily scripted, the number of characters to talk to, quests to take, places to explore and items to collect makes the game seem far more open and complex. Also, the real-time strategy game Age of Mythology (Microsoft, 2002) allows the player to choose from three different civilisations, from three different major deities in each civilisation and a new minor deity each time they advance to a new age. Each of these choices gives the player access to different units, upgrades, powers and mythology, affording the game increased variability and flexibility.

1.4 Emergent Game Worlds

Even though some games have allowed more open gameplay through property-based game objects (e.g. Half-Life 2) or increased game content (e.g. Elder Scrolls III: Morrowind), the actual environment in these games is still static. The players have more freedom in interacting with objects and more choices or content to keep them occupied, but there is still no solution for the game worlds as a whole. The game objects are only a small part of the game world, which also includes the environment (e.g. buildings, terrain and scenery) and game agents (e.g. characters or units). The game environment in most games is inert and unresponsive to players, objects and events. Also, the agents in most games are unaware of their surroundings and do not react to changes in the game world. Both of these aspects reduce the player's freedom, impact and control in the game world, as well as making the game world seem lifeless and flat. The question, therefore, is how can game worlds as a whole (e.g. environment, objects and agents) be made more open and emergent?

Emergent behaviour occurs when simple, independent rules interact to give rise to behaviour that wasn't specifically programmed into a system (Rabin, 2004). There are a variety of techniques from complex systems, machine learning and artificial life that have the potential to facilitate emergent behaviour in games. Some examples of emergent techniques that can and have been used in games are flocking, cellular automata, neural networks and evolutionary algorithms. Flocking is an artificial life technique for simulating the natural behaviour of a group of entities, such as a flock of birds or school of fish (Reynolds, 1987). Cellular automata are spatial, discrete time models that are used to simulate complex systems (Bar-Yam, 1997). Neural networks are machine learning techniques inspired by the human brain that are used for prediction, classification and decision-making (Haykin, 1994). Finally, evolutionary algorithms are techniques for optimisation and search that use concepts from natural selection and evolution to evolve solutions to problems (Mitchell, 1998).

Each of the described techniques can be applied to games in varying ways. Flocking has been used in games as an alternative to scripting the movement of entities in a group individually. For example, Half-Life (Valve, 1997) uses flocking to give its monsters more lifelike responses (Woodcock, 2003). Neural networks have been used for character behaviour and decision-making (see Sweetser (2004b) on the CD for a review). For example, BattleCruiser: 3000AD (Smart, 1996) uses neural networks to control non-player characters, as well as to guide negotiations, trading and combat (Woodock, 2003). Finally, evolutionary algorithms have been used to evolve strategies and monsters (see Sweetser (2004c) on the CD for a review). In Cloak, Dagger and DNA, evolutionary algorithms are used to guide and refine opponent behaviour (Woodcock, 2003).

Even though each of these techniques have been used in games, neural networks, evolutionary algorithms and flocking are not appropriate to modelling game environments as they are not spatial techniques. Neural networks are techniques for decision-making and classification, evolutionary algorithms are used for optimisation and search, and flocking is used for simulating group behaviour. However, cellular automata are suitable to modelling game environments as they are designed to provide spatial representations, although they are previously unused in games.

1.5 Cellular Automata in Games

Insight to modelling game environments can be gleamed from environmental simulations. Most approaches to modelling real-world phenomena in virtual worlds aim to develop accurate, error-free models. These models are often developed for the purposes of accurately simulating forest fires (Hargrove, Gardner, Turner, Romme & Despain, 2000; Consolini & De Michelis, 2001; Barros & Mendes, 1997) or visually realistic smoke (Stam, 2000; Treuille, McNamara, Popović & Stam, 2003; Fedkiw, Stam & Wann Jensen, 2001) and fluid flow (Stam, 2003). Equations and models that are commonly used in these applications include Navier-Stokes equations (Stam, 2000), Euler equations (Fedkiw, Stam & Wann Jensen, 2001), the Stable Fluids algorithm (Stam, 2003) and cellular automata (Barros & Mendes, 1997; Consolini & De Michelis, 2001).

In games, it is not necessary to use these complex, computationally-expensive methods as game worlds do not need to be accurate and error-free. Rather, they need to be credible and acceptable to the player. Game worlds only require environments and physics that approximately model reality. Forsyth (2002), a game developer, has identified ways in which environmental processes such as air, water, flow, heat and fire can be simplified for games using cellular automata. Also, he has formulated some example equations for these processes in human-sized (e.g. first-person shooter) games. However, there is no implementation (in games or research) of using cellular automata for modelling real-time game environments. Furthermore, Forsyth's suggested equations are only for individual game environment processes (e.g. fire or water) and do not address the integration of these processes into a complex system or the incorporation of game objects, agents and players.

Using cellular automata in games seems to be a good idea in theory, but it is necessary to determine how cellular automata can be made to work in practice. The lack of cellular automata in current games gives rise to the questions of whether cellular automata are appropriate for use in game systems and to what extent cellular automata can facilitate emergent gameplay. Additionally, if it is possible to make game worlds more emergent through the use of cellular automata, it is also necessary to consider the implications there will be for game developers and players. How will emergent game worlds affect the development process for game developers (e.g. time, effort, difficulty, control) and the enjoyability of the games for players?

1.6 Thesis Overview

The overall aim of the research reported in this thesis was to investigate emergence as an alternative to the current scripted approach to game development and the resulting implications for game players and developers. The specific aims of the research were:

- to define the issues associated with interacting in game environments by incorporating the players' perspective
- to evaluate the potential of cellular automata to facilitate emergence in game environments
- to design, implement and test an emergent game system based on cellular automata
- to determine how an emergent game system based on cellular automata will affect developing and playing games

The thesis is divided into three sections: (i) identifying the player-centred issues of interacting in game worlds, (ii) designing, implementing and testing the Emergent Games Engine Technology (EmerGEnT) system, and (iii) evaluating the facilitation of player enjoyment in the EmerGEnT system.

1.6.1 Part I: Identifying the Player-Centred Issues of Interacting in Game Worlds

The issues associated with scripted game worlds have been well-defined by game developers. However, there is no empirical evidence of how scripted game worlds affect player enjoyment. Therefore, the aim of the first section of the research was to assess the players' perspective on the issues that impact on player enjoyment when interacting in game worlds and to determine to what extent the players' perspective supports or differs from the insights provided by game developers. The method used consisted of two player-centred studies, including a focus group and a questionnaire (see Chapter 3). The player-centred studies aimed to identify and define the issues of interacting in game worlds from the players' perspective and to investigate how the issues relate to the context of play (i.e. game-type preference and player experience). The player-centred studies allowed the comparison of the players' perspective to the insights of game developers, to determine to what extent these two perspectives align and differ and to gain a well-rounded view of the issues that impact on player enjoyment when interacting in game worlds.

1.6.2 Part II: Designing, Implementing and Testing the EmerGEnT System

The aim of the second section of the research was to design, implement and test an integrated model for game worlds with cellular automata as a foundation. The purpose of this model was to facilitate emergent behaviour, gameplay and player enjoyment. This section consisted of three studies that involved developing the environment, game objects and agents of the EmerGEnT system.

The first step involved developing a game environment system based on cellular automata that models fluid flow, heat and pressure (see Chapter 4). The aims were to determine the suitability of cellular automata for modelling game systems and to determine the extent to which cellular automata can facilitate emergent behaviour and gameplay. Second, game objects (e.g. buildings) with a property-based design were integrated into the environment to investigate how the affordances of different objects impact on the design of an emergent game world and how game objects can facilitate emergent behaviour and gameplay (see Chapter 5). Finally, agents (e.g. villagers) were integrated into the environment to determine to what extent an active environment can promote reactive agent behaviour and to what extent reactive agents can facilitate emergent behaviour and gameplay (see Chapter 6).

1.6.3 Part III: Evaluating the Facilitation of Player Enjoyment in the EmerGEnT System

The third and final section of the research consisted of evaluating the EmerGEnT system in terms of player enjoyment. A model of player enjoyment in games, GameFlow, was used to evaluate how, and to what extent, the EmerGEnT system can facilitate player enjoyment in games (see Chapter 7). GameFlow is an extension of flow (Csikszentmihalyi, 1990), an accepted model of enjoyment. Flow has been applied extensively by researchers to assess enjoyment in a wide variety of domains, including problem solving (Vass, Carroll & Shaffer, 2002), ecommerce websites (Jennings, 2000), an interactive music environment (Pachet & Addressi, 2004) and information systems (Artz, 1996).

In a study independent of this thesis, flow was adapted to specifically model player enjoyment in games, by drawing on games usability and user-experience literature to identify how the criteria of flow are manifested in games (see Sweetser & Wyeth, in press, on the CD). The EmerGEnT system was evaluated against the GameFlow criteria by qualitatively evaluating how and how much each criterion is facilitated by the EmerGEnT system.

1.7 Contribution

The research presented in this thesis contributes to the understanding of emergence as an approach to game design. The major achievements are:

- **Proof of concept** of the validity and value of using an emergent approach to developing game worlds

- **Cellular automata** – the first investigation of using cellular automata to model game worlds

- **Design** for an emergent game world that includes:
  - **environmental effects** – cellular automata model rain, fluid flow, heat, fire, pressure and explosions
  - **property-based objects** – interact with environment via properties for structure and composition
  - **reactive agents** – react to and interact with the environment via properties and influence maps for decision-making

- **Player Enjoyment** – an understanding of player enjoyment in:
  - **current game worlds** – the first empirical investigation of the affect of current game worlds on player enjoyment
  - **emergent game worlds** – how emergent game worlds affect player enjoyment

2

Scripting versus Emergence:

Contrasting Approaches to Developing Game Worlds

The approach that is used to develop game worlds holds considerations for game developers and players. The current approach to developing game worlds is a scripted approach. Scripting involves a specific, low-level, entities-based approach to developing game worlds. The considerations of the scripted approach for game players include inconsistencies in the game world, unintuitive interactions, a slow learning curve, limited freedom for the player and no possibility of emergent gameplay. For game developers, developing scripted game worlds involves substantial effort in planning, implementing and testing, difficulties in extending and modifying, and issues with quality assurance due to inconsistencies. However, the current scripted approach does afford developers full creative control, no uncertainty in how the game system will behave and ease of giving feedback and direction to players. The current proliferation of the scripted approach is partly due to these reasons and partly due to the widespread use of scripted and static software techniques, such as scripting and finite state machines.

The proposed alternative to the current scripted approach is an emergent approach to developing game worlds. Emergence involves a top-down, systems-based approach to developing game worlds. Emergence has been integrated to a limited degree in previous games to allow emergent gameplay or emergent narrative. Considerations of an emergent approach for game developers include significant planning and tuning in development, a loss of creative control, difficulties in giving feedback and direction to players and uncertainty in how the game will respond to the player. However, emergent systems are easier to modify and extend and the uncertainty gives the possibility for emergent gameplay. Emergent systems can potentially improve player experience as they are inherently consistent, interactions can be more intuitive, the players' learning curve can be reduced, and emergent systems allow far more freedom for players and the possibility of emergent gameplay. Techniques that can potentially be used to facilitate emergence in games include flocking, neural networks, cellular automata and evolutionary algorithms.

2.1 Current Approach to Game Design

The majority of current games are developed with a scripted approach, which involves the game developer predefining specific paths and interactions that the player will take throughout the game. Scripted game design is the creation of gameplay out of the ideas of a particular designer, as needed for a specific, localised occurrence in the game. Scripted design involves limited awareness of global game patterns and relies on a given designer's ideas of what is consistent and fun (Smith, 2002). The environments, objects and agents in these games are limited to the narrow and static behaviour that the developer has predefined. As a result, the players' possible interactions with these game elements and resulting gameplay is confined, inflexible and lifeless. These scripted systems have also been referred to as "emulations" (Church, 2002) and "specific" systems (Smith, 2002).

2.1.1 Issues for Players

Some of the issues of player enjoyment that need to be considered when designing game worlds have been identified and discussed by game developers. These issues include the ability of the game to uphold the player's suspension of disbelief, consistency in the game world, the intuitiveness of the environment, player expectation and learning, and how well the game facilitates player expression and emergent gameplay. Each of these issues is discussed in this section with respect to current scripted game systems.

2.1.1.1 Consistency

Game worlds that behave consistently and in ways that the player understands enable the player to become immersed in the environment and suspend disbelief (Smith, 2001). Conversely, inconsistencies in games remind that player that it is just a game, breaking their suspension of disbelief. For example, if the player becomes stuck in a wall when adventuring in a dungeon (Hecker, 2000) or a monster attacks them through the wall then inconsistencies occur with the fantasy that the game has created. Similarly, if a boom microphone appears in an emotional scene in a movie, the immersion the viewer feels – their suspension of disbelief – is instantly broken (Hecker, 2000). The viewer of the movie or the player of the game is transported back to the real world, reminded and disappointed that their experience was fake. Scripted game systems inherently break the player's immersion, as their specific interactions and situations give rise to many inconsistencies.

2.1.1.2 Intuitiveness and Learning

Another important aspect of player interaction with the game environment is intuitiveness and player expectation. A casual game player or a non-game player is likely to be baffled by the physics of the game world (Smith, 2001). In the game world, only "explosive" barrels burn, some pieces of light furniture cannot be moved, the player's character might not be able to climb onto a desk and sometimes glass does not break. In order to be able to play computer games, it is necessary to relearn the physics of the world like a child (Smith, 2001). These types of problems arise in scripted games because the possible interactions that the player can have with the game environment are not intuitive and they do not meet player expectation.

The intuitiveness of interactions in game worlds can be partly attributed to how the interactions correspond to interactions with the same objects in the real world. Game worlds are populated with objects that are visually similar to objects that we use every day, but that are functionally different. Not only can these interactions be counter-intuitive for the player, but they can often confuse and frustrate the player (Hecker, 2000). It is natural for a player to expect that they will be able to pick up a phone, kick a chair and break a window, as they have learned these actions are possible throughout their whole life. However, in scripted systems, these actions are only possible if the developer has specifically coded them for each game object. Consequently, it is likely that many intuitive and seemingly logical actions will not be possible.

2.1.1.3 Emergent Gameplay and Player Expression

The final issue identified in the game development literature is the degree of freedom of player expression and the possibility of emergent gameplay that is supported by the game system. In scripted games, the designers manually define a number of outcomes or interactions and allow the player to pick one. The result is a handful of canned solutions to each particular problem (Smith, 2001), which makes the game linear (i.e. only one path through the game). The player is given a choice of a small number of static courses of action to take, which have been predefined by the game designers. The game is played in the exact way it was specified, which might not accommodate player creativity (Church, 2002).

2.1.2 Issues for Developers

There are five central issues in the game development literature that are important to consider when designing game systems. These issues are (1) effort in designing, implementing and testing, (2) effort in modifying and extending, (3) level of creative control for game developers, (4) uncertainty and quality assurance, and (5) ease of feedback and direction to players. Each of these issues is described in this section and discussed with respect to the current scripted approach to game development.

2.1.2.1 Effort in Designing, Implementing and Testing

In developing scripted games, specific interactions need to be planned by the game designers (Church, 2002) and the possible courses of action that the players can take need to be manually setup by the developers (Smith, 2001). Scripted systems require a “look and feel” approach to the placement of units, weapons, tools, resources, and specific puzzles or scripted sequences. Scripted games require a great deal of time and effort by the designers, as well as vigilant manual effort to ensure consistency in the game world (Smith, 2002).

2.1.2.2 Effort in Modifying and Extending

Scripted systems scale poorly and do not lend themselves to extensibility (Church, 2002). The properties and parameters of objects in scripted systems are different for each instance. Also, objects must have explicit relationships with other game elements for interactions to occur. For example, for a bullet from a gun to break a window, there needs to be a direct relationship between the gun entity and the window entity (Smith, 2001). The gun class would need to contain code listing all the things it could affect. Consequently, any changes that need to be made to the system require revision of any aspect of the game that is affected by the change (Church, 2002). Also, fixing bugs in the system requires each instance of a game element to be visited and reconfigured manually (Smith, 2002).

2.1.2.3 Level of Creative Control

As game developers manually plan and set up specific situations, interactions and events in scripted games, the game designers have full creative control over the game. The designers are empowered to create a specific narrative flow for the game, by defining the order and nature of the players' actions and encounters in the game.

2.1.2.4 Uncertainty and Quality Assurance

Similarly, nothing occurs in the game that was not intended or planned by the game developer. Consequently, there is no uncertainty or unexpected events in the game. The player plays the game in the exact way that the developer had intended. However, due to the inconsistencies that can exist in scripted games, quality assurance requires extensive testing of each game element, interaction and event. The scripted approach is effective for developing simple systems or specific complex behaviour, but can be difficult to manage on a larger scale.

2.1.2.5 Ease of Feedback and Direction

As with creative control, giving feedback and direction to players is simple in scripted systems as the developer knows when and how the player will interact with various game elements. As the desired outcome is known, it is straightforward to give players feedback on their success at performing actions or fulfilling goals.

2.1.3 Techniques for Scripting Games

The techniques that are used to implement a game's environment, objects and agents define whether the system will be static and scripted or dynamic and emergent. There are two main techniques that are used for implementing scripted game systems: scripting and finite state machines. Almost every commercial computer game uses scripting or state machines for some, if not all, of the game system. These techniques require everything to be built into the system during development, which means that the system can only behave as it has been told to behave with no room for adaptation or unexpected behaviour.

2.1.3.1 Finite State Machines

A finite state machine (FSM) is a device that consists of a set of states, a set of input events, a set of output events and a state transition function, which takes the current state and an input event and returns the new set of output events and the next state. The purpose of an FSM is to divide a game object's behaviour into logical states so that the object has one state for each different type of behaviour it exhibits (Rabin, 2000).

FSMs are by far the most popular technique in modern games, as they are simple to program, easy to understand and debug, and general enough to be used for any problem (Rabin, 2002). FSMs are amongst the simplest computational devices and provide a large amount of power relative to their complexity. Consequently, FSMs are ideal for the conditions of game development, which involves limited computational resources, as well as limited development and testing time. Some problems with using FSMs are that they tend to be poorly structured with poor scaling, so that they increase in size uncontrollably as the development cycle progresses. As a result, FSM maintenance can be very difficult and game FSMs that are not well planned and structured can grow out-of-hand quickly.

2.1.3.2 Scripting

Scripting languages are designed to simplify some set of tasks for a game and hide many complicated aspects (Berger, 2002), thus allowing non-programmers, such as designers and artists, to write script for the game. Scripting languages for games, such as Quake's QuakeC or Unreal's UnrealScript, allow game code to be programmed in a high-level, English-like language (LaMothe, 1999), which is used to control the game engine from the outside. The scope of a scripting language can vary significantly depending on the problems it is designed to solve, ranging from a simple configuration script to a full-blown runtime interpreted language (Poiker, 2002).

Scripting languages are ideal for games as they are suitable for non-programmers, such as designers, artists and end users. During development, the designers use scripting to implement stories (Poiker, 2002), while artists use scripting to automate repetitious tasks, do things that the computer can do better than humans and add new functionality (Stripinis, 2001). After the game is shipped, "mod" groups and hobbyists write scripts if the scripting system has been exposed to the public (Poiker, 2002). Also, scripting languages are generally separate from the game's data structures and codebase and thus provide a safe environment for non-programmers and end users to make changes to the game, so that bugs in the script will not cause the game to crash. However, as with FSMs, scripting languages are deterministic and they require the game developer to hard-code character behaviour and game scenarios. Therefore, the developer must anticipate and hard-code each of the player's possible situations, making the game predictable and linear.

2.2 Emergence as an Alternative Approach

As discussed in the introduction (Chapter 1), there is an opportunity to develop a new approach to designing game worlds, which allows more flexible, open gameplay. A possible alternative to the current scripted approach to game design is to design general, rule-based systems that allow the creation of gameplay out of combinations of existing game elements with globally defined, consistent characteristics and behaviour. This approach is an emergent approach to game design and is also referred to as "simulation" (Church, 2002) and "systemic" system design (Smith, 2002) in the game development literature. An emergent approach to game design would involve a globally designed game system that provides rules and boundaries for player interactions, rather than prescripted paths.

2.2.1 Complex Systems

"Complex systems" is the field of research that includes chaos theory, artificial life, evolutionary computation and genetic algorithms. Complex systems are systems in which individual, simple entities interact to give rise to overall complex behaviour. There are several attributes that are common to complex systems, including short-range relationships, non-linear relationships, feedback loops, openness, no control component, nesting, fuzzy boundaries and emergence (Holland, 1998; Johnson, 2001).

Ant colonies are examples of complex systems (Johnson, 2001). The queen does not give direct orders. Instead, each ant reacts to stimuli in the form of chemical scent from larvae, other ants, intruders, food, waste, and leaves behind a chemical trail that provides stimulus to other ants. Each ant is an autonomous unit that reacts depending on its local environment and its genetically encoded rules. Despite the lack of direct organisation, ants demonstrate complex behaviour (e.g. building complex colonies) and can solve geometric problems (e.g. navigating around obstacles).

2.2.2 Emergence

Complex systems are distinguished from systems that are merely "complicated" by the possibility of emergence. Emergence is the process of deriving some new and coherent structures, patterns and properties in a complex system (Holland, 1998). Emergent phenomena occur due to the pattern of interactions between the elements of a system over time. They are often unexpected, nontrivial results of simple interactions of simple components. Emergent behaviour occurs when a number of simple entities operate in an environment, forming more complex behaviour as a collective (Johnson, 2001). Emergence arises when a complex system reaches a combined threshold of diversity, organisation and connectivity. It is not a property of any single entity and cannot be predicted or reduced from the behaviour of the entities individually. Emergent systems are more than the sum of their parts as the entities cannot simply coexist, they must interact.

2.2.3 Emergence in Games

Emergence in games can be separated into two potential approaches: emergence in gameplay and emergence in narrative. Both of these approaches have been included to some extent in previous games. Emergence in gameplay has been included in games through emergent game objects. Games such as The Sims and Half-Life 2 use property-based objects and rules for interactions with these objects to allow more open and emergent gameplay. Emergence in gameplay has also been created through many choices and possibilities. The real-time strategy game Age of Mythology allows the player to make many choices that affect the way they will be able to play the game. The player chooses their civilisation, major god and minor gods, which determines the units, upgrades, powers and mythology the player will be able to access.

Games have provided emergence in narrative via sheer size of game worlds. The Elder Scrolls III: Morrowind includes a massive world with numerous characters, objects, events and quests that gives the player great freedom and diversity. The player is still tied down to a central, linear storyline, but the many diversions and options along the way make each player's experience emergent and a product of their interactions with the game.

Despite the few examples given, the emergence that has been possible in previous games has been quite limited. Games could potentially allow the player to play the game in a way that was not designed or implemented by the game developer, but that works nonetheless. Emergent behaviour occurs as the player is able to use the basic elements that are provided by the game developer to create new gameplay (e.g. stories or strategies). Emergence in narrative could potentially involve generating a storyline based on the interactions between the game world, characters and objects. Emergence in gameplay could be developed to the extent of a fully emergent game world, in which there are no scripted paths, interactions or behaviours. The aim of this thesis is to investigate the potential and considerations of creating a fully emergent game world.

2.2.4 Issues for Game Developers

The same five issues need to be considered for designing emergent game systems as were discussed for designing scripted game systems. Emergent game systems have different considerations for (1) effort in designing, implementing and testing, (2) effort in modifying and extending, (3) level of creative control for game developers, (4) uncertainty and quality assurance, and (5) ease of feedback and direction to players than scripted systems. Each of these issues is described in this section and discussed with respect to the proposed emergent approach to game development.

2.2.4.1 Effort in Designing, Implementing and Testing

Creating emergent systems involves designing types of objects and interactions, rather than specific ones (Church, 2002), which can give rise to greater efficiency in development and testing. The properties and parameters reside at a higher level (Smith, 2002). Rather than having a specific gun able to break a specific window, there is an additional layer of abstraction that allows a gun to break anything made of glass. For example, the gun would project a bullet entity that has certain properties (e.g. ballistic damage, heat or electricity) and the glass is a stimulus-receiving entity (Smith, 2001). The system would have a set of rules about the relationship between the entities' general-case properties and when the bullet meets the glass, the game's object-property system looks up the effect of the bullet's properties on the glass entity. Therefore, the gun will work on any window (or any other stimulus-receiving object), rather than only the specified windows.

Emergent systems often require considerable initial effort in planning and building, as the rules and properties need to be defined in advance. Additionally, the system can require substantial tuning to get the rules and properties to function correctly. However, development can be more efficient as programmers can build tools that allow designers to "drop" objects into levels, with the properties and behaviour of the object already defined. Designers can also create new objects and attribute properties to the objects using the tools (Smith, 2002).

2.2.4.2 Effort in Modifying and Extending

Once an emergent system is built successfully, the design scales well (i.e. increases in size easily, maintaining robustness and manageability) and is easily extended (Church, 2002). Making changes to the system (e.g. fixing bugs) has the potential to be more efficient as changes can be made to object types, rather than each particular instance of an object than needs to be changed (Smith, 2002).

2.2.4.3 Level of Creative Control

The use of emergent systems in games could result in a possible loss of creative control for the game designer. Using an emergent system involves defining types of interactions and behaviours, which makes it is more difficult to set up specific narrative and sequences. Consequently, controlling the flow of game and telling a specific story is not as straightforward in an emergent system.

2.2.4.4 Uncertainty and Quality Assurance

Emergent systems also introduce uncertainty, which means that the game can behave in ways that the developers had not anticipated. Although this uncertainty can give rise to desirable, emergent gameplay, it can also be undesirable if the system allows behaviour that is detrimental to the game (Church, 2002). Extensive testing is required to ensure that the game does not allow detrimental behaviour. However, the emergent events can be too numerous or subtle for the development team to predict or detect during testing (Smith, 2002).

2.2.4.5 Ease of Feedback and Direction

Players have a greater need for feedback on the outcome and success of their actions in emergent systems, as the openness of the game world gives rise to more possibilities for action (Smith, 2001). Consequently, the players need more feedback to know that they are on the right track and that their actions are successful.

2.2.5 Issues for Game Players

Game developers and researchers have identified and discussed some of the issues of player interaction in game worlds and how they relate to emergent games. The issues discussed in the game development literature include the ability of the game to uphold the player's suspension of disbelief, consistency in the game world, the intuitiveness of the environment, player expectation and learning, and how well the game facilitates player expression and emergent gameplay. Each of these issues is discussed in this section with respect to emergent game systems.

2.2.5.1 Consistency

Emergent games have the potential to be used to create more consistent game worlds (Smith, 2001). The game worlds in emergent systems are inherently consistent as the rules and properties are defined globally, for types of objects, rather than locally for each specific object. For example, the player knows that bullets affect everything that is damageable, such as windows, vases and chairs, rather than some windows and no vases. Furthermore, the player can deduce that if they can move objects and put objects on top of one another then they can stack crates. Games that obey a consistent set of physical laws allow the player to stay immersed in the game, sparing them from unpleasant surprises (Hecker, 2000).

2.2.5.2 Intuitiveness and Learning

Game worlds that work in a way that reflect players' lifelong experiences (in the real world) are more intuitive and easier to understand for the average person, even in fantasy realms and alien dimensions (Smith, 2001). Emergent game systems are more likely to be intuitive to the average person as it is easier to create objects that behave and interact in more natural ways, with a wider variety of interactions. The objects in emergent systems are not limited to specific interactions that have been hard-coded. Instead, they interact in ways that are conducive to their properties and rules for interaction.

An important benefit of making game worlds more intuitive is that they become easier to learn. The player is more likely to develop an intuitive understanding of the game elements if they are consistent with real world elements (Smith, 2002). For example, if fire in the game behaves like fire in the real world then the player will have an inherent understanding of how the fire works, without needing to be retaught the rules of fire within the game (Smith, 2001). With the use of intuitive game elements, the player is more likely to understand the elements, even when encountering them for the first time. As a result, the learning curve of the player is substantially decreased, which means that the player spends less time learning and more time playing the game (Smith, 2002).

2.2.5.3 Emergent Gameplay and Player Expression

Emergent systems define global possibilities for actions the player can perform, which can be applied in more open ways in specific situations. Players have more freedom to express their creativity and gameplay can occur that wasn't anticipated by the designers. Emergent gameplay allows players to solve game problems by using strategies that were not envisaged by the designers (Smith, 2001; Garneau, 2002). Emergent gameplay occurs when a player's actions result in a second order of consequence that the development team did not predict and the game behaves in a rational but unplanned way (McLean, 2002; Smith, 2002). For example, in the game Deus Ex, players used proximity mines to create ladders up walls to climb off the map, a possibility that was not foreseen by the developers.

Emergent game systems empower the player by putting them centre stage (Church, 2002), giving them freedom to experiment, greater control, a sense of agency, and less of a feeling of uncovering a path set for them by the designers (Smith, 2002). Consequently, the game can be more satisfying and interesting for the player. Game worlds that are not full of prescripted one-to-one interactions are empowering to the player as the gameplay becomes largely about exploring the possibility space and the game experiences become richer (McLean, 2002). Emergent games also have high replayability as each time the player plays the game they make different decisions, which change the game as a whole and result in different possibilities for action (Garneau, 2002).

The major difference between scripted and emergent games is that emergent games focus on what the player wants to do, whereas scripted games focus on what the designer wants the player to do (Smith, 2001). However, it is important to realise that emergence alone isn't a game (Church, 2002). Emergence in games needs to be used to improve gameplay, not simply for its own sake.

2.2.6 Techniques for Emergent Games

Techniques that are given the boundaries for behaviour (rather than the script) or are able to grow and change have the potential to give rise to behaviour that may not have been foreseen (or expected) by the developers. Emergent behaviour occurs when simple, independent rules interact to give rise to behaviour that wasn't specifically programmed into the system (Rabin, 2004). Techniques that can be used to facilitate emergence come from complex systems, machine learning and artificial life. Some examples of emergent techniques that can or have been used in games are flocking, cellular automata, neural networks and evolutionary algorithms.

2.2.6.1 Flocking

Flocking is a technique for simulating natural behaviours for a group of entities, such as a herd of sheep or a school of fish (Reynolds, 1987). Flocking was devised as an alternative to scripting the paths of each entity individually, which was tedious, error-prone and hard to edit, especially for a large number of objects. Flocking assumes that a flock is simply the result of the interaction between the behaviours of individual birds. In flocking, the generic simulated flocking creatures are called boids. The basic flocking model consists of three simple steering behaviours, separation, alignment and cohesion, which describe how an individual boid manoeuvres based on the positions and velocities of its nearby flockmates. Separation enables the boid to steer to avoid crowding local flockmates, alignment allows the boid to steer towards the average heading of local flockmates and cohesion makes the boid steer to move toward the average position of local flockmates (Reynolds, 1987). Each member in the flock revaluates its environment at every update cycle, which reduces the memory requirements and allows the flock to be purely reactive, responding to the changing environment in real time.

Flocking has been successfully used in various commercial games, including Half-life, Unreal, Theme Hospital and Enemy Nations, as it provides a powerful tool for unit movement (Johnson & Wiles, 2001) and for creating realistic environments the player can explore (Woodcock, 2003). It is a relatively simple algorithm and only composes a small component of a game engine. However, flocking makes a significant contribution to games by making an attack by a group of monsters or marines realistic and coordinated. It therefore adds to the suspension of disbelief of the game and is ideal for real-time strategy or first-person shooter games that include flocks, swarms or herds.

2.2.6.2 Cellular Automata

Cellular automata (CA) are widely-used techniques in the field of complex systems, which studies agents and their interactions. A traditional CA is a spatial, discrete time model in which space is represented as a uniform grid (for a comprehensive review see Bar-Yam, 1997). Each cell in the grid has a state, typically chosen from a finite set. In a CA, time advances in discrete steps. At each time step, each cell changes its state according to a set of rules that represent the allowable physics of the model. The new state of a cell is a function of the previous state of the cell and the states of its neighbouring cells. A CA can be represented in one, two or more dimensions. A one-dimensional CA consists of a single line of cells, where the new state of each cell depends on its own state and the state of the cells to its left and right. In a two-dimensional CA, each cell can have four or eight neighbours, depending on whether cells diagonally adjacent to a cell are considered neighbours. CA have been proposed as a solution to the static environments that are prevalent in current computer games (Forsyth, 2002). The use of CA could lead to more dynamic and realistic behaviour of many game elements that are currently scripted, such as fire, water, explosions, smoke and heat.

A variation of CA, influence mapping, is a method for representing the distribution of power within a game world in a two-dimensional grid (Rabin, 2004). Influence maps are commonly used for strategic assessment and decision-making in games (Sweetser, 2004a[1]), but were also used in the game SimCity to model the influence of various social entities, such as police and fire stations around the city (Rabin, 2004).

2.2.6.3 Neural Networks

Neural networks are machine learning techniques inspired by the human brain. Neural networks are comprised of artificial neurons, called units, and artificial synapses, called weights. In a neural network, knowledge is acquired from the environment through a learning process and stored in the network's connection weights (Haykin, 1994). The network learns from a training set of data by iteratively adjusting its weights until each weight correctly reflects the relative influence that each unit has on the output. After training is complete, the network is ready to be used for prediction, classification or decision-making.

Some of the considerations when developing neural networks for games include which variables from the game world will be used as input, the design of the structure of the network, what type of learning will be used, and whether learning will be conducted in-game or during development (Sweetser, 2004b[1]). If the neural network is allowed to learn during the game then it will be able to dynamically build up a set of experiences and adapt to new situations and the human player as the game progresses. Alternatively, training the neural network during development will produce a network that will behave within expectations and require minimal resources. Overall, advantages of neural networks include their flexibility for different applications, their ability to adapt when trained in-game and the efficiency of their evaluation once trained. However, neural networks can also consume significant resources when training, can require substantial tuning to produce optimal results and can learn unpredictable or inaccurate information if trained incorrectly.

2.2.6.4 Evolutionary Algorithms

An evolutionary algorithm (EA) is a technique for optimization and search, which evolves a solution to a problem in a similar way to natural selection and evolution (Mitchell, 1998). An EA includes a population of possible solutions to a problem,

[1] A study independent of the research reported in this thesis.

referred to as chromosomes, as well as processes that evaluate each chromosome's fitness and select which chromosomes will become parents. The chromosomes that are selected to be parents take part in a process similar to reproduction in which they generate new offspring by exchanging genes. The new offspring also have a chance that they will mutate, similar to natural mutation. As the cycle continues over time, more effective solutions to the problem are evolved.

Considerations that need to be made when designing an EA for a game include the many parameters that need to be tuned, such as choice of a suitable representation, population size, number of generations, choice of a fitness function and selection function, and mutation and crossover parameters (Sweetser, 2004c[2]). There are many advantages to using an EA, as they are a robust search method for large, complex or poorly-understood search spaces and non-linear problems. An EA is useful and efficient when domain knowledge is limited or expert knowledge is difficult to encode as they require little information to search effectively. Also, they are useful when traditional mathematical and search methods fail. On the down side, an EA is computationally expensive and requires substantial tuning to work effectively. In general, the more resources they can access the better, with larger populations and generations giving better solutions. However, an EA can be used offline, either during development or between games on the user's computer, rather than consuming valuable in-game resources.

2.3 Scripting-Emergence Continuum

The two extreme approaches to game design discussed in this chapter ranged from hand-crafted, hard-coded, scripted environments to rule-based, general, emergent environments. An emergent approach to game design is significantly different from the current scripted approach to game design, in terms of modelling techniques, as well as the implications for developers and players. However, the two approaches are not mutually exclusive. Rather, scripting and emergence can be seen as two extremes of a continuum (Church, 2002; Smith, 2002).

[2] A study independent of the research reported in this thesis.

Both extremes hold benefits and drawbacks for game developers, as well as consequences for the game players. At the scripted end of the continuum, the developers must hand-craft, implement and test every aspect of the game individually but are able to keep full creative control and rest assured that the game won't break after release. With the scripted extreme, the players are often locked into playing the game in a predefined way, unable to express their own creativity and may encounter inconsistencies in the game world. At the other end of the continuum are emergent game worlds that simply contain general rules for how the environment, objects and agents will interact, and the specific behaviours and events emerge from the interactions of the general rules. However, emergence is a frightening prospect for developers, who cannot be sure how the game will actually behave after it is released, and is a sandbox type environment even a game? The emergent extreme does, however, hold the potential for players to express their own creativity and for intuitive and consistent interactions to take place in the game world.

It seems that the future of game development lies somewhere between these two extremes; that there needs to be the right combination of scripted, narrated gameplay and freedom to interact within the world. There needs to be some way to define the boundaries of action, moving the story forwards, but still letting the player do their own thing along the way. We suggest that a game world that facilitates emergent interactions, based on a technique such as cellular automata, can be used in conjunction with other more conventional techniques for gameplay, such as scripting, to allow the player sandbox-style interaction within the boundaries of a predefined story and game objectives.

Part I

Identifying the Player-Centred Issues of Interacting in Game Worlds

3

Player-Centred Game Worlds:

Assessing Player Opinions, Experiences and Issues

Acquiring the player's perspective on game design issues is central to enhancing the gaming experience, by understanding, and ultimately meeting, the desires and expectations of the player. Although many game developers gather player feedback in some form, there is limited published literature on game design in terms of the aspects of game environments that affect player enjoyment. Furthermore, the majority of the literature (discussed in Chapter 2) is based on the personal experiences and thoughts of game developers (e.g. Smith, 2002; Church, 2002), rather than systematic and rigorous studies with players.

The game development literature focuses on three major themes: consistency in the game world, the intuitiveness of the environment and how well the game facilitates player expression and emergent gameplay. Consistency is crucial for keeping players immersed in the game world (Hecker, 2000). If the game world seems to behave consistently and in ways that the player understands then the player has less difficulty immersing themselves in the environment and suspending disbelief (Smith, 2001). The second theme, intuitiveness, suggests that interactions with the game environment and objects in the game environment should be intuitive and meet player expectation. An important benefit of making game worlds more intuitive is that they become easier to learn. The third theme identified in the literature is freedom of player expression and the possibility of emergent gameplay. Defining a limited number of outcomes or interactions for the player restricts their freedom in interacting and removes the possibility of emergent gameplay (Smith, 2001).

Although a great deal can be learned from the thoughts and experiences of game developers, there is also a need to assess the thoughts and opinions of game players regarding the factors that affect their enjoyment of games. Previous empirical studies have provided insight into the players' perspective on non-player characters in games (see Sweetser, Johnson, Sweetser & Wiles, 2003[3], on the CD; Drennan, Viller & Wyeth, 2004). However, there is no published work on empirical studies conducted by game developers or researchers to ascertain player perspective on interacting in game environments. Therefore, the aim of the player-centred studies reported in this chapter was to investigate the aspects of game environments that affect player enjoyment from the players' perspective in order to support or contest the developers' views. A focus group was conducted with experienced game-players to identify the issues that they felt most affected their enjoyment of a game. The focus group was followed up by a questionnaire, which aimed to assess how the issues identified in the focus group vary in importance across players with different game-playing experience and game-type preference.

3.1 Defining the Player-Centred Issues

The goal of the focus group study was to collect the opinions and experiences of a group of experienced game players with respect to the issues that impact upon their enjoyment of game environments. The focus group consisted of four experienced game players, consisting of one female and three males aged between 21 and 25. Each member of the group considered him or herself to be an experienced game player and reported playing games on a daily basis, with a minimum of five years experience playing games. The focus group involved several general points of discussion, but the group was mostly encouraged to discuss their experiences. The guiding questions related to general experience with interacting in games, experience with specific games, consistency, gameplay, immersion and suspension of disbelief, physics and intuitiveness of interactions. An audio recording was taken during the focus group,

[3] A study independent of the research reported in this thesis.

which was later transcribed and analysed using grounded theory (Glaser, 1998). The analysis gave rise to five major themes: consistency, immersion and suspension of disbelief, freedom of player expression, intuitiveness and physics.

3.1.1 Consistency

A strong theme arising throughout the focus group was the importance of consistency in games. Participants felt that it is highly important for objects that look the same to act the same. For example, one member was frustrated with “glass windows that break sometimes but don’t break other times.” Similar problems were identified for crates, barrels, lanterns and mirrors. Inconsistencies can cause difficulties for the player in learning the rules of the game, which appear to be constantly changing. If the player learns in one instance to kick a barrel to break it and the next time they kick a barrel it doesn’t break then they become confused and even frustrated with the game.

On the other end of the scale, it is important for objects that have different behaviour to look different, signalling to the player that a different kind of interaction is possible. For example, “some games signal actions by having different coloured walls for bits of wall you can kick out, for example Bloodrayne” and referring to the game Dungeon Siege “there were certain walls that looked a bit different, but you knew that you could shoot that wall out”. However, it was also expressed that the visual difference shouldn’t be in the form of something unrealistic, such as a big red circle around the section of wall. Rather, it should be a subtle, realistic difference that the player can detect, such as a “worn-looking part of the wall” that might be more fragile.

3.1.2 Immersion and Suspension of Disbelief

Another major theme of the discussion was immersion and suspension of disbelief. The group agreed that audio is very important for keeping the player immersed in the game, in terms of a “powerful and moving soundtrack”, as well as sound effects. The group thought that a game is immersive if it can cause an emotional response, such as fear or happiness. The group discussed how sounds can be used to build up suspense, such as in a horror movie when “you know that something is creeping up on you, to the point that you’re afraid and shifting in your seat”. Furthermore, audio was highlighted as being important for “drawing you into the game, but inconsistent graphics can quickly knock you back out again”. The graphics don’t need to be spectacular, but they do need to be consistent and ensure “nothing catches your eye as being wrong or out of place”. A good introduction and a strong narrative were also identified as being important for immersion. The introduction gives the player the storyline and background, tells them who their character is and what is going on. The player then feels like they are “part of the story and they want to find out more”. As they play the game, the player is given more of the storyline, “similar to reading a book, except that you need to complete certain tasks” to be rewarded with more of the story.

3.1.3 Freedom of Player Expression

Another theme that arose was player expression, which is the freedom that the player has in expressing their creativity and intentions by playing the game in the way that they want, not the way that the designer had intended it to be played. The group discussed linearity in games and agreed that they are often forced to solve problems and perform tasks the way the designer had imagined, which can rely on the player using trial and error. For example, many “quests aren’t even quests, they’re completely linear, you’ve been told exactly what to do, you just have to go pick this thing up and come back, you should be able to go out and do the quest your own way”. One member said that it becomes “more like trial and error than playing and it’s not as fun as looking around at a collection of objects and working out how to use them to solve the problem”. It was also considered to be important that the player has a range of interactions that can be performed with the environment and game objects and that each game should have some kind of new and unique interaction.

3.1.4 Intuitiveness

The group reflected on many experiences in games where their interactions with the environment had not been intuitive. A major source of frustration came from objects in games that were simply scenery and hence could not be used or affected. For example, furniture that cannot be moved as it seems to be bolted to the floor or “a flimsy little plastic chair that can be shot with a shotgun and it’s resilient enough to take that and not be damaged”. Unintuitive interactions can also cause problems for gameplay, when the player cannot use objects in the way that they would expect, in order to complete a quest or solve a problem. The group discussed the unintuitive nature of problem solving in some games. The group found that the way the designers intend the problem to be solved is often not intuitive for the player and that they “resort to trial and error”. The group suggested that “if it takes 10 hours to find a switch or the player needs to go to the internet to get a walkthrough then there is a serious problem with the game”. Therefore, it is important to conduct extensive testing to ensure that the players’ expectations are met and that they will be able to solve the problems in a reasonable time frame, rather than assuming the designer’s intentions will be easily determined.

3.1.5 Physics

The group discussed their expectations of physics in games and reflected on their good and bad experiences. There was consensus that gravity in games is important for actions such as jumping, falling, taking falling damage, trajectory when launching rockets and so on. Modelling gravity can give rise to realistic effects such as bouncing grenades around corners, falling off a platform or rolling down a hill when shot. More importantly, the “gravity needs to behave consistently, even if it’s not entirely realistic”, such as in first person shooter games like Unreal Tournament and Quake, where the game may be in low gravity mode. Momentum was also identified as an important attribute of physics that needs to be modelled in games, especially in space simulations and first person shooters. For example, if the player shoots the enemy or is shot by the enemy then “being pushed backwards is natural”.

Another important aspect of physics concerns fire and explosions. Flammable game objects should burn and ignite when affected by a flamethrower or incendiary grenade. Also, when a flash grenade explodes next to a character it should adversely affect their sight and hearing, or when an explosion occurs the player “should be able to jump into a pool of water to be protected” from damage. Water was also identified as a substance that needs to be modelled more accurately in games. For example, there was a considerable discussion about how most weapons shouldn’t work under water, especially flamethrowers and fire-based weapons. Other attributes of water that the group decided were important were the effects of the flow and currents of the water, as well as visual effects such as ripples.

In summary, consistent physics are important in games to ensure the game reacts in the way that the player expects, to allow the player to perform actions in an intuitive manner and to keep the player immersed in the game world. Currently, players expect gravity, momentum and the basic laws of physics to work in an intuitive way and they look forward to more interactive physics in water, fire and explosions.

3.1.6 Focus Group Summary

In summary, the focus group provided supporting evidence for the themes of consistency, freedom of expression and intuitiveness identified in the game design literature. First, players need consistency in games to be able to learn the rules of the game, to know when they can interact with game elements and to avoid frustration and confusion. Second, players want to be free to play games and solve problems in the way that they want, not the way the designer had intended. Third, counterintuitive interactions often result from game objects having no function or behaving in a way that conflicts with player expectation. Furthermore, the focus group provided insight into two new issues that affect player enjoyment in games, immersion and physics. Immersive games draw the player into the game and affect their senses and emotions through elements such as audio and narrative. Finally, consistent physics are important in games to ensure the physical elements of the game world, such as gravity, momentum, fire and water, behave in the way that the player expects, to allow the player to perform actions in an intuitive manner and to keep the player immersed in the game world.

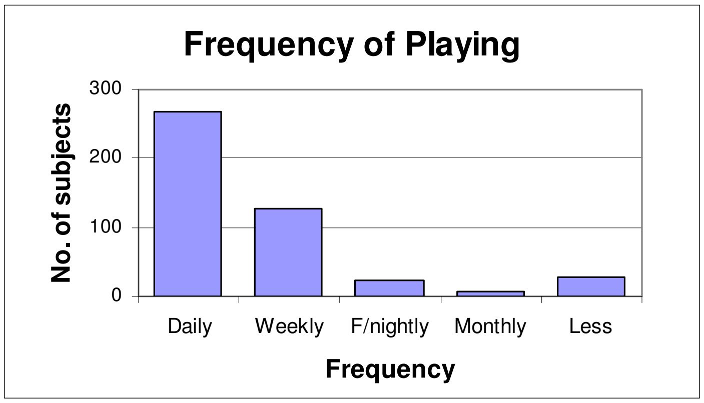
3.2 Investigating the Player-Centred Issues

The results obtained from the focus group were used as a basis for constructing a questionnaire (see Appendix A) that aimed to further investigate the issues of consistency, immersion, freedom of expression, intuitiveness and physics in game environments. Whereas the focus group provided in-depth insight into the opinions and experiences of a small group of experienced game players, the questionnaire was designed to provide a survey of the opinions of a large, diverse group of game players. There were two main aims of the questionnaire study. First, to determine how the different issues defined in the focus group affect the enjoyment of people who play different types of computer games, such as role-playing games or first-person shooter games. The second aim was to determine how these issues affect the enjoyment of people with different levels of experience playing computer games. One question of particular interest was whether intuitive interactions with game environments and objects have a greater effect on the enjoyment of people with less game-playing experience. People who have less experience playing games are not well-versed in the "rules" of game worlds and it is therefore likely that unintuitive interactions have a greater effect on their enjoyment.

3.2.1 Method

PARTICIPANTS

The questionnaire was administered online (http://www.itee.uq.edu.au/~penny/questionnaire.htm) and invitations to participate were emailed to university staff and students and posted on several online game forums (http://vnboards.ign.com/). The boards posted to included Half-Life 2, Neverwinter Nights, EverQuest, Morrowind and Eve Online. The questionnaire was completed by four hundred and fifty-five participants, of which 421 (92.5%) were male and 34 (7.5%) were female. Participants ranged in age from 11 to 56 years, M = 24.8 (SD = 6.91). The sample consisted mainly of frequent game-players, with 94% playing computer games at least monthly. The distribution of game-playing frequency and age are shown in Figure 3.1.



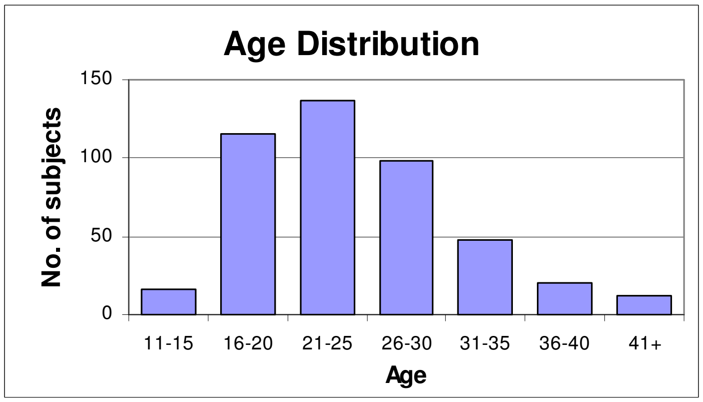


Figure 3.1. Participant Demographics. The majority of subjects (94%) played games at least monthly. Participants ranged in age from 11 to 56 years, with the majority in their late teens to late twenties

MEASURES

Independent. There were three between-subject variables, gender, game-type preference and experience. The participants selected their preferred type of game from a list of seven common game types, with the majority of participants nominating role-playing games (41%), first-person shooter (28%) or strategy games (16%) as their preferred game type. The other four game-types, simulation, action, racing and sports games accounted for twelve percent of the sample. The participants indicated their level of experience at playing computer games on a seven-point Likert scale, ranging from very inexperienced to very experienced, with the majority of participants rating themselves as experienced, M = 5.88 (SD = 1.37). The distribution of game-type preference and self-rated experience are shown in Figure 3.2.

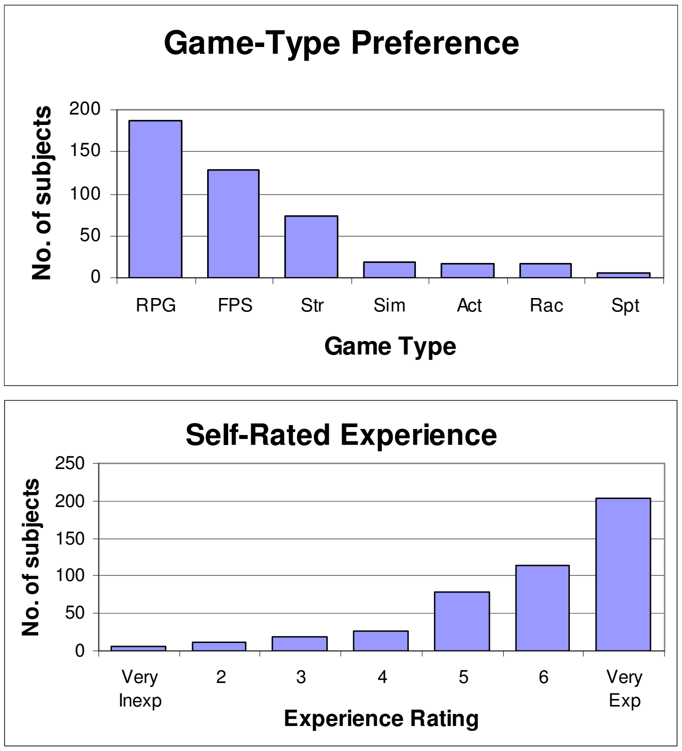


Figure 3.2. Independent Measures. Participants selected their preferred type of game from role-playing (RPG), first-person shooter (FPS), strategy (Str), simulation (Sim), action-adventure (Act), racing (Rac) and sport (Spt). The majority of participants rated themselves as very experienced

Dependent. The participants were required to complete 28 nine-point Likert scales to indicate the degree to which different aspects of games affect their enjoyment of their preferred type of game, where one indicated "much less enjoyable", five indicated "no effect" and nine indicated "much more enjoyable" (see Appendix A). All measures were assessed using multiple item scales and all negatively worded items were reverse-scored. Factor analysis via principal components was conducted to identify sets of variables that could be combined into scales. On the basis of eigenvalues greater than one criterion, a five factor solution was obtained accounting for 68.6% of the variance. Cronbach's coefficient alpha was used to assess the reliability of each scale. All scales were found to have acceptable reliabilities (.66 to .90). The five factors derived were physics, sound, narrative, intuitiveness and freedom of expression (see Table 3.1):

- Physics The physics scale consisted of eight items related to gravity, momentum, life-like graphics and the behaviour of water and fire.
- Sound The sound scale consisted of three items related to a game's soundtrack and sound effects.

- Narrative The narrative scale consisted of two items related to a game's introduction and storyline.
- Intuitiveness The intuitiveness scale consisted of three items related to consistent behaviour of objects, scenery and interaction with objects.
- Expression The freedom of expression scale consisted of two items related to the variety of ways of interacting with objects and having new and unique ways of interacting with objects.

Table 3.1. Descriptive data for dependent measures

Variable	*Mean*	*Standard Deviation*	*Possible Range*	*Actual Range*	*Reliability*
Sound	7.53	1.26	1 – 9	1 – 9	.85
Narrative	7.20	1.31	1 - 9	1.5 – 9	.68
Physics	7.07	1.09	1 - 9	3.13 – 9	.90
Expression	7.40	1.29	1 - 9	1 – 9	.79
Intuitiveness	6.31	1.06	1 - 9	2.67 - 9	.66

3.2.2 Results

Regression analyses were used to examine the main and interactive effects of gender, game-type preference and experience on each of the dependent measures (physics, sound, narrative, intuitiveness and expression). The main effect terms (gender, game-type preference and experience) were entered into the regression equation followed by the two-way interaction terms (gender-by-game-type preference, gender-by-experience and experience-by-game-type preference). The main effect terms, but not the two way-interaction terms, accounted for a significant increment of variance in physics (F(3,385) = 3.185, p < .05), sound (F(3,384 = 6.662, p < .05), expression (F(3,385) = 5.287, p < .05), and narrative (F(4,387) = 2.788, p < .05). Also, the main effect terms (F(3,385) = 4.395, p < .05) and the two-way interaction terms (F(6,382) = 3.810, p < .05) accounted for a significant increment of variance in intuitiveness.

It was found that physics (β = .095, t = 1.89, p = .06), sound (β = .117, t = 2.347, p < .05) and intuitiveness (β = .098, t = 1.937, p < .05) have a greater effect on enjoyment for people who prefer first-person shooter games than people who prefer other types of games. For people who prefer strategy games, it was found that narrative (β = .112, t = 1.668, p < .10) has a greater effect on enjoyment than for people who prefer other types of games. Also, it was found that of the people who prefer role-playing games, the effect that intuitiveness has on enjoyment increases as level of experience increases ($\beta = -.804$, $t = -2.841$, $p < .05$). Finally, it was found that the effect that sound ($\beta = .095$, $t = 1.853$, $p < .10$), freedom of expression ($\beta = .179$, $t = 3.448$, $p < .05$) and narrative ($\beta = .150$, $t = 2.829$, $p < .05$) have on player enjoyment increases as level of experience increases. Analyses of the main and interactive effects of gender, game-type preference and experience on the measures of physics, sound, narrative, intuitiveness and expression are presented in Table 3.2.

Table 3.2. Multiple Regression Analysis. Multiple regression analysis predicting Physics, Sound, Intuitiveness, Narrative and Expression from Gender, Game-Type Preference and Experience

		Physics			
Step	*Predictor*	*R2*	*R2 change*	*F change*	*β*
1	Gender	.024	.024	3.185*	.080
	Game-Type Preference				.095+
	Experience				.073
2	Gender x Game-Type	.032	.007	.974	.415
	Gender x Exp				.279
	Game-Type x Exp				-.331
		Sound			
Step	*Predictor*	*R2*	*R2 change*	*F change*	*β*
1	Gender	.049	.049	6.662***	.138**
	Game-Type Preference				.117*
	Experience				.095+
2	Gender x Game	.051	.002	.254	.617
	Gender x Exp				.823
	Game-Type x Exp				.052
		Intuitiveness			
Step	*Predictor*	*R2*	*R2 change*	*F change*	*β*
1	Gender	.033	.033	4.395**	.128*
	Game-Type Preference				.098*
	Experience				.052
2	Gender x Game-Type	.056	.023	3.152*	.519+
	Gender x Exp				.307
	Game-Type x Exp				-.804**
		Narrative			
Step	*Predictor*	*R2*	*R2 change*	*F change*	*β*
1	Gender	.021	.021	2.777*	.031
	Game-Type Preference				.112+
	Experience				.150**
2	Gender x Game-Type	.022	.001	.158	-.339
	Gender x Exp				-.010
	Game-Type x Exp				-.184
		Expression			
Step	*Predictor*	*R2*	*R2 change*	*F change*	*β*
1	Gender	.040	.040	5.287***	.044
	Game-Type Preference				.045
	Experience				.179***
2	Gender x Game-Type	.047	.007	.996	.370
	Gender x Exp				.367
	Game-Type x Exp				-.316

*** p < .001, ** p < .01, * p < .05, + p < .10

3.2.3 Discussion of Questionnaire

Effects of Game-Type Preference. The first aim of the questionnaire study was to investigate how game-type preference affects the aspects that make games more enjoyable for players. It was found that physics, sound and intuitiveness have a greater effect on enjoyment for people who prefer first-person shooter games than people who prefer other types of games. This finding can be attributed to the centrality of physics, sound and intuitiveness to first-person shooter games. First, physics is vital in first-person shooter games as typical behaviours include jumping, shooting and exploding, which need to be modelled with a certain degree of realism. Second, sound is important as it provides immediate feedback and information to the player about what is happening around them in the information-rich environment of first-person shooter games, which includes fast gameplay, different enemies, rapid movement and numerous interactions with objects and the environment. Also, sound aids in setting the mood of the game and provides an additional level of immersion, by making the player feel frightened or excited. Finally, intuitive interactions with objects are important in first-person shooter games as they require far more direct interaction with the environment (e.g. direct manipulation of objects and interaction with scenery) than any other type of game. Therefore, it is important that interactions in first-person shooter games are intuitive as the player will be carrying out a greater number of interactions with greater frequency.

For people who prefer strategy games, it was found that narrative has a greater effect on enjoyment than for people who prefer other types of games. It might be expected that narrative would have a greater effect on the enjoyment of people who prefer role-playing games. However, this finding could be due to the fact that every role-playing game includes narrative, but it is uncommon in strategy games. Therefore, the addition of well-placed narrative in strategy games has a significant effect in increasing player enjoyment, whereas narrative is commonplace and expected in role-playing games and therefore the players fail to recognize it as something that affects their enjoyment.

Effects of Game-Playing Experience. The second aim of the questionnaire study was to investigate how experience affects the aspects that make a game enjoyable. In particular, it was expected that the ability to interact intuitively with objects would have a greater effect on the enjoyment of people with less experience playing games. However, there was no evidence to support this theory. On the contrary, it was found that of the people who prefer role-playing games, the effect that intuitiveness has on enjoyment increases as level of experience increases. Maybe more experienced game players expect to be able to interact with game environments and objects in a particular way, learned from their prior experience playing games, and find it annoying when the game doesn't behave in the way they have learned to expect. On the other hand, the inexperienced players have no concept of how the game objects should behave, rather they only have the concepts they have learned in real life. Additionally, it was found that the effect that sound, freedom of expression and narrative have on player enjoyment increases as level of experience increases. As with intuitive interactions, perhaps the player's expectations of these three aspects are built up as they play more games. As a result, the more experienced game players know from experience that they have really enjoyed games with good sound and narrative. More experienced players find freedom of expression more enjoyable, as their experience means that they are more likely to want to experiment, try different strategies and try to play the game in their own way.

3.3 Discussion and Conclusions

The aim of the player-centred studies reported in this chapter was to determine the issues that have an impact on player enjoyment in a game environment from a player-centred perspective, by integrating the findings of a focus group and questionnaire with the insights of game developers. The focus group confirmed three of the themes identified in the game development literature. The first theme was consistency, which related to objects behaving in a consistent manner, enabling players to learn the rules of the game, to know when they can interact with game objects and to avoid frustration and confusion. The questionnaire, however, did not provide any further insight into consistency. The second theme from the literature that was supported by the focus group was freedom of expression, which refers to the ability of players to play the game and solve problems in their own way or a variety of ways, rather than the way the designer had intended. The questionnaire refined freedom of expression by showing that the effect freedom of expression has on player enjoyment increases with level of experience. The third theme that was confirmed by the focus group was intuitive interactions with the game environment, which related to not being able to interact with objects or solve problems in the way that the player would expect. The findings from the questionnaire were that the effect that intuitiveness had on player enjoyment was higher for people who prefer first-person shooter games and people with more experience playing games.

There were two new themes related to the enjoyment of players interacting in game worlds that were identified in the focus group. The first new theme was immersion, which related to game aspects such as audio and narrative drawing the player into the game, enabling them to believe it is real or suspending their disbelief. The questionnaire further expanded immersion with the findings that sound has a significant effect on the enjoyment of people who prefer first-person shooter games and players with more experience. Also, it was found that narrative has a significant effect on people who prefer strategy games and players with more experience. The second new theme found in the focus group was consistent physics, which related to gravity, momentum and the basic laws of physics behaving consistently with player expectation, as well as more realistic behaviour and interactivity in water, fire and explosions. The questionnaire showed that physics has a greater effect on the enjoyment of people who prefer first-person shooter games.

The background of the sample that participated in the focus group and questionnaire should also be considered. For the focus group, the participants were experienced game-players and as such their opinions and views may differ from a group of novice players. For the questionnaire, the majority of the sample consisted of experienced game-players and a large proportion of the participants were attracted from online game forums, such as Eve Online and EverQuest. As such, the results obtained from the questionnaire may be more biased towards experienced and online game players. Interesting and valuable future work lies in investigating different groups of players and comparing the results to the findings of these studies.

In conclusion, the focus group provided supporting evidence from the player's perspective for some of the issues that were identified in the game development literature, including intuitiveness, consistency and freedom of expression. Additionally, two new issues were defined in the focus group, immersion and physics. As discussed in Chapters 1 and 2, emergent game worlds have the potential to enhance player enjoyment in terms of consistency, freedom of expression and intuitive interactions, which were each shown to be major factors of player enjoyment through the player-centred studies. Additionally, the physics in emergent game worlds is inherently consistent, as the laws of physics are defined globally. Consequently, it can be concluded from the player-centred studies that emergent game worlds have the potential to enhance player enjoyment by providing consistency, freedom of expression, intuitive interactions and consistent physics. The aims of the research reported in the next section were to design, implement and test an emergent game world that has the potential to enhance player enjoyment.

Part II

Designing, Implementing and Testing the EmerGEnT System

4

Cellular Automata in Game Environments

The environment is the central component of an emergent game system as it defines the game world and the interactions that are possible within the world. The rules that are defined for the interactions within the environment itself dictate the rules that will apply to entities that exist in the environment, such as objects and agents. Therefore, defining the rules for the interactions between cells is a crucial step in developing a game world that facilitates emergent behaviour. An active environment, based on simple interactions between cells, provides a foundation for emergent behaviour to occur in game objects and agents, as well as the environment itself. A possible technique that can be used as a foundation for an emergent game environment is cellular automata. However, there is currently no integrated implementation of cellular automata for modelling game environments. The lack of cellular automata in current games gives rise to the questions of whether cellular automata are appropriate for use in game systems and to what extent they can facilitate emergent behaviour and gameplay. Consequently, the aims of the study presented in this chapter were to design and implement a game environment, based on cellular automata, which models basic elements of the environment, such as heat, pressure and fluid. Also, the study aimed to determine the suitability of cellular automata for modelling game systems and to determine the extent to which cellular automata can facilitate emergent behaviour and gameplay. The approach taken was to begin with simplified equations from thermodynamics, implement a two-dimensional cellular automata and simple strategy game environment, tune the rules and properties until reasonable observable behaviour was achieved, and test the system's behaviour with possible strategy game scenarios.

4.1 Strategy Games as a Modelling Environment

Cellular automata could be used to model game environments in a variety of game genres, each with individual constraints. However, a strategy game was chosen as an appropriate test-bed for several reasons. First, the abstract nature of strategy games means that the rules and properties can be more abstract. More specifically, a strategy game is conducted on a map that represents a world or a large region, with each cell representing an area that covers several kilometres. On this scale, there is no need to model effects such as ripples in water or drops of rain causing splashes. Rather, it is the large scale effects, such as forest fires and dams bursting that are important. Second, the environmental interactions are more likely to directly impact on gameplay in a strategy game. As the interactions are inherently on a larger scale, it is more likely that they will have a more significant effect on gameplay. For example, a forest burning down can give the player a way into an opponent's base or destroy a needed resource of wood. Third, due to the abstract nature of the game map, the world only needs to be represented in two-dimensions, as such the cellular automata only needs to be represented in two-dimensions. In games such as first person shooters, the entire three-dimensional world would need to be represented. However, in a strategy game, the important interactions are only occurring on the surface of the world. The system may need to take the height of the surface into consideration, but there is still only one plane of cells that require calculations to be performed. Fourth, strategy games are almost always divided into grids, called influence maps (Tozour, 2001). As influence maps are already widely used in games, a system that uses influence maps is likely to be easier to implement and more acceptable by game developers. Fifth, strategy game maps are generally a lot smaller than maps in other types of games. Strategy games usually have one static map that represents the world, whereas other types of games have multiple cities or regions that continually need to be loaded as the player moves into each region. For these reasons, strategy games have been selected as an ideal test-bed, as they are far simpler, have more obvious effects and will involve far fewer issues in implementing and incorporating into current games.

4.2 Physical Modelling with Cellular Automata

As discussed in the introduction (Chapter 1), most approaches to modelling real-world phenomena in virtual worlds aim to develop accurate, error-free models. These models are usually developed for the purposes of simulating natural disasters (e.g. forest fires) or visually realistic effects (e.g. smoke or fluid flow), using complex equations. However, these computationally-expensive, complex methods are not needed in game worlds, where the emphasis is on credible and acceptable behaviour, rather than accurate and error-free simulation. Games worlds only need to approximately model reality for the purposes of entertainment. Forsyth (2002) has identified ways in which environmental processes can be simplified for games using cellular automata and has formulated some example equations for these processes in human-sized (e.g. first-person shooter) games. In this section, Forsyth's equations for heat, pressure, fluid flow and fire are summarised.

4.2.1 Heat

Forsyth (2002) discusses three different mechanisms for transmitting heat through the environment: conduction, convection and radiation. In conduction, neighbouring cells pass heat to each other until they reach the same temperature (see Figure 4.1). Forsyth also describes mechanisms for convection (heat rises) and radiation (hot objects emit light).

```
// Find current heat capacities
float  HCCell = cell->material->SHC * cell->Mass;
float HCNeigh = neigh->material->SHC * neigh->Mass;
float EnergyFlow = neigh->Temp – cell->Temp;
// Convert from heat to energy
if (EnergyFlow > 0.0f)
        EnergyFlow *= HCNeigh;
else
        EnergyFlow *= HCCell;
// A constant according to cell update speed.
// Usually found by trial and error.
EnergyFlow *= ConstantEnergyFlowFactor;
Neigh->Temp -= EnergyFlow / HCNeigh;
Cell->Temp += EnergyFlow / HCCell;
// Detect and kill oscillations.
if (((EnergyFlow>0.0f)&&(neigh->Temp<cell->Temp)) ||
        ((EnergyFlow<=0.0f)&&(neigh->Temp>cell->Temp)))
{
        float TotalEnergy = HCCell * cell->Temp + HCNeigh * neigh->Temp;
        float AverageTemp = TotalEnergy / (HCCell + HCNeigh);
        cell->Temp = AverageTemp;
        neigh->Temp = AverageTemp;
}
```

Figure 4.1. Equations for heat. Neighbouring cells pass heat to each other until they reach the same temperature (reproduced from Forsyth, 2002, p. 209)

4.2.2 Pressure

Forsyth (2002) provides a general algorithm for diffusion, which calculates the difference in pressure between a cell and its neighbour and divides that amount by the number of neighbours (see Figure 4.2).

```
for (neigh = each neighbour cell)
{
        if (neigh->Material->IsInert( )  ) continue;
        float DPress = cell->Pressure – neigh->Pressure;
        float Flow = cell->Material->Flow * DPress;
        Flow = clamp (Flow,
                Cell->Pressure / 6.0f,
                -neigh->Pressure / 6.0f  );
        cell->NewPressure -= Flow;
        neigh->NewPressure += Flow;
}
```

Figure 4.2. Equations for pressure diffusion. Pressure is divided equally between a cell and its neighbours (reproduced from Forsyth, 2002, p. 205)

4.2.3 Fluid Flow

The equations for fluid flow suggested by Forsyth (2002) are based on his general equations for pressure diffusion (see Figure 4.2). However, fluid is made compressible, so that fluid at greater depth seems to have greater pressure (more fluid stored in the same space). The compression property allows water to behave realistically in three dimensions (see Figure 4.3).

```
if (neighbour cell is above this one)
{
        if ( ( cell->Mass < material->MaxMass ) ||
                (neigh->MaxMass < material->MaxMass ) )
        {
                Flow = cell->Mass – material->MaxMass;
        }else{
                Flow = cell->Mass – neigh->Mass - material->MaxCompress;
                Flow *= 0.5f;
}
else if ( neighbour cell is below this one )
{
        if ( ( cell->Mass < material->MaxMass ) ||
                (neigh->Mass < material->MaxMass ) )
        {
                Flow = material->MaxMass – neigh->Mass;
        } else {
                Flow = cell->Mass – neigh->Mass + material->MaxCompress;
                Flow *= 0.5f;
        }
}
else    // neighbour is on same level
{
        Flow = (cell->Mass – neigh->Mass ) * 0.5f;
}
```

Figure 4.3. Equations for fluid flow. Fluid is made compressible so that water at greater depths has more pressure (reproduced from Forsyth, 2002, p. 207)

4.2.4 Fire

Forsyth (2002) notes that the process of burning materials is extremely complex and identifies that the best results are given by using a function that shows the amount of heat energy that is released per unit of time when a material burns at a certain temperature. In order to model burning in real-time the materials need to be reduced to their main characteristics. With Forsyth's approach, the two variables maximum burning rate and burning temperature can be tweaked to allow the burning of any material to be simulated.

```
float Temp = cell->Temp – material->Flashpoint;
// Damage the cell
CellDamage = Temp * material->BurnRate;
float Burn;
// Convert to actual burning value.
if (Temp > material->MaxBurn * 2 )
        Burn = material->MaxBurn;
else
        Burn = ( 1.0f – ( 0.25f * Temp / material->MaxBurn ) ) * Temp;
ASSERT ( Burn <= material->MaxBurn );
ASSERT ( Burn >= 0.0f );
// And heat the cell up from the burning.
Cell->Temp += Burn * material->BurnTemp;
```

Figure 4.4. Equations for burning. Burning can be reduced to the amount of heat energy that is released per unit of time when a material burns at a certain temperature (reproduced from Forsyth, 2002, p. 211)

4.2.5 Limitations

The equations discussed by Forsyth (2002) provide a foundation for developing a game world based on cellular automata. However, there is a need to develop these equations into a full world model that includes structure, as well as means for integrating the individual components and updating the game world. Also, the equations need to be developed into complete algorithms and adapted to strategy games, as opposed to human-sized games. Finally, extensive tuning is required to ensure the behaviour of the system meets requirements. Consequently, the aim of this study was to develop and tune such a system, which is discussed in the following sections.

4.3 EmerGEnT System Structure

The Emergent Games Engine Technology (EmerGEnT) system designed in this study can be viewed as a hierarchy with three levels (see Figure 4.5). The top level is the behaviour of the system that is observable by the player, such as the effects of fire, damage and fluid flow. The player can see that a cell is on fire or that it is damaged as these are observable effects. The second level consists of the simple rules that give rise to the visible, complex, top-level behaviour. There are two types of rules at the second level of the system. First, there are rules that define the interactions between neighbouring cells, such as the spread of heat from one cell to another. An example rule for interactions between cells is that heat flows from a hot cell to a cooler cell. At the second level, there are also are rules for interactions within a cell, such as the burning of a cell. An example rule for interactions within a cell is that hot cells catch on fire. Finally, the third level of the hierarchy contains the properties of the cells, which determine how cells act and react in accordance with the rules at the level above. For example, each cell is made of a material that has a certain flashpoint, burning temperature and burning rate that determines how hot it needs to be to catch on fire, how fast it burns and how long it will burn. The multi-levelled design of the EmerGEnT system allows the system to be systemic, facilitates emergent top-level behaviour and lends it to future extension.

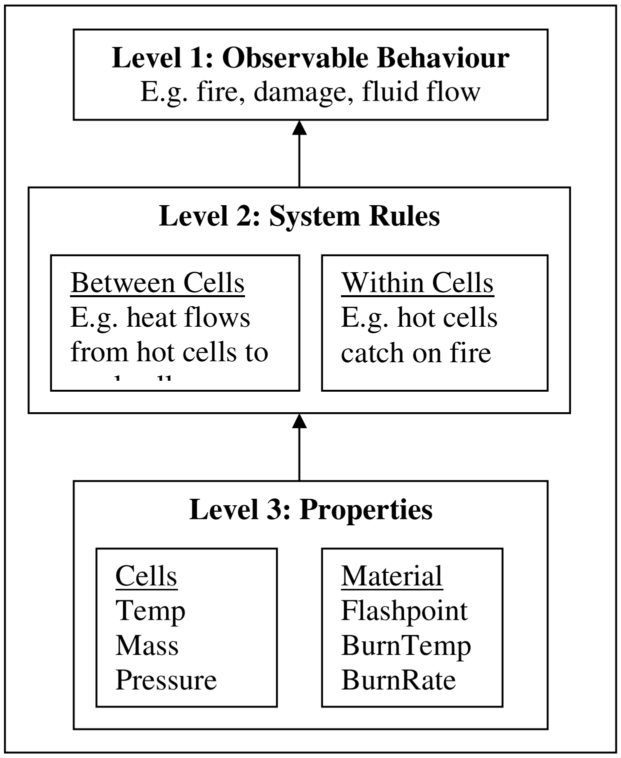


Figure 4.5. EmerGEnT system structure. The multi-levelled structure of the environment facilitates top-level emergent behaviour

4.4 Properties

The bottom level of the EmerGEnT system hierarchy (see Figure 4.5) contains the properties of the cells. This level contains the data structures for the cellular automata and materials of the system. The system consists of a grid of cells and each cell has a set of properties. Included in the set of properties is the material (or terrain) of the cell. Each material also has its own set of properties that govern its behaviour. The main data structure in the system is the grid for the cellular automata. The grid consists of 100 cells (10 by 10), each of which represents a piece of a strategy game map of arbitrary size. In commercial strategy games, the size of the cells that are used in structures such as influence maps is arbitrary and there is a trade-off between accuracy and efficiency (Tozour, 2001). If the cells are too large then the influence map will miss important features and if the cells are too small then there is redundant information and substantial memory is used. Usually, the cells are made fairly large, approximately big enough to fit 10-20 standard units side by side, and from there the cell size is tuned to obtain optimal results.

Each cell is a record with a set of 12 associated properties, including the terrain type (material), temperature, mass, damage, wetness, height, fluid and pressure. There is also a set of materials, which contains information about all the materials in the system. Each material has a set of properties, including flashpoint, burning temperature, maximum burning rate, specific heat capacity (SHC) and maximum fluid level before overflow (see Table 4.1).

Table 4.1. Cell material properties. Possible material for cells include water, grass and woods

Property	***Description***	***Values***		
		Water	*Grass*	*Woods*
Flashpoint	The ignition temperature of the material.	99999	99	2000
BurnTemp	A multiplier for the temperature that the material burns at (amount of heat released).	0	3	5
BurnRate	A multiplier for the rate that the material burns at (rate of consuming fuel).	0	20	10
MaxBurn	The maximum rate that the material can burn at.	0	200	300
SHC	Specific heat capacity – the amount of energy required to heat up this material.	1	100	100
MaxFluid	The maximum amount of fluid a cell can hold.	60	60	60

4.5 Rules for Interactions between Cells

The second level of the hierarchy contains the rules of the cellular automata, which define the interactions between cells. There are two important functions for maintaining the cellular automata, get neighbours and update. Apart from these upkeep functions, there are rules for the environmental systems that use the cellular automata to spread their effects, including heat, fluid flow and pressure. Appendix B contains pseudo-code for the heat, pressure and fluid flow algorithms, which illustrates how the equations are integrated.

4.5.1 Get Neighbours and Update

There are two main processes that each cell in the grid needs to go through, update and get neighbours. In the update process, the new values for fluid, temp, damage and pressure are substituted for the old values in the cellular automata. Also, any rain that is currently falling is added to the fluid value. Finally, the buffer is cleared. The buffer is made up of the outermost layer of cells in each direction and is used to simulate the effects of the system moving out into the world beyond the simulation. For example, heat is released into the buffer each turn and at the beginning of each turn the buffer is reset to zero.

4.5.2 Heat

The algorithm for conduction described in Forsyth (2002) was used as a basis for heat diffusion in the EmerGEnT system. In the EmerGEnT system, each material has a specific heat capacity, which determines how much heat is required to raise the temperature of that material. Materials with a high specific heat capacity require more energy to raise their temperature. In order to diffuse heat in the EmerGEnT system, the heat capacity of the cell, *HCCell*, is first calculated. The heat capacity is equal to the specific heat capacity of the material, *material(cell).SHC*, multiplied by the mass of the material in the cell, *cell.Mass*. The heat capacity for the neighbour, *HCNeigh*, is calculated in the same way.

```
HCCell = material(cell).SHC * cell.Mass
HCNeigh = material(neigh).SHC * neigh.Mass
```

The energy flow, *EnergyFlow*, between the cell and its neighbour is equal to the difference in temperature between the cell, *cell.Temp*, and its neighbour, *neigh.Temp*. The energy flow value is converted from heat to energy by multiplying it by the heat capacity of the transmitting cell. The energy flow is also multiplied by a constant, *ConstantEnergyFlowFactor*, to control the speed of the cell update.

```
EnergyFlow = cell.Temp - neigh.Temp
EnergyFlow *= HCCell
EnergyFlow *= ConstantEnergyFlowFactor
```

Subsequently, the new heat values for the cell, *cell.NewTemp*, and neighbour, *neigh.NewTemp*, are calculated by dividing the energy flow by the heat capacity for each cell.

```
neigh.NewTemp += EnergyFlow / HCNeigh
cell.NewTemp -= EnergyFlow / HCCell
```

To reduce oscillations (heat moving back and forth between the same two cells), the heat of neighbouring cells is distributed evenly if the neighbour cell has more heat than the cells as a result of the heat transfer.

```
TotalEnergy = (HCCell * cell.Temp) + (HCNeigh
  * neigh.Temp)
AverageTemp = TotalEnergy / (HCCell + HCNeigh)
cell.NewTemp = AverageTemp
neigh.NewTemp = AverageTemp
```

The diffusion of heat is increased in the direction that the wind is blowing and reduced against the wind, proportional to the speed of the wind.

Convention and radiation were not modelled in the EmerGEnT system as the scale of the strategy game environment means that these processes are likely to go unnoticed. In a first-person game, where the environment is on a human-sized scale, convection and conduction are likely to be more important. A possible enhancement to the EmerGEnT system would be to model convection to some extent. In a strategy game environment, the most appropriate application of convection would be to allow heat to transfer faster uphill by a small amount. However, such a subtle difference may not be noticed by the player or add to the gameplay in any way.

5.4.3 Fluid Flow

The method suggested for fluid flow by Forsyth (2002) is not well suited to a strategy game environment, as it is designed to simulate the effects necessary in a 'human-sized' game, such as a first-person shooter. For example, it is more suited to water flowing into a container, such as a bucket, and would not be viable for simulating large bodies of water, such as a river in a strategy game. Due to the difference in the scale of a 'human-sized' game and a strategy game, a new approach was needed for the EmerGEnT system. A new algorithm was developed that was similar to the heat and pressure diffusion algorithms, with the addition of terrain height. For heat and pressure, the terrain can be treated as though it is flat. However, for fluid flow, it is necessary to know the contours of the landscape, as fluid has different rules for flowing downhill, uphill and on level ground.

The main rule modelled in the EmerGEnT system's fluid flow algorithm is the flowing of the fluid from one cell to its neighbouring cells. Once the fluid in a cell, *cell.Fluid*, exceeds a certain amount (dependent on the maximum amount of fluid the material in the cell can hold, *material(cell).MaxFluid*) the fluid then flows into the surrounding cells. The flow of fluid between cells is also affected by the relative height of the cell, *cell.Height*, and its neighbours, *neigh.Height.*

```
if (cell.Fluid > (material(cell).MaxFluid
  * (neigh.Height / cell.Height)))
```

Fluid flows faster to lower cells and less fluid is accumulated in a cell before it starts flowing to a lower neighbour. When flowing uphill, more fluid is accumulated in the cell, until the cell overflows to its higher neighbours. Fluid also flows at a slower rate uphill. The steeper the slope, the more effect it has on the flow of the fluid, in terms of the rate of the flow (flow) and the amount of fluid that is accumulated before flowing to the neighbours.

```
flow = flow * (cell.Height / neigh.Height)
```

Also, fluid flows faster when there is a greater difference between the amount of fluid in a cell, *cell.Fluid*, and the amount in the neighbouring cells, *niegh.Fluid*, as there would be greater pressure. The difference is divided by four as each cell has a maximum of four neighbouring cells, providing a good approximation to how the fluid will be divided into the neighbours.

```
    flow = (cell.Fluid - neigh.Fluid) * 0.25
```

Another effect of fluid is that it wets. Material that is wet has a lower temperature, as well as being harder to ignite and slower to burn. The wetness in a cell also reduces slowly overtime, as the moisture evaporates, and gradually repairs damage over time.

Currently, there is only one type of fluid that is modelled in the EmerGEnT system. However, a possible enhancement to the EmerGEnT system would be to model different types of fluids. Different fluids flow at different rates, depending on properties such as their viscosity. However, the need to model fluids other than water would be more relevant to 'human-sized' games, rather than strategy games. In a strategy game, it is unlikely that there will be large bodies of fluid other than water, but a possible example would be lava or petrol, which would flow very differently to water.

4.5.4 Pressure

The EmerGEnT system extends the simple pressure diffusion equations presented by Forsyth (2002) to include a model for explosions. In the EmerGEnT system, the rate of diffusion of pressure, *PressureFlow*, is determined by the difference in pressure between a cell, *cell.Pressure*, and its neighbour, *neigh.Pressure*. As with fluid, the pressure flow is divided by four, as each cell has a maximum of four neighbour cells, providing a good approximation.

```
    PressureFlow = cell.Pressure - neigh.Pressure
    neigh.Pressure += PressureFlow * 0.25
    cell.Pressure -= PressureFlow * 0.25
```

If there is a large enough difference in pressure between two adjacent cells then an explosion occurs, as there is a rapid change in pressure. An explosion occurs when a material produces a large amount of air in a short period of time.

```
    pressure_ratio = cell.Pressure / neigh.Pressure
```

When an explosion occurs in the EmerGEnT system, heat is released in an amount proportional to the difference in pressure (i.e. the size of the explosion).

```
    cell.NewTemp += (explosion_const * pressure_ratio)
     * 0.25
```

If the amount of heat is great enough, then a fire is started (see the fire section) and damage is caused to the surrounding cells. Additionally, explosions cause damage due to high absolute pressures, as well as high pressure differences. Therefore, the EmerGEnT system models the effects of high absolute pressure in cells. A cell with a high enough pressure causes damage to itself and its contents, irrespective of the pressures of the surroundings cells.

A possible enhancement to the EmerGEnT system would be to have the pressure system affecting the wind, so they are part of the same force rather than different forces acting independently. This integration would be particularly useful when objects are introduced into the system. When an explosion occurs, small objects and debris should be picked up and carried by the force of the explosion (Forsyth, 2002).

4.6 Rules for Interactions within Cells

The second level of the EmerGEnT system hierarchy (see Figure 4.1) also includes rules for interactions that occur within a cell. Similar to the rules for interactions between cells, the rules for interactions within cells interact with the lower level of the hierarchy, the properties. However, the rules for interactions within cells are specific to what is happening within an individual cell, irrespective of what is happening in neighbouring cells, and include rules for fire, wind and rain. Appendix B contains pseudo-code for the fire, wind and rain algorithms, which illustrates how the equations are integrated.

4.6.1 Fire

The simulation of fire in the EmerGEnT system is based on the equations provided by Forsyth (2002). In the EmerGEnT system, if the temperature of a cell, *cell.Temp*, exceeds the flashpoint of the material in the cell, *material(cell).Flashpoint*, then the cell ignites. The rate that the cell burns at depends on the burning rate of the material in the cell, *material(cell).MaxBurn*, and the temperature of the cell. The wetness of the cell, *cell.Wetness*, also affects the rate that it burns and how difficult it is to ignite.

```
Temp = cell.Temp - (material(cell).Flashpoint
 + cell.Wetness)
Burn = (1.0 -((0.25 * Temp)/material(cell).MaxBurn))
 * Temp
```

As a cell burns, damage is caused to cell, *cell.NewDamage*, proportional to the temperature of the cell. As a cell becomes more damaged, it burns slower until all the fuel in the cell is used up and it can burn no longer.

```
cell.Damage += (Temp * material(cell).BurnRate)
 - cell.Wetness) * burn_const
Burn -= cell.Damage
```

As the cell burns, the fire releases heat and the cell heats up proportional to the burning rate of the cell and burning temperature of the material, *material(cell).BurnTemp*.

```
cell.NewTemp += Burn * material(cell).BurnTemp
```

Different materials burn at different rates and have different flashpoints. Three materials are modelled in the EmerGEnT system, water, grass and woods. Water cannot burn, grass is easy to ignite and burns quickly, and wood is harder to ignite and burns longer as it provides more fuel per cell.

4.6.2 Wind

In the EmerGEnT system, wind is a global value that is comprised of two components, speed and direction. Wind blows from one of four directions, north, south, east or west. The speed of the wind is set to an arbitrary strength of ten. The only effect of the wind in the EmerGEnT system is that it modifies the spread of heat. Heat spreads slower against the wind and spreads faster in the direction that the wind is blowing, depending on the speed of the wind, *windspeed.*

```
if (neighbour is with wind)
then cell.NewTemp /= 1 + (windspeed * wind_const)
        neigh.NewTemp *= 1 + (windspeed * wind_const)
```

```
if (neighbour is against wind)
then cell.NewTemp *= 1 + (windspeed * wind_const)
        neigh.NewTemp /= 1 + (windspeed * wind_const)
```

Wind currently affects the entire grid in the same way (i.e. there is a uniform wind speed and direction for the entire grid). A possible enhancement to the EmerGEnT system would be to have local wind effects, as opposed to the current global wind effects.

4.6.3 Rain

In the EmerGEnT system, rain effects fluid flow by adding fluid to the cells where it is raining. In turn, cells that contain fluid are also wetted by the fluid, increasing their wetness value. The effects of wetness are described in the section on fluid flow. In the update cycle, the amount of rain in a cell is added to the fluid in the cell.

4.7 Visualisation of the EmerGEnT System

Originally, the EmerGEnT system was implemented in Direct X and visualised in two dimensions (2D). The 2D visualisation included three types of terrain, grass, water and woods, which were represented as light green, blue and dark green, respectively (see Figure 4.6). Also, the observable effects in the system were visualised, including fire, fluid flow, rain and damage caused by heat. Each of the effects were visualised as different coloured pixels, where fire was red, fluid was blue, rain was light blue and damage was black. The number of pixels for each effect in each cell was equal to the magnitude of the effect. For example, as a cell became more damaged there were more black pixels to illustrate the damage.

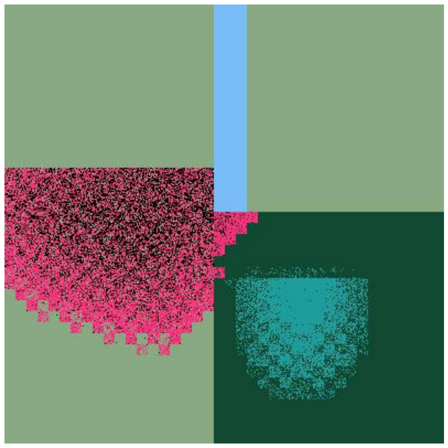


Figure 4.6. 2D EmerGEnT system. The visualisation of the EmerGEnT system in 2D, including the terrain in each cell and the effects of fire, fluid flow, rain and damage

Subsequently, the EmerGEnT system was ported into a three-dimensional (3D) game engine, the Auran Jet (www.auran.com/jet), to provide a realistic game-like environment. Similar to the 2D representation, the 3D representation included the same three terrain types, water, woods and grass. The observable effects are modelled in the 3D system by the use of sprites (2D objects), including a rain cloud, water, fire and damage (see Figure 4.7). The 3D representation also visualises contour of the terrain and includes a user-interface that allows the modification of the scenario by clicking the mouse. The user-interface enabled rapid and dynamic scenario setup for testing the system.

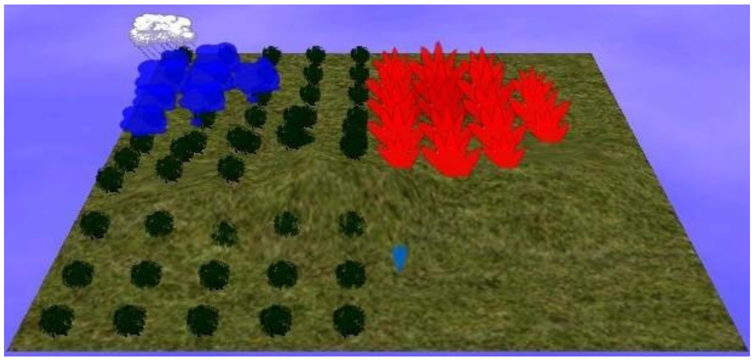


Figure 4.7. 3D EmerGEnT system. The visualisation of the EmerGEnT system in 3D, including the terrain and the effects of fire, fluid flow, rain and damage

4.8 Observable Behaviour

In games, scenarios and sequences of actions are often specifically scripted and coded. The advantage to using a cellular automata and a systemic approach, as used in the

EmerGEnT system, is that the observable behaviour of any object or situation in the game can be dynamic and emergent, without needing to be scripted individually for each object or situation. As a result, the game engine is flexible and responds consistently and realistically to a wide range of actions that the player may take and events in any situation in the game. This section examines four scenarios that are possible situations for a strategy game and evaluates the performance of the CA system and a conventional scripted system in terms of the observable behaviour. The four scenarios are heat and fire, rain and water flow, pressure and explosions, and the integrated system with each of the previous components.

4.8.1 Scenario 1: Heat & Fire

Consider an example where a fire starts in a forest or woods. To appear natural, the fire needs to burn, cause damage to the woods and spread through the woods, all at a believable rate. Subsequently, the fire is blown north to grasslands by wind. As grass has different properties than woods, the fire should behave differently when interacting with the grass, it will burn out faster as the grass provides less fuel, it will spread faster as the grass is easier to ignite and it will release less heat as it burns. Finally, the fire runs into a river that runs across the map and the fire spreads no further as the river cannot be ignited, although some heat can be passed across it (see Figure 4.8).

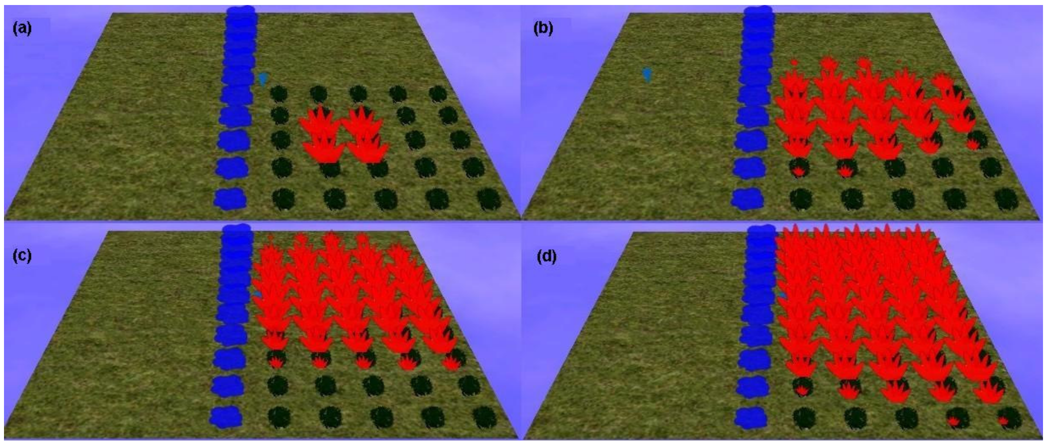


Figure 4.8. Heat and fire scenario in the EmerGEnT system. A fire starts in a forest (a), wind spreads the fire north to grasslands (b-c) and the fire is blocked by a river (d)

In a game where the heat and fire scenario is scripted, the rate at which the fire burns, spreads and damages would need to be specifically coded. Also, the rate of the fire spreading from the woods to the grass would need to be specified, as would the information that the fire must stop burning when it reaches the river. For example, the script would include instructions such as burn each cell of woods for x seconds, spread fire to neighbouring woods cells after y seconds, spread fire from woods to grass after z seconds and so on. If the scenario were changed slightly, such as increasing the density of the woods, then the variables that are dependent on the density of the woods, such as the rate at which the fire spreads in woods, would need to be changed. It becomes apparent that a system with a specifically-coded, static architecture is not robust to even small changes in the heat and fire scenario. Each time a change is made, each variable that is associated with the change would need to be updated by hand, or the system would not work. Furthermore, extending this scripted system would become difficult and awkward. For example, if a new type of terrain were added, such as scrub, then there would need to be new variables for fire moving from woods to scrub, grass to scrub, scrub to woods, and scrub to grass, as well as all the variables for burning scrub. It is clear that the number of instructions needed to run the heat and fire scenario would rapidly grow out of control with even a few small extensions.

In contrast to scripted systems, when the heat and fire scenario is run in the EmerGEnT system, the resulting behaviour depends on the underlying properties of the terrain, which are used by the equations for heat and fire to determine the behaviour of the system dynamically. The approach that is taken when scripting is to attempt to encode the behaviour directly. For example, the woods burn for five seconds because it is told to burn for five seconds. However, the EmerGEnT system uses lower-level information so that the visible behaviour of the system will be emergent. With the EmerGEnT system, the woods burn for five seconds because the system rules consider its flashpoint, specific heat capacity, rate of burning, amount of fuel, temperature and so on, and calculates at each time step whether it is still burning. Due to the extra level of complexity, the EmerGEnT system works on any materials in any scenarios. It may seem more complex to make these calculations, but there is one set of formulae that applies in any situation with any material and the only difference is the properties of each material. Therefore, once the set of formulae is coded, the burning of any material can be simulated as long as the values of the properties are available, making it a data-driven system. There is no awkward expansion of the system, no limit to the number of materials that can be added and a great deal of flexibility in handling new situations, which reduces the difficulty and cost of quality assurance.

4.8.2 Scenario 2: Rain & Water Flow

The second scenario presents a case study with rain and water flow, in which rain falls on the map in a position that is at the top of a hill. A certain amount of the water is able to soak into the ground at the top of the hill (depending on the terrain), but after some rain has fallen the water starts to run down the hill. The harder the rain, the faster the water runs down the hill. Also, the steeper the slope, the faster the water runs down the hill. At the bottom of the hill is a valley, which rises back up into a hill on the other side. When the water reaches the valley it starts to build up and runs outwards to fill the level ground in the valley. With sufficient rainfall, the water will fill the valley and start to flow back up the hills on either side of the valley (see Figure 4.9).

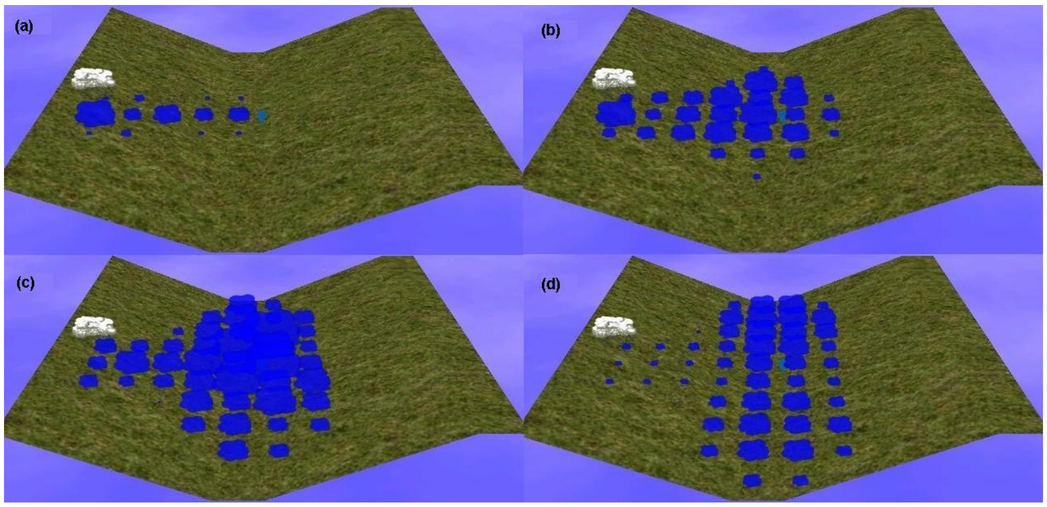


Figure 4.9. Rain and water flow scenario in the EmerGEnT system. Rain falls at the top of a hill (a) and the water runs down the hill (b) and spreads out in the valley below (c-d)

When scripting the rain and water flow scenario specifically, the script would need to encode information for where the water will run, how fast it will run and other behaviours such as how deep it will pool in different areas. Scripting the behaviour of the water specifically would be relatively simple, but would rely on the rain falling in the same position and at the same rate and the contours of the landscape always being the same. However, if the rain were to fall from a different position, or if the contours of the landscape were to change even slightly, then the behaviour of the water would be completely different and a new script would be needed. So, if the rain and water flow scenario was an event in a game that always played out the same way, then a script would be fine. However, if rain was a random event that occurred in the game, then it could not be foreseen where it would fall or what path it should follow.

In the EmerGEnT system, the position of the rain, the speed of the rain and the contour of the landscape are not a problem. The EmerGEnT system makes calculations every time step for which way the water will flow, how fast it will flow and how deep it will pool. The EmerGEnT system uses its formulas and the data it is provided, such as the relative height of the neighbours of each cell and the amount of water in each cell, and it dynamically calculates the behaviour of the water at each time step. As a result, the EmerGEnT system can accommodate changes in the environment and simulate the flow of water down the hill for different positions and speed of the rain and for different contours of the hill. The EmerGEnT system uses its rules and properties to determine the behaviour of the water in real-time, dependent on the current situation, rather than have the behaviours of the water pre-scripted.

4.8.3 Scenario 3: Pressure & Explosions

The third scenario presents a case study with pressure and explosions, in which an explosion occurs somewhere on the map. The high absolute pressure of the explosion causes immediate damage in the vicinity of the explosion. The area that is affected by the pressure depends on the magnitude of the explosion. As well as the damage from pressure, energy is generated as a result of the difference in pressure between the explosion and the surrounding area. As a result of the energy (heat) that is released, a fire is started in the area around the explosion. The fire then spreads and causes damage to a wider area around the initial explosion (see Figure 4.10).

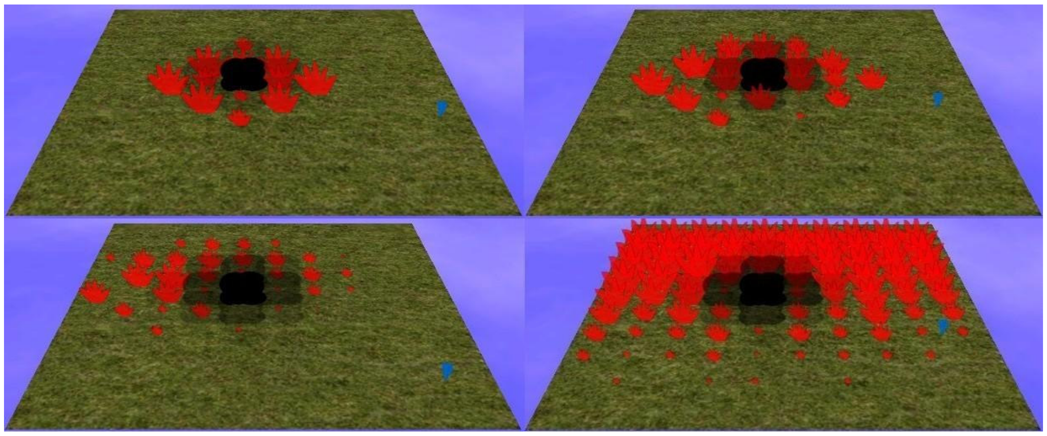


Figure 4.10. Pressure and explosions scenario in the EmerGEnT system. An explosion occurs, the high absolute pressure causes immediate damage and a fire starts as a result of the heat released by the explosion. Each panel shows different sized explosions

An explosion is relatively simple to simulate in the conventional manner, and is commonly scripted in many commercial games. However, there is a difference between a scripted explosion occurring that is triggered by a bomb going off and an explosion naturally occurring as there is suddenly a large difference in pressure between two cells. With the scripted example, there must always be a high-level trigger for the explosion, such as a bomb, a grenade or some other discrete explosion event. On the other hand, if there are conditions under which an explosion occurs, such as a large difference in pressure brought about by piercing a gas cylinder, then the explosion will naturally occur when these conditions are met. Therefore, there doesn't need to be a discrete event, rather the explosion will occur because an explosive device has just caused a large difference in pressure or for unforeseen reasons.

Another interesting aspect in the pressure and explosions scenario is the second degree effect of the explosion, namely the fire that is started as a result of the rapid increase in heat around the explosion. This second degree effect could be manufactured in a scripted explosion, but would not reach the same level of realism as the EmerGEnT system, which can dynamically take into consideration the type of terrain, objects in the area and other important factors. Also, depending on the magnitude of the explosion, there would be different amounts of heat generated and different effects of the fire. Although this variation could be scripted, it would not have the flexibility of the EmerGEnT system. For example, if an explosion occurs in an area containing highly flammable material, the behaviour of the EmerGEnT system would be significantly different, but a scripted explosion would only consider the magnitude of the explosion and burn an area around the explosion accordingly.

4.8.4 Scenario 4: Integrated System: Heat, Fluid & Pressure

The fourth scenario is a combination of the previous three case studies and illustrates the real power of the EmerGEnT system. An explosion occurs in the woods and the high pressure causes damage to the immediate area. As a result of the energy released from the explosion a fire is started, which then spreads through the woods. A wind then blows the fire west into a neighbouring grassy valley, which contains a small village. It has been raining at the top of the hill next to the valley and the water flows down the hill into the valley and puts out the fire. The fire recedes into the woods and burns until the woods are burnt out and there is no more fuel (see Figure 4.11).

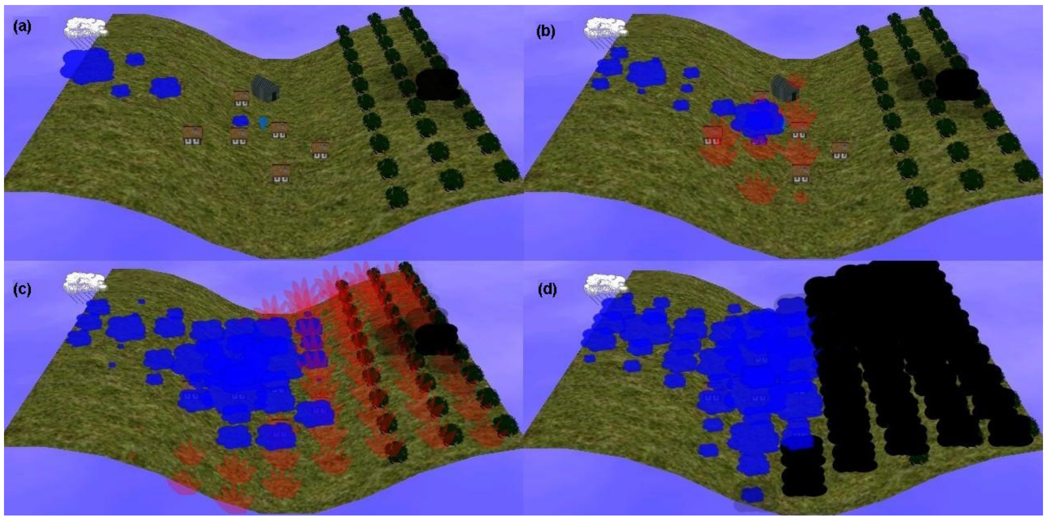


Figure 4.11. Integrated scenario in the EmerGEnT system. An explosion in the woods causes immediate damage and starts a fire (a). The fire is blown west by the wind (b) and extinguished by the water flowing down the hill from the rain falling above (c-d)

The complex situation described in the integrated scenario demonstrates the benefit of the emergence of the EmerGEnT system. Rather than the complex behaviour having to be specifically scripted, it just happens as a result of all the components of the system working together. Each element in the EmerGEnT system looks after its own behaviour, the heat spreads, the fire burns, the water flows and so on, dependent on its own set of rules and the properties and state of the materials it is acting on. As these elements simultaneously work independently, they impact on each other and give rise to emergent, complex behaviour that wasn't specifically programmed into them. The observable complex behaviour in the scenario arises from the independent behaviour of the elements, making the whole more than the sum of its parts.

It would take a great deal of effort to specifically script the complex behaviour of the environment in the integrated scenario, as well as careful attention to detail. Also, every small change to the scenario would result in different desired behaviour, requiring the scenario to be rescripted. Therefore, specifically scripting a scenario as complex as the fourth scenario would require substantial initial effort as well as significant ongoing effort to update and maintain the system. On the other hand, the use of cellular automata means that the complex behaviour arises as a result of the individual elements interacting with each other. The EmerGEnT system accommodates the possible variations in a complex scenario, such as different types of terrain, the affect of wind, the contours of the terrain, the interactions of heat, pressure and fluid, without the need to recode for each specific scenario.

4.9 Discussion and Conclusions

The goal of the presented study was to develop a system that supports the requirements defined by game developers and players as discussed in Chapters 2 and 3, including consistency, immersion, intuitive interactions, freedom of expression and realistic physics. The steps taken to achieve this goal were to determine the properties and rules that need to be modelled to create the basic environment of a game system that supports these requirements. In order to develop the rules for the EmerGEnT system, simplified equations from thermodynamics were developed into three complete, independent systems: heat, fluid flow and pressure, with local effects of rain, explosions and fire. The heat, fire and pressure systems were based on the equations presented by Forsyth (2002). The fluid flow system required an entirely new approach to be appropriate for a strategy game. Finally, explosions, rain and wind were new initiatives of the EmerGEnT system.

Each of the major systems, heat, fluid and pressure, demonstrated various advantages over conventional scripted approaches. For heat and fire, the major advantage was that any number of terrain types could be added to the system and no additional scripting or calculating would be required. In contrast, in a conventional system, the script would need to be updated for each new material and transition between materials. In the rain and fluid flow system, the advantage was found in the contours of the terrain. As the EmerGEnT system used the underlying rules and properties to calculate the speed and direction of the flow, any possible terrain contours could be accommodated, whereas a scripted system would need to have the contours of the terrain specified in advance. For the pressure and explosions system, advantages were that the explosions could occur naturally due to a difference in pressure between two cells, rather than needing an event to trigger the explosion. Also, a fire was started as a second degree effect of the explosion, if sufficient heat was generated by the explosion to ignite the material. Each of these sub-systems had advantages over conventional scripting methods, which were mostly related to the EmerGEnT system being able to dynamically accommodate changes or variations to the environment that a scripted system would be unable to do. However, the real power of the EmerGEnT system became apparent when the heat, pressure and fluid flow systems were combined. In combining these systems, the EmerGEnT system was able to demonstrate the emergent interactions that take place in a complex scenario that would otherwise need to be scripted specifically, such as water flowing downhill to put out a fire that had been blown there from an explosion to the east.

It is important to note that emergent systems based on cellular automata also possess drawbacks and issues. Often, emergent systems require extra time in development to design and tune the rules and properties. Additionally, while an emergent system can open up the game world for the player's expression and experimentation, the game developer can lose the ability to setup specific narrative by being unable to predict and control exactly what will happen in a given situation. Therefore, developing a custom emergent game environment is not necessarily the best approach for game developers with limited time and resources. Ideally, the EmerGEnT system could be made into middleware that game developers can easily integrate into their game, without the issues that relate to designing and tuning the system. Finally, it is likely that the best results will involve integrating scripted scenarios seamlessly into an emergent system, to enable open-ended gameplay while still advancing narrative at the designer's discretion.

In conclusion, the reported study provided evidence that cellular automata are a suitable foundation of a game system that allows emergent interactions. Furthermore, the EmerGEnT system, which was based on cellular automata, demonstrated various advantages over conventional scripted systems. The major advantages were related to the ability of the cellular automata to dynamically respond to changes in the environment and the extensibility of the system. Finally, it was in combining each of the component systems of the cellular automata to form the integrated EmerGEnT system that the power of cellular automata to create an emergent environment became evident. The cellular automata techniques that were used and demonstrated in the EmerGEnT system form a compelling case for using cellular automata in games. Further extension and experimentation with the EmerGEnT system will potentially lead to richer gameplay and playing experiences.

5 Property-Based Game Objects

Game objects are an integral part of any game world as they compose the major source of player interactions. Objects in games are numerous and varied and include weapons (e.g. guns, swords) in first-person shooter games, quest items (e.g. holy grail, diary) in role-playing games and buildings (e.g. house, factory) in strategy games. Each type of game object interacts with the game environment and the player in different ways, which gives rise to interesting possibilities for action for the player but also complicates the job of the game developer.

In many games, the properties and parameters of objects are different for each instance and objects have explicit relationships with every other game element to define their interactions. As discussed in Chapters 2 and 3, specifically scripting game objects causes various problems for the game players, such as unintuitive interactions and inconsistencies. Also, problems exist for the game developers, such as substantial effort in designing, maintaining and debugging the game objects (Church, 2002; Smith, 2002).

An alternative to scripting the interactions for each object specifically is to plan for types of interactions, based on general properties (Church, 2002). The questions that arise are how can game objects with a property-based design be incorporated into an active game environment and to what extent can property-based game objects facilitate emergent behaviour and gameplay? Consequently, the aims of the study presented in this chapter were to design and integrate property-based game objects into the EmerGEnT system and to determine the extent to which property-based objects and the active game environment can facilitate emergent behaviour and gameplay. The approach taken was to first implement objects as though they were cells, using the same low-level properties based on the object's material. Subsequently, objects were also imbued with high-level properties, based on their structure, to constrain the possible interactions of the objects.

5.1 Object Structure

In the EmerGEnT system, the objects were designed to have a similar structure as the cells of the environment. Each object has a set of properties that include coordinates (position in the cellular automata), temperature, pressure, fluid, mass, wetness and material. The materials that objects are composed of have the same properties as materials for the terrain, including flashpoint, burning temperature, specific heat capacity and maximum burning rate. In strategy games, the majority of objects in the environment are buildings, such as houses, bunkers or factories. Therefore, the materials that have been chosen for the EmerGEnT system are common types of building materials, such as wood, metal and brick. These materials have significantly varying properties and are therefore ideal for illustrating the emergent behaviour of the system when interacting with different objects (see Table 5.1). First, metal is hard to ignite, transfers heat easily and absorbs little fluid. Second, wood is much easier to ignite but doesn't transfer heat as well and is far more absorbent. Third, brick is moderate to ignite, poor for heat transfer and can absorb a moderate amount of water. In addition to these low-level properties, objects also have a set of high-level properties (represented as Boolean values) that encode attributes of the object's physical structure, including whether the object has volume, is solid or is open.

Table 5.1. Object material properties. Possible materials for objects include wood, metal and brick

Property	***Description***	***Values***		
		Wood	*Metal*	*Brick*
Flashpoint	Ignition temperature of the material	2000	5000	4000
BurnTemp	Multiplier for the temperature the material burns at (amount of heat released)	5	10	8
BurnRate	Multiplier for the rate the material burns at (rate of consuming fuel)	10	5	7
MaxBurn	Maximum rate the material can burn at	300	100	200
SHC	Specific heat capacity – the amount of energy required to heat up this material	100	50	150
MaxFluid	The maximum amount of fluid the material can absorb	50	0	20
Strength	Modifier for the pressure the material can withstand before it breaks	0.6	1.0	0.8

5.2 Object Design

At the basic level, objects are the same as cells in the environment in that they both exist in the physical world and are therefore subject to the same rules of physics, such as heat transfer, fluid flow and pressure. However, whereas all cells are uniform in structure, in that they are all squares of terrain, objects have comparatively complex physical structures. In order to model the structure of objects at the same level as the rest of the system, it would be necessary to divide the objects up into cells in the same way the terrain is divided into cells. Unit mapping objects would allow the cellular automata to dynamically determine the contours (i.e. structure) of the object in the same way as the contours of the terrain are dynamically determined. However, there are two main considerations with breaking objects down into cells. First, given the number of objects in a game environment and the complexity of game objects, it would be prohibitively expensive to perform the necessary calculations. Second, the relative benefit to the game would be minimal, especially in comparison to the cost involved. For example, whereas it is of great benefit to a game to be able to map how water would flow from one cell of the map to the next, it is much less important to calculate exactly how water will flow into an object, such as a bucket. However, knowing that water will flow into the bucket has significant potential in terms of emergent gameplay, in that a player could fill up a bucket with water, carry it to a fire and extinguish the fire.

The potential benefit of structural object properties to gameplay lies in knowing the important features of the game objects that will have an impact on the possible actions and interactions of the objects. A possible solution is to give the objects descriptors of the important features of their physical structure that predispose them to certain actions and behaviours. For example, only an object that has volume can be filled with water, so that a bucket can be filled with water but a sword cannot. Consequently, it can be seen that a logical and computationally viable solution is to have two different levels of properties for objects. As with cells, objects have low-level properties, which define how the matter of the objects interacts in the world. Additionally, objects have high-level properties, defined by the structure of the object, which determine whether it is structurally able to participate in certain interactions. As a result, objects in the EmerGEnT system have low-level properties related to their composition and high-level properties related to their structure.

Having two levels of properties necessitates a two-part approach to modelling objects in a game environment. The first part of the approach is to treat the objects as cells, where each object has only one neighbour (its host cell). The second part of the approach is to consider the high-level properties of the objects. High-level properties can be assembled into affordances that determine whether the object is able to participate in each interaction. For example, only objects that afford flow can participate in fluid or pressure flow. The two-part approach used for defining the interactions of objects with the environment in the EmerGEnT system is described in the following sections.

5.3 Low-Level Properties

At the basic level of physical interactions, objects are the same as cells, as they are both entities in the physical environment that are subject to the rules of physics. Therefore, the first part of the approach to modelling objects in the EmerGEnT system is to treat the objects as cells, where each object interacts exclusively with its host cell. There are two key types of interactions possible between objects and cells. First, objects within a cell are affected by the cell in the same way that neighbouring cells are affected by the cell (i.e. exchange of heat, pressure and fluid). Second, objects affect their host cell as if the host were the object's only neighbour. Therefore, the object both affects and is affected by its host cell.

In some components of the EmerGEnT system, such as heat transfer, there is no difference between object-to-cell interactions and cell-to-object interactions. However, in other components there needs to be a differentiation between these two types of interactions. For example, a high-pressure object in a comparatively low-pressure cell will explode. However, the effect of the inverse of this state, a high-pressure cell (with a low or high-pressure object), is to damage anything within the cell from high absolute pressure (including the cell itself), rather than the cell exploding. The interactions that take place between cells and objects are described in this section, with the differences from cell-to-cell interactions highlighted. Appendix C contains pseudo-code for the heat, pressure, fluid flow, fire, wind and rain algorithms, which illustrates how the equations are integrated.

5.3.1 Heat

When transferring heat between two entities, whether they are two cells or an object and a cell, the heat capacities of the two entities first need to be calculated. The main difference between transferring heat between two cells and between a cell and an object is that objects are much smaller than cells, indicated by the mass of the object. As such, the heat transfer between an object and a cell will be much less than the transfer between two cells of the same size. The heat capacity of the object, *HCObj*, is equal to the specific heat capacity of the object's material, *material(obj).SHC*, multiplied by the mass of the object, *obj.Mass*. Similarly, the heat capacity of the cell, *HCCell*, equals the specific heat capacity of the cell's material, *material(cell).SHC*, multiplied by the mass of the cell.

```
HCObj = material(obj).SHC * obj.Mass
HCCell = material(cell).SCH * cell.Mass
```

The energy flow, *EnergyFlow*, between the object and the cell is equal to the difference between the cell's temperature, *cell.Temp*, and the object's temperature, *obj.Temp*.

```
EnergyFlow = cell.Temp - obj.Temp
```

Once the energy flow is calculated, it can be used to determine whether energy will flow from the object to the cell or from the cell to the object. If *EnergyFlow* is a positive value, then the cell's temperature is greater than the object's temperature and heat will flow from the cell to the object. Otherwise, heat will flow from the object to the cell. In the case where the cell's temperature is greater and heat flows from the cell to the object, the *EnergyFlow* is converted from heat to energy by multiplying by the heat capacity of the cell.

```
    EnergyFlow *= HCCell
```

Otherwise, if energy is flowing from the object to the cell, then the *EnergyFlow* is converted from heat to energy by multiplying by the heat capacity of the object.

```
    EnergyFlow *= HCObj
```

In both cases, the *EnergyFlow* is then multiplied by a constant, *ConstantEnergyFlowFactor*, to control the speed of the cell update.

```
    EnergyFlow *= ConstantEnergyFlowFactor
```

Subsequently, the new heats for the cell, *cell.NewTemp*, and the object, *obj.NewTemp*, are calculated by dividing the *EnergyFlow* by the heat capacity for the cell and object, respectively. In the case where heat flows from the cell to the object, the cell's temperature decreases and the object's temperature increases. Otherwise, if the heat flows from the object to the cell then the object's temperature decreases and the cell's temperature increases.

```
    cell.NewTemp (+/-) = EnergyFlow / HCCell
    obj.NewTemp (+/-) = EnergyFlow / HCObj
```

To avoid oscillations (heat moving back and forth between the object and cell), the difference in heat is distributed evenly between the object and the cell if the transfer of heat would result in the entity that previously had less heat having more heat after the exchange.

```
    TotalEnergy = (HCObj * obj.NewTemp) + (HCCell
      * cell.NewTemp)
    AverageTemp = TotalEnergy / (HCObj + HCCell)
    obj.NewTemp = AverageTemp
    cell.NewTemp = AverageTemp
```

5.3.2 Fluid Flow and Wetness

There are two main interactions of fluids with objects: flow and wetness. Fluid flow between an object and a cell is relatively simple compared to fluid flow between cells, as there is no height difference between a cell and the objects in the cell. There are two cases for fluid flow between an object and a cell modelled in the EmerGEnT system, flow from cell to object and flow from object to cell. The two fluid flow cases are discussed in this section. The wetness of an object is simply dependent on the wetness of the object's host cell and the absorbency of the object's material.

5.3.2.1 Flow from Cell to Object

Fluid flows from a cell to an object within the cell if the cell contains fluid and if the object contains less than the maximum amount of fluid it can hold, which depends on the size of the object. Also, the object must afford flowing, which is discussed in the section 5.4. The amount of fluid that flows into the object from the cell, *flow*, is equal to the difference between the amount of fluid in the cell, *cell.Fluid*, and the amount of fluid in the object, *obj.Fluid*. This difference is then divided by four, as the fluid from the cell must be divided amongst the cell's neighbours and the object and one quarter provides a good approximation. Also, the difference between *cell.Fluid* and *obj.Fluid* is multiplied by the ratio of the mass of the object, *obj.Mass*, to the mass of the cell, *cell.Mass*, to account for the difference in size between the cell and the object. Consequently, the object is filled with the same proportion of fluid as is in the cell.

```
flow = (cell.Fluid - obj.Fluid) * 0.25
 * (obj.Mass / cell.Mass)
```

5.3.2.2 Flow from Object to Cell

Fluid flows from a cell when the material in that cell contains the maximum amount of fluid that it can hold. As a result, the excess fluid spills from the cell into its neighbouring cells. However, fluid flow from an object does not depend on the amount of fluid that the object's material can hold. Rather, it depends on the maximum amount of fluid that the structure of the object can hold, which depends on the volume of the object. When the fluid in an object exceeds the volume of the object, as determined by the mass of the object, the excess fluid spills over into the object's host cell. The excess fluid in an object, *excess*, is the difference between the amount of fluid in the object, *obj.Fluid*, and the mass of the object, *obj.Mass*.

```
    excess = obj.Fluid - obj.Mass
```

If the object contains more fluid then it can hold then it overflows into the host cell. The amount of fluid that flows, *flow*, from the object to the cell is equal to the excess, multiplied by the ratio of the *obj.Mass* to the *cell.Mass*.

```
    flow = excess * (obj.Mass / cell.Mass)
```

5.3.3 Pressure

The three main interactions of pressure with objects modelled in the EmerGEnT system are high-pressure damage, flow and explosions. Objects that are located in a cell that has high enough pressure are damaged, as is the cell itself. Similar to fluid, pressure can also flow from objects to cells and cells to objects. Finally, if an object contains significantly more pressure than its host cell then it will explode, under the right conditions. The pressure ratio between an object and a cell that is necessary for an explosion to occur depends on the object's material. For example, a metal crate can hold more pressure than a wooden crate before it explodes, due to the strength of the material. Additionally, the metal crate exploding results in a bigger explosion than the wooden crate due to the increased pressure capacity.

In the environment study reported in Chapter 4, neighbouring cells with high pressure differences exploded. However, it was decided that cells themselves should not explode in the EmerGEnT system, due to the scale of cells in a real-time strategy game. It would make sense for small cells in a human-sized game to be able to explode based on pressure differences, but not for cells several kilometres across to explode in strategy games. Instead, only objects can explode within cells in the EmerGEnT system. The rules used for each of the pressure interactions are explained in this section.

First, if an object is located in a cell that has high absolute pressure then the object will be damaged. If the pressure of the cell, *cell.Pressure*, is greater than the constant for high absolute pressure, *high_pressure*, then the object will incur damage, *obj.NewDamage*, proportional to the cell's pressure.

```
obj.NewDamage += cell.Pressure*pressure_damage_const
```

If the pressure in the cell, *cell.Pressure*, is higher than the pressure in the object, *obj.Pressure*, then pressure flows from the cell to the object. Pressure only flows between a cell and an object if the object affords flow, as discussed in the section 5.4. The amount of pressure that flows from the cell to the object, *PressureFlow*, is equal to the difference between the cell's pressure, *cell.Pressure*, and the object's pressure, *obj.Pressure*. The rate of the *PressureFlow* between a cell and an object is modified by the ratio of the size of the object, *obj.Mass*, to the size of the cell, *cell.Mass*.

```
PressureFlow = (cell.Pressure - obj.Pressure)
  * (obj.Mass / cell.Mass)
```

If the pressure in an object is substantially greater than pressure in its host cell, then it is possible that an explosion will occur. First, the ratio of the pressure, *pressure_ratio*, in the object to the cell is calculated.

```
pressure_ratio = obj.Pressure / cell.Pressure
```

If the *pressure_ratio* is greater than the pressure ratio required for an explosion, *explosion_ratio*, then an explosion will occur. The *explosion_ratio* is modified by the strength of the object's material, *material(obj).Strength*, so that the *explosion_ratio* required for an object that is made of a relatively weak material, such as wood, is much lower than for an object made of a much stronger material, such as metal. Also, the object must afford exploding, as discussed in the section 5.4. The explosion releases an amount of heat that is proportional to the size of the explosion, *pressure_ratio*, multiplied by a constant, *explosion_const*. The heat generated by the explosion of the object is released into the host cell, increasing the temperature of the cell, *cell.NewTemp*.

```
if ((pressure_ratio > (explosion_ratio
  *material(obj).Strength)) and AffordsExploding(obj))
then cell.NewTemp += pressure_ratio * explosion_const
```

When an explosion occurs, as well as releasing heat, the pressure from the object is transferred to the cell. The amount of pressure that flows from the exploded object to the cell, *PressureFlow*, is equal to the difference between the pressure of the object and the pressure of the cell.

```
    PressureFlow = obj.Pressure - cell.Pressure
```

If an explosion does not occur because the *pressure_ratio* is not high enough then pressure will simply flow from the object to the cell. In order for pressure to flow from the object to the cell, the object must afford flowing, as discussed in the next section. The amount of pressure that flows from the object to the cell, *PressureFlow*, is equal to the difference in pressure between the object and the cell. The rate of the *PressureFlow* between an object and a cell is modified by the ratio of the size of the object, *obj.Mass*, to the size of the cell, *cell.Mass*.

```
    PressureFlow = (obj.Pressure - cell.Pressure)
     * (obj.Mass / cell.Mass)
```

5.3.4 Fire

The rules for the burning of an object are the same as the rules for the burning of a cell of the terrain, as burning is only dependent on the properties of the object or cell and its material. The burning temperature of the object, *Temp*, is the difference between the temperature of the object, *obj.Temp*, and the flashpoint of the object's material, *material(obj).FlashPoint*, modified by the wetness of the object, *obj.Wetness*.

```
    Temp = obj.Temp - (material(obj).FlashPoint
     + obj.Wetness)
```

If the burning temperature is greater than zero, then the object will incur damage proportional to the burning temperature multiplied by the burning rate of the object's material, *material(obj).BurnRate*, modified by the wetness of the object and multiplied by a constant.

```
    obj.NewDamage += ((Temp * material(obj).BurnRate)
     - obj.Wetness) * burn_const
```

The burning temperature of the object must then be converted to an actual burning value, *Burn*. If the burning temperature is greater than double the maximum burning rate of the object's material, *material(obj).MaxBurn*, then *Burn* is equal to the maximum burn rate of the object's material. Otherwise, *Burn* equals one minus the temperature divided by the maximum burn rate, multiplied by the temperature.

```
    Burn = (1.0 - ((0.25*Temp) / material(obj).MaxBurn))
      * Temp
```

As the object burns, the fire releases heat and the object heats up from burning. The amount the object is heated is proportional to the burning value multiplied by the burning temperature of the object's material, *material(obj).BurnTemp*, and a constant value.

```
    obj.NewTemp += (Burn * material(obj).BurnTemp)
      * BurnHeatConst
```

5.3.5 Wind and Rain

Currently, wind and rain do not directly affect the objects in the EmerGEnT system. Instead of rain directly affecting the objects, rain affects the fluid in a cell, which determines the wetness of a cell, which in turn determines the wetness of the objects in the cell. Wind does not affect objects, as the wind is on the scale of cell to cell, rather than within a cell. Within the context of a strategy game, it is unlikely that wind would affect the objects in the environment as they are mostly very large objects, such as buildings. Wind would be a more interesting effect modelled in human-sized games, such as first-person shooter or role-playing games, where there are small objects to blow around, such as paper and cans.

5.4 High-Level Properties

Whereas the low-level properties discussed in the previous section are pertaining to an object's composing material, the high-level properties relate to an object's structure. Low-level properties consist of continuous numeric values, such as a material's flashpoint or burning temperature, which are substituted into equations to determine when and how that material will be affected by heat, water and pressure. In contrast, high-level properties consist of discrete descriptors of an object's structural properties, which constrain the interactions that an object will be able to take part in due to its physical structure. For example, only an object that has volume will be able to hold water, whereas an object's volume has no effect on whether it will catch on fire. High-level properties are implemented in the EmerGEnT system as Boolean values, such as "is_open = false" or "has_volume = true".

Objects with different combinations of high-level properties afford different types of interactions. An object's high-level properties can be combined to form different affordances. For example, for an object to afford fluid flow it must have volume, some kind of opening and be solid (as opposed to perforated). Also, an object can only contain a large amount of pressure if it does not have an opening. Affordances can then be used as preconditions for certain interactions. For instance, only objects that afford flow can engage in fluid or pressure flow. Affordances are implemented in the system as functions that read in an object, check the relevant properties of the object and return a true or false value to indicate whether the object affords the given behaviour.

It would be possible to define affordances for materials in the same way as for structure. For example, rather than having a flashpoint, burning temperature and so on, a material could simply afford burning. However, encoding these properties at a high level would reduce the emergent potential of the environment. These properties are modelled at a low-level so that the high-level behaviour will be emergent. Furthermore, modelling the high-level properties at a low-level (reducing the object to a set of cells to define its structure) would be impractical and provide little benefit to the gameplay in comparison to the computational complexity. Therefore, it is necessary to separate the properties of objects into two levels, a low level related to the object's material to allow emergent behaviour and a high level related to the object's structure to constrain the object's set of interactions to what is possible given its physical structure.

5.4.1 Heat and Fire

Heat is a system that is solely dependent on the composition of an object. The structure of an object does not determine whether it can transfer heat, only the material of the object does. For example, an object made of metal will transfer heat in the same way, dependent on the rules of heat transfer, whether it is a metal crate or a metal chair. Similarly, the burning of an object only depends on its material, not its structure. A wooden table will burn in the same way as a wooden house, following the rules of fire, which take the object's mass into consideration.

5.4.2 Fluid Flow and Wetness

There are two behaviours of fluids that need to be considered with respect to objects. First, fluids can fill objects that have structures that afford filling. Second, fluids can wet objects, dependent on the material of the object. The second behaviour depends solely on the material of the object. Objects made of wood can absorb a considerable amount of water, whereas objects made of metal can absorb substantially less water. As discussed in the previous section, the wetness of an object is simply determined by the wetness of its host cell and the object's material. In contrast, objects can only be filled with fluid if their structure affords filling or flow. There are three main high-level properties that have been identified for an object to afford flow. First, an object must have volume, as an object such as a sword or a book cannot hold fluid. Second, an object must have an opening for the fluid to flow into, as a crate that is sealed has volume but no way for the fluid to enter. Third, the object must have solid sides (as opposed to perforated), as a crate that is made of wire mesh would not be able to hold fluid. Therefore, an object that *has_volume*, *is_open* and *is_solid* affords flow. The function *AffordsFlow* is used as a precondition for fluid flowing from an object to a cell and from a cell to an object.

```
AffordsFlow (object obj)
{
   if (obj.has_volume and obj.is_open
    and obj.is_solid)
   then return true
}
```

5.4.3 Pressure

There are two affordances relevant to pressure modelled in the EmerGEnT system. First, pressure can only flow to and from objects that afford flow, identical to fluid flow. Second, only objects with structures that afford exploding (can contain high pressure) are able to explode. For the first behaviour, pressure flow, the function *AffordsFlow* is used as a precondition for pressure flowing from an object to a cell and from a cell to an object. Therefore, only an object that *has_volume*, *is_open* and *is_solid* affords pressure flow. The same three properties used in fluid and pressure flow are relevant to whether an object affords exploding. First, in order for an object to hold the necessary pressure, the object must have volume. Second, the object must have solid sides to keep in the pressure. Third, the object must be closed (or not open), which also keeps in the pressure. If the object does not have these properties then it is not possible for the pressure inside the object to build up enough to exceed the external pressure so much that it explodes. Therefore, an object that *has_volume*, not (*is_open*) and *is_solid* affords exploding. The function *AffordsExploding* is used as a precondition for an object exploding.

```
AffordsExploding (object obj)
{
   if (not(obj.is_open) and obj.has_volume
    and obj.is_solid)
   then return true
}
```

5.5 Observable Behaviour

In games, scenarios and sequences of actions are often specifically scripted and coded. The advantage to using a cellular automata and a systemic approach is that the observable behaviour of any object or situation in the game can be dynamic and emergent, without needing to be scripted individually for each object or situation. This allows the game engine to be flexible and respond consistently and realistically to a wide range of actions that the player may take and events in any situation in the game. This section examines four possible scenarios within a strategy game world, focusing on the interactions between the objects and the environment. For each scenario, the performance of the EmerGEnT system and a conventional scripted system are compared and evaluated in terms of the observable behaviour of the objects, environment and the interactions between the two. The four scenarios are heat and fire, rain and water flow, pressure and explosions, and the integrated system including each of the previous components.

5.5.1 Scenario 1: Heat & Fire

Consider a situation where a wooden building, located in a forest, is set on fire (see Figure 5.1). Not only should the building burn and be damaged as a result, but the heat from the fire should also spread into the surrounding area. As the fire burns, the surrounding forest should also heat up and catch on fire, which will cause subsequent damage to the forest and the further spread of heat to surrounding areas. The heat and fire will continue to spread until the fuel is exhausted (there are no more trees) or until some other event causes the fire the stop (e.g. rain).

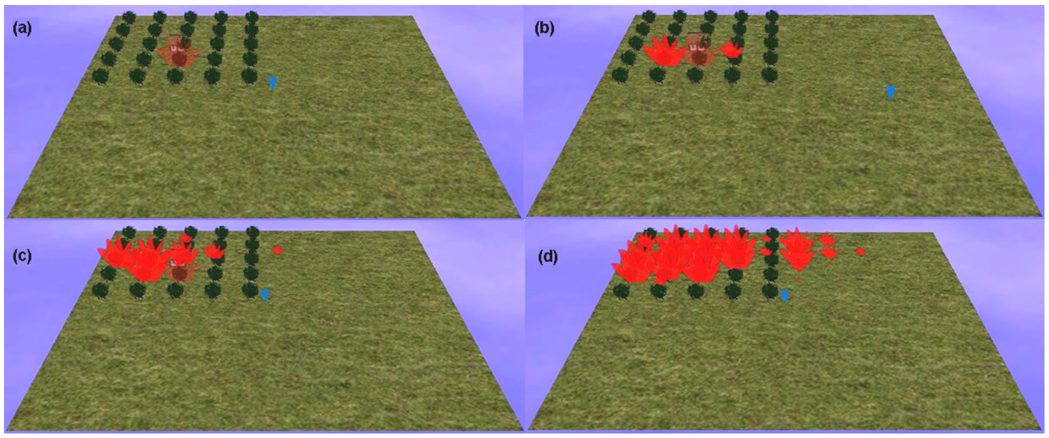


Figure 5.1. Heat and fire scenario in the EmerGEnT system. A wooden building in a forest is set on fire (a), the fire spreads to the forest (b-c) and the surrounding area (d)

In a system where the heat and fire scenario is scripted, the observable behaviour of the fire burning the building, then burning the surrounding area and eventually stopping would need to be specifically encoded. This would include specifying how long the fire will burn the building and each section of the terrain, how much damage will be caused by the fire in the process and what the fire will look like while it is burning. However, specifically scripting the observable behaviour of the heat and fire scenario means that the scenario will only behave in a pre-scripted and limited way. Any changes to the scenario would require the behaviour of the fire to be manually rescripted. For example, there could be more than one building, the building could be made of brick instead of wood and the surrounding terrain could be grass instead of forest. As the observable behaviour of burning brick is different than burning wood, the script would need to be updated to encode the new observable behaviour. The alternative would be to have a brick building burn in the same way as a wood building, which would not be as believable. The same problem would exist for the surrounding terrain. If the building were surrounded by grass rather than forest, the way the fire spreads from the building to the terrain and then through the terrain should be significantly different for grass than for forest, which would require further alterations to the script. Therefore, the major problems that exists for specifically scripting the heat and fire scenario is that there is considerable initial effort in implementing the scenario and small changes to the material of the terrain and objects requires the script to be rewritten or the same behaviour to be used for different scenarios.

On the other hand, when the heat and fire scenario is implemented using the EmerGEnT system, the resulting behaviour depends on the underlying properties of the materials of the cells and objects, which are used by the equations for heat and fire to determine the behaviour of the system dynamically. As such, the system is robust to changes in the heat and fire scenario, such as the number and position of buildings, the material of the buildings and the material of the surrounding terrain. Whereas a scripted system requires the observable behaviour to be encoded directly, the EmerGEnT system works at a lower level so that the observable behaviour of the system is emergent. Emergent behaviour is possible because cells and objects are subject to the same rules of heat and fire and their low-level properties and materials determine how they are affected by the system's rules.

5.5.2 Scenario 2: Fluid & Wetness

The second scenario presents a case study for fluid and wetness, in which rain falls on a hillside and the water from the rain runs down the hill into the valley below (see Figure 5.2). Subsequently, the water pools in the valley and floods the village that is located in the valley. In the village there are wooden houses, which absorb water from the flood. There is also a factory made of metal, which absorbs much less water from the flood. Some of the buildings are closed up, but others are open become flooded. Eventually, the rain stops and over time the flood water and buildings dry out.

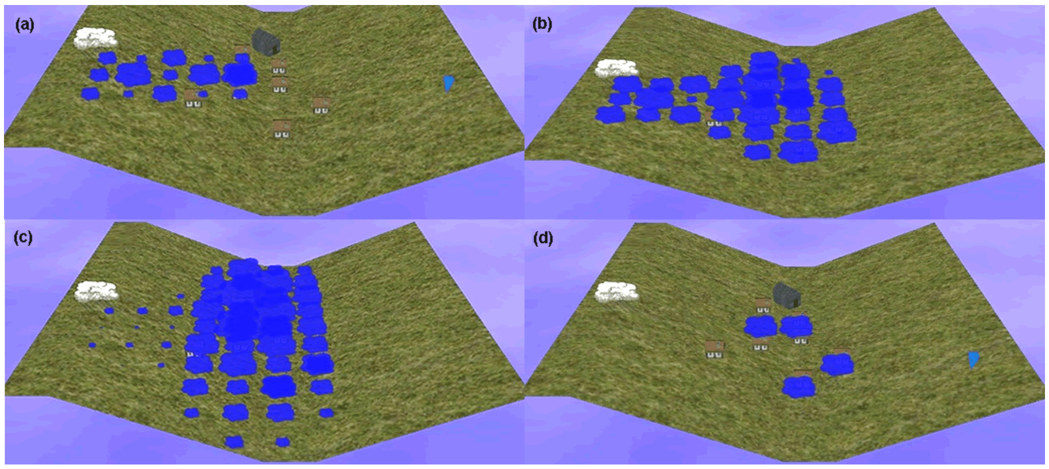


Figure 5.2. Fluid and wetness scenario in the EmerGEnT system. Rains falls on a hillside (a) and runs into the valley below (b). The houses that are open are flooded (c) and remain flooded after the water has dried up (d)

Scripting the fluid and wetness scenario would require many details to be considered. The water must follow a certain path to flow down the hill into the valley, the water will accumulate in the valley to cause a flood, the flood water will soak and flow into the buildings in a certain way and the water will dry out. For the fluid and wetness scenario, the observable behaviour of each component would need to be scripted, paying careful attention to detail. Due to the complexity of the fluid and wetness scenario, the resulting behaviour would be very specific to the situation. For example, the contours of the hill, the location of the town or the position of the rain would each greatly change the observable flowing of the water. If the town were located on the side of the hill, if the rain fell on the other side of the hill or if the contours of the hill forced the water to flow in a different direction then the town would not flood. Additionally, when the water does flood the town, the way it interacts with each type of building is different and this behaviour would need to be specifically scripted. If the buildings were different, in terms of material or affordances (such as whether they're open or not), then the behaviour of the water interacting with the buildings would need to be rescripted. Therefore, there are several variables that affect the observable behaviour in the fluid and wetness scenario that require careful attention to detail and considerable initial effort in implementing. Additionally, small changes to any of the variables would give rise to significant changes in the observable behaviour of the system, requiring considerable effort in maintenance and updating.

In contrast to scripted systems, the complex observable behaviour of the fluid and wetness scenario simply emerged in the EmerGEnT system as a result of the simple, low-level rules interacting with each other. There are three main components of the EmerGEnT system that allow the behaviour in fluid and wetness scenario to be emergent. First, as with the previous scenario for fire and heat, objects and cells have common low-level properties and materials and are subject to the same rules, which allow complex, high-level behaviour to emerge. The low-level properties allow the varying materials of the objects and terrain to absorb water differently. Second, objects in the EmerGEnT system have affordances so that structurally different objects behave differently. The high-level properties allow water to interact differently with buildings depending on their structure (e.g. water can only flow into buildings that are open). Third, in the EmerGEnT system, the way that water flows over the terrain is determined by the contours. Consequently, in the fluid and wetness scenario, the water flow over the terrain is dynamically determined by where the rain falls and the contour of the terrain, which may or may not cause the town to flood. Therefore, the EmerGEnT system allows the behaviour in the fluid and wetness scenario to be dynamic and emergent due to the properties of the cells and objects, the affordances of the objects and the contours of the terrain, which all contribute to the complex, high-level behaviour that is observable to the player.

5.5.3 Scenario 3: Pressure & Explosions

The third scenario presents a case study with pressure and explosions, in which a high pressure object is placed in an area with significantly lower pressure, located in a village (see Figure 5.3). The object explodes and the pressure in the surrounding area is immediately increased. As a result of the high absolute pressure, the surrounding buildings are damaged. Some buildings are more damaged than others, depending on their material. After the immediate release of pressure, the pressure flows out from the explosion into surroundings buildings and into the surrounding area.

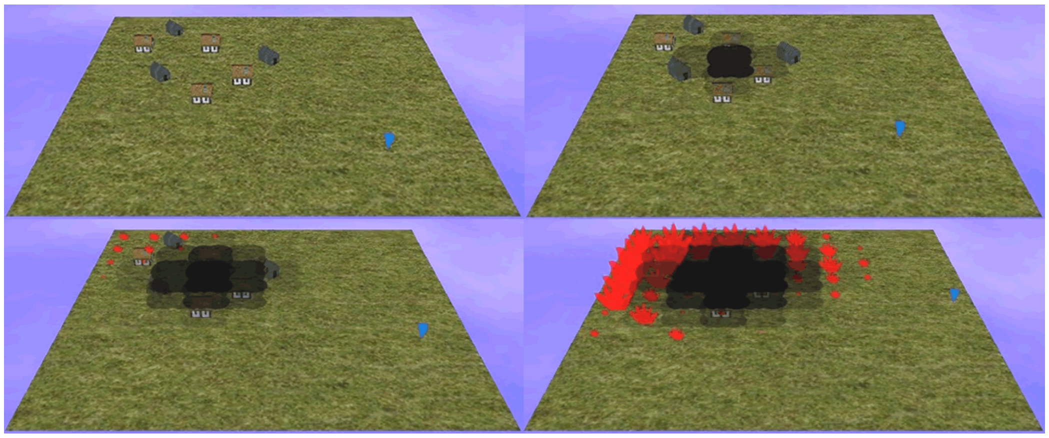


Figure 5.3. Pressure and explosions scenario in the EmerGEnT system. A high pressure object is placed in an area with significantly lower pressure (a). The effects of three different sized explosions are shown (b-d)

The factors that need to be considered when implementing the pressure and explosions scenario include the size of the explosion, the trigger for the explosion, the spread of pressure, damage to objects and cells and any secondary effects of the explosion. In a system that is specifically scripted, each of these factors needs to be specified and hard-coded. The size of the explosion will determine the area that the explosion effects and how much damage is done. If the size of the explosion varies then the variables need to be manually adjusted accordingly. In a scripted system, there would need to be a scripted trigger for an explosion to occur, such as a grenade being thrown. Additionally, any secondary effects of the explosion, such as a fire starting, would need to be hard-coded. Finally, the effect that the explosion has on different objects and terrain would need to be scripted. For example, more damage would be done to a tin can than to a lead barrel and this behaviour would need to be specifically scripted.

In the EmerGEnT system, the rules for explosions and pressure flow allow any changes in the system to be automatically accommodated. For example, larger explosions cause more heat to be released from the explosion, automatically causing a larger area to be effected and more damage to be done to nearby objects and cells. Additionally, events do not need to be scripted to trigger explosions (although they can be), as an explosion will naturally occur under the right conditions. Similarly, secondary effects, such as the release of heat and subsequent fires occur naturally as a result of the explosion, rather than needing to be scripted. Also, the flow of pressure between the objects and environment occurs dynamically, due to the affordances of the objects. Finally, objects and cells made of different materials are affected by the explosion to varying degrees, depending on their low-level properties. Therefore, the EmerGEnT system can respond dynamically in pressure and explosions scenario due to the affordances of objects (determining how pressure will flow between objects and cells), the low-level properties of objects and cells (determining how they will be affected by explosions) and the global rules of the system. The global rules of the EmerGEnT system allow the explosion and secondary effects to occur naturally, as well as variations to the pressure and explosions scenario (e.g. size of the explosion) to be automatically accommodated.

5.5.4 Scenario 4: Integrated System: Heat, Pressure & Fluid

The fourth scenario is a combination of the previous three scenarios. Rain falls on a hillside and runs down the hill, causing a flood in a village that is located in the valley below. As the flood washes through the village, the buildings made of wood absorb fluid, as does the grassland on the hillside and in the valley. Immediately afterwards, a small explosion occurs as a result of a high pressure object in the village. The surrounding buildings in the village are damaged, but due to their wetness they are difficult to ignite. The wooden buildings catch on fire and burn briefly and the fire spreads to the forest on the hillside on the other side of the valley (see Figure 5.4).

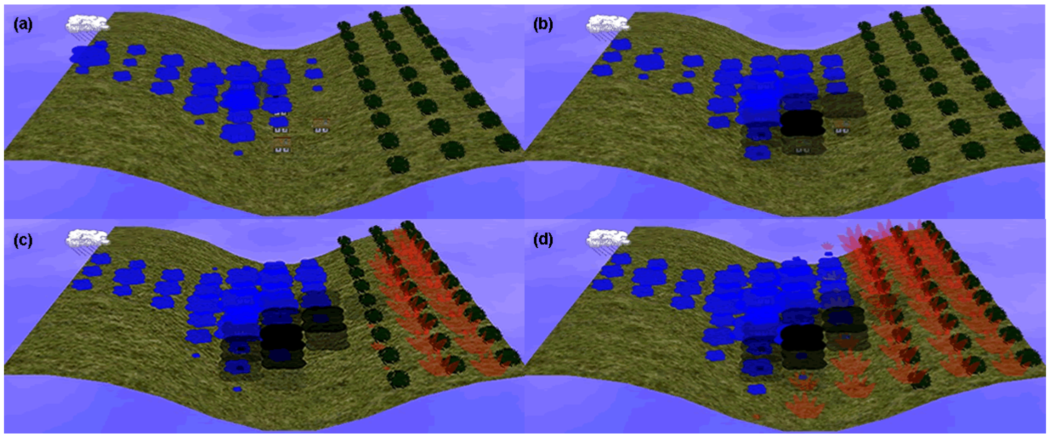


Figure 5.4. Integrated scenario in the EmerGEnT system. Rain falls on a hillside and causes a flood in the valley below (a). An explosion occurs in the village (b), damaging the buildings and causing a fire that spreads to a nearby forest (c-d)

The integrated scenario combines the attributes of each of the previous scenarios. The same problems exist for the specifically scripted system as in each individual scenario, but to a far greater degree due to the increase in complexity in the integrated scenario. There are many more variables in the integrated scenario, including the number, composition, structure and placement of buildings, the composition, contours and layout of the terrain, as well as attributes of the flood, explosion and fire. Changes that are made to any of these variables present complications for every aspect of the observable behaviour, including water flow, fire, explosions and damage, giving rise to a problem of combinatorial complexity. Similarly, the increased complexity of the integrated scenario has an impact on the performance of the EmerGEnT system. However, this impact increases the value and visibility of the emergence of the EmerGEnT system. In the integrated scenario, the rules of each of the systems of heat, pressure and fluid, as well as wetness, fire and explosions, combine with the affordances of the objects, the low-level properties of the objects and cells and the contours of the terrain, so that many simple, local interactions occur simultaneously to give rise to a living, complex environment with realistic, global, observable behaviour.

5.6 Discussion and Conclusions

The goal of the reported study was to design and integrate game objects into the active game environment of the EmerGEnT system discussed in Chapter 4 and to determine the extent to which property-based objects and the integrated system can facilitate emergent behaviour and gameplay. The approach taken was to implement objects as though they were cells, using the same low-level properties, based on the object's material. Subsequently, objects were also given high-level properties, based on their structure, to constrain the possible interactions of the object with the environment. The interactions that were modelled for objects were the same as for the cells in Chapter 4, including heat, pressure, fluid, fire, wetness and explosions. The key differences between the object-to-cell interactions developed in this study and the cell-to-cell interactions discussed in Chapter 4 were that objects only have one neighbour (its host cell) and objects have affordances (determined by high-level properties). In this study, high-level properties were related to an object's structure, whereas low-level properties were related to its composition (i.e. material). Structure was modelled at a high-level as modelling it at a low-level would result in impossible computational complexity for a computer game, with marginal improvement in behaviour. Composition was modelled at a low-level as modelling it at a high-level would substantially decrease the emergent capacity of the objects and remove the link to the game environment. For game objects in the EmerGEnT system, determining whether object properties were high or low level was clear cut. However, deciding on the appropriate level to model properties comes down to a tradeoff between computational complexity and the gain of modelling at a lower level. In some cases, it will be prohibitively expensive to model properties at a low level, in other cases there will be no gain of modelling properties at a low level, and other cases will not be as clear cut.

As with the environmental interactions in Chapter 4, each of the major systems designed in this study, heat, fluid and pressure, demonstrated various advantages over conventional scripted approaches for game object interactions. For heat and fire, the major problems that exist for specifically scripted systems is that changes to the material of objects and the terrain, as well as the number and placement of objects, requires the script to be rewritten. However, the model of heat and fire implemented in the EmerGEnT system uses the underlying properties of the materials of the objects and cells, as well as global rules for heat and fire, resulting in emergent high-level behaviour. Scripting the behaviour of fluid and wetness requires considerable initial effort and careful attention to detail due to the number of variables that affect the observable behaviour of the system, such as the material of objects and cells, structure of objects and contours of the terrain. However, the EmerGEnT system allows the behaviour of fluid and wetness to be emergent due to the global rules of the cellular automata, which dynamically process the low-level properties of cells and objects, the high-level properties of objects and the contours of the terrain. For pressure and explosions, the factors that are problematic in scripted systems include the size of the explosion, the trigger for the explosion, the spread of pressure, damage to objects and cells and any secondary effects of the explosion. In a scripted system, the cause and effect of each of these factors needs to be anticipated and specifically scripted. However, the EmerGEnT system can respond dynamically to pressure and explosions due to the affordances of objects (for pressure flow), the low-level properties of objects and cells (effect of explosions) and global rules of the system (secondary effects and variations to scenario). Finally, the integrated scenario illustrated that a complex system with many variables poses significant problems for scripted systems as changes to one variable in the system presents complications for every aspect of the observable behaviour of the system. However, in the EmerGEnT system, the rules of each of the systems of heat, pressure and fluid, as well as fire, wetness and explosions, combine with the affordances of the objects, the low-level properties of the objects and cells and the contours of the terrain, so that many simple, local interactions occur simultaneously to give rise to a living, complex environment with realistic, global, observable behaviour.

In conclusion, this study answered the questions of how can game objects with a property-based design be incorporated into an active game environment and to what extent can property-based game objects facilitate emergent behaviour and gameplay. First, objects with a property-based design can be incorporated into an active game environment by dividing their properties into two levels, high-level and low-level. An object's low-level properties are the same as the cells in the active environment and as such the objects are easily incorporated into an active environment as they are subject to the same rules. An object's high-level properties constrain which rules affect various types of objects, as different objects have different affordances, unlike cells of the environment, which are uniform in structure. Second, this study provided evidence that property-based objects, in conjunction with the active environment, can facilitate emergent behaviour and gameplay by means of the global rules of the system and the properties of the objects and cells. These aspects of the EmerGEnT system enable real-time, dynamic processing of the game environment and objects to generate emergent, high-level behaviour that was not specifically planned and scripted in advance. The game objects designed in this study address the problems associated with current scripted systems for game developers and game players. For game developers, the objects have general, global properties, simplifying design of objects, planning of interactions and subsequent implementation and testing. For the game players, the global properties of the objects give rise to intuitive, consistent, emergent interactions, rather than pre-scripted, specific gameplay. With the use of two-part property-based objects, within an active environment, game design can be simplified and gameplay can be enhanced in strategy games, as well as other game genres.

6
Reactive Agents

Agents are an important type of object in game worlds as they give the game life, story and atmosphere. Agents serve many different purposes and hold many different positions in games, which contributes to making the game world rich, interesting and complex. For example, strategy games include units (e.g. marines) that the player controls and role-playing games include agents that fill a wide range of different roles in society, from kings to cobolds. Game-players expect agents to behave intelligently by being cunning, flexible, unpredictable, challenging to play against and able to adapt and vary their strategies and responses (Sweetser, Johnson, Sweetser & Wiles, 2003[4]). However, players often find that agents in games are unintelligent and predictable (Sweetser et al, 2003). Furthermore, players believe that agents' actions and reactions in games should demonstrate an awareness of events in their immediate surroundings (Drennan, Viller & Wyeth, 2004). However, many games are proliferated with agents that do not demonstrate even a basic awareness of the situation around them. These agents often occupy the landscape as glorified pieces of scenery and behave in exactly the same way in any number of situations, ranging from rain to open gun fire. The questions that arise are how can agents that appear intelligent to the player by reacting sensibly to the game environment be incorporated into an active game environment and to what extent can reactive agents give rise to emergent behaviour and gameplay?

Various techniques that can be used for agents in games are reviewed in section 6.1 and influence maps are identified as a potential solution. The aim of the study

[4] A study independent of the research reported in this thesis.

presented in this chapter was to design, implement and test reactive agents in the EmerGEnT system (see Chapter 4 to 6) and to assess the extent to which the behaviour of these agents is appropriate, intelligent, realistic and emergent. The approach taken was to design, implement and test agents that use influence maps, in conjunction with the cellular automata in the EmerGEnT system, to dynamically respond to the environment. Three structured experiments were conducted to determine the design that would achieve the most appropriate agent behaviour, as indicated by criteria for efficiency, effectiveness and visible behaviour.

6.1 Reactive Agents in Current Games

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators (Russel & Norvig, 2003). Furthermore, an agent is considered autonomous if it relies on its own percepts, rather than the prior knowledge of its designer (Russel & Norvig, 2003). As discussed in Chapter 2, the agents in most games are hard-coded, relying heavily on the prior knowledge of their designers and little on their current situation. Many agents in games, such as units in strategy games and villagers in role-playing games, simply do not react to the environment in any way. This behaviour demonstrates a lack of situational awareness, which is an agent’s dynamic mental model of its operating environment and its place in it (Perla, Markowitz, Nofi, Weuve, Loughran & Stahl, 2000).

Situational awareness gives an agent a sense of what is happening in its current environment, what could happen next, what options there are for action and the possible outcomes of those actions. Situational awareness is the foundation for making decisions in complex operational environments. There are some games in which the agents demonstrate situational awareness by actively sensing and reacting to other agents in their environment. For example, the agents in Half-Life have sight and hearing and periodically “look” at and “listen” to the world (Leonard, 2003). Also, the game Thief: The Dark Project uses the same core concepts as Half-Life, but with a wider spectrum of states (Leonard, 2003). However, the agents in these games still fall into the category of hard-coded AI, as they are based on finite state machines that take into consideration what they can “see” and “hear” in the environment. The method used by these agents is to periodically run through a list of rules to determine whether they sense an opponent (see Figure 6.1). Another limitation is that these agents actively check to determine whether they can sense something periodically, whereas real vision and hearing arrive at the senses continuously (Leonard, 2003). Therefore, depending on the agents’ frequency of probing the environment, it is likely that events and actions will be missed.

```
Begin look
--Gather a list of entities within a specified distance
--For each entity found…
----If I want to look for them and
----If they are in my viewcone and
----If I can raycast from my eyes to their eyes then…
------If they are the player and
------If I have been told to not see the player until they see me and
------If they do not see me
--------End look
------Else
--------Set various signals depending on my relationship with the seen entity
End look
```

Figure 6.1. Core sensory logic in Half-Life. The agents in Half-Life periodically run through a list of rules to determine whether they sense an opponent (reproduced from Leonard, 2003, p. 2)

Another game that requires the agents to sense and react to information in the environment is The Sims. However, unlike Half-Life and Thief, the agents in The Sims constantly receive information from the environment. In The Sims, the AI is embedded in the objects in the environment, known as “Smart Terrain”. Each agent has various motivations and needs and each object in the terrain broadcasts how it can satisfy those needs (Woodcock, 2000). For example, a refrigerator broadcasts that it can satisfy hunger. When the agent takes the food from the refrigerator, the food broadcasts that it needs cooking and the microwave broadcasts that it can cook food. Consequently, the agent is guided from action to action by the environment (Woodcock, 2000). Therefore, the agents in The Sims are not hard-coded like the agents in Half-Life and Thief. Instead, their behaviour is autonomous and emergent, based on their current needs and their environment. Whereas the agents in Thief and Half-Life use a list of rules to determine their behaviour, the agents in The Sims use weighted sums to determine the best behaviour based on the current situation. These weighted sums are known as utility functions, which map states onto real numbers that describe the degree of happiness associated with each state (Russel & Norvig, 2003). Utility functions allow rational decisions to be made when the agent has conflicting goals or multiple goals.

Although the agents in each of these games are able to sense entities in the environment in some way, they are still unable to sense the state of the environment itself. The agents in Thief and Half-Life are limited to sensing other agents in the environment and the agents in The Sims are limited to sensing other agents and objects in the environment. These agents would still be unable to react to events and states of the environment such as rain, fire, gunfire and so on. The approach used in The Sims could be extended to incorporate the environment as well as the game objects. However, there are a finite number of objects in the environment and a finite number of ways to interact with these objects. When considering the environment itself and the possible events and states in the environment, the problem becomes infinitely more complex and the approach used in The Sims would become too computationally demanding and difficult to manage. Translating the approach used in The Sims to a game environment would require every element and event in the environment to project information to every agent in every location. Another approach that is more applicable to the problem of agents reacting to the game environment is a technique used in many strategy games, influence maps. A summary of influence maps related to games is provided in Sweetser (2004a), included on the CD.

6.1.1 Influence Maps

Influence maps divide the game map into a grid of cells, with multiple layers of cells that each contains different information about the game world. For example, the layers could store data for combat strength, vulnerable assets, area visibility, body count, resources or traversability (Tozour, 2001). The values for each cell in each layer are first calculated based on the current state of the game and then the values are propagated to nearby cells, thereby spreading the influence of each cell. This influence propagation gives a much more accurate picture of the current strategic situation, as it not only shows where the units are and what they're doing, but also what they might do and the areas they potentially influence (Tozour, 2001). Influence maps can be used for strategic assessment and decision making as their structure makes it possible to make intelligent inferences about the characteristics of different locations in the environment. For example, areas that have high strategic control can be identified, as well as weak spots in an opponent's defences, the enemy's front, flanks and rear, prime camping locations, strategically vulnerable areas, choke points on the terrain, and other meaningful features that human players would choose through experience or intuition (Tozour, 2001; Woodcock, 2002). Each layer, or set of layers, provides information about a different aspect of the game. For example, the influence map can indicate where the AI's forces are deployed, the location of the enemy, the location of the frontier, areas that are unexplored, areas where significant battles have occurred and areas where enemies are most likely to attack in the future (Tozour, 2001; Woodcock, 2002). Furthermore, when these layers are combined, they can be used to make strategic decisions about the game. For example, they can be used to make decisions about where to attack or defend, where to explore, and where to place assets for defence, resource-collection, unit-production and research.

Currently, influence maps are used in games for strategic, high-level decision-making. However, it would also be possible to use them for tactical, low-level decision-making, such as individual agents or units reacting to the environment. Similar to strategic influence maps, a tactical influence map would require values for the environmental factors that need to be considered and a method for combing these factors into values that can be used for decision-making. The factors that need to be considered by the agents include rain, fire, gunfire and any other events and effects that are relevant to their decision. The method to combine these factors would be the same as in strategic influence maps, a weighted sum that weights the factors in the environment depending on how important they are for the decision that is being made.

The advantage of using influence maps over methods that are currently used in games, such as Smart Terrain in The Sims, is that the agent is presented with a single value (calculated using the weighted sum to combine all the factors) instead of numerous messages being sent to the agent about the environment. Therefore, the agents can simply choose the best cell (based on the weighted sum) for the decision that it is making (e.g. where to move to avoid danger). Also, this approach has further advantages over the method used in games such as Half-Life and Thief as the agent is continuously adapting its behaviour to the environment (rather than probing at given time intervals) and its behaviour is a function of its environment (rather than following a prescribed set of rules). Finally, the influence map structure fits nicely with the cellular automata that are already being used to model the environment in the EmerGEnT system. Both use the same data structure and the raw values for the influence map are supplied by the calculations of the cellular automata. Therefore, the approach of using an influence map for tactical decision-making is investigated in this chapter as it accommodates passive sensing of a continuous environment (as opposed to discrete entities), allows the agents' situational awareness to evolve as a function of the environment, gives rise to reactive and emergent behaviour and combines well with the cellular automata model of the environment.

6.2 Agent Structure

In the EmerGEnT system, agents have an identical structure to the objects discussed in Chapter 5. Each agent has a set of properties that include coordinates (position in the cellular automata), temperature, pressure, fluid, mass, wetness and material. The materials that agents are composed of have the same properties as materials for terrain and objects, including flashpoint, burning temperature, specific heat capacity and maximum burning rate. In strategy games, there are a wide range of possible agent types, such as humans (e.g. marine), vehicles (e.g. tank), boats (e.g. fishing boat) and aircraft (e.g. B-52). Therefore, possible materials for agents include flesh, metal and wood (see Table 6.1). The values for each material were initially estimated then tuned until the materials burned at an acceptable rate and duration. Agents also have the same high-level properties as objects, which encode attributes of the agents' physical structure. These attributes determine whether agents will engage in various interactions (e.g. a human cannot be filled with water but a boat can).

Table 6.1. Agent material properties. Possible materials for agents include flesh, metal and wood

Property	***Description***	***Values***		
		Wood	*Metal*	*Flesh*
Flashpoint	Ignition temperature of the material	2000	5000	300
BurnTemp	Multiplier for the temperature the material burns at (amount of heat released)	5	10	3
BurnRate	Multiplier for the rate the material burns at (rate of consuming fuel)	10	5	15
MaxBurn	Maximum rate the material can burn at	300	100	200
SHC	Specific heat capacity – the amount of energy required to heat up this material	100	50	100
MaxFluid	The maximum amount of fluid the material can absorb	50	0	20
Strength	Modifier for the pressure the material can withstand before it breaks	0.6	1.0	0.5

6.3 Agent Design

The agents in the EmerGEnT system are identical to the objects in terms of structure and composition. The defining difference between agents and objects is that an object is simply acted upon by the environment and responds according to its physical composition and structure, whereas agents have a choice of how to respond to the environment. If an agent does not respond to the environment in a way that seems reasonable to the player (e.g. preserving its own safety), then they seem unrealistic and unintelligent. For example, if an object (e.g. a crate) is in a room that is on fire, the object's only course of action is to sit there and burn. However, an agent in the same situation would be expected to try and escape from the room in order to preserve its own life. If the agent were simply to stand in the room and burn then it would appear neither intelligent nor realistic. Therefore, it is necessary for an agent to have two things in order for it to react sensibly to the environment. First, it must have a way to sense the environment and second, it must have a way to choose a suitable reaction, based on what it has sensed.

As discussed in the previous section, an agent's understanding of its situation in the EmerGEnT system is represented as a weighted sum of the factors affecting each cell on the map. Based on the utility value of each cell, the agent chooses a cell to move to and reacts at a level that reflects its current situation (e.g. if the agent's current cell is on fire then it panics). After the agent chooses a destination, its task is simply to move towards it. This section discusses the "comfort" function that determines the utility of each cell, the agent's level of reaction and the agent's choice of destination cell.

6.3.1 Comfort Function

The utility function for the agents in the EmerGEnT system determines how comfortable each cell is for the agents and is therefore called a comfort function. The comfort function is a weighted sum of the factors that affect the agents' comfort in each cell and includes temperature, fire, pressure and wetness. Each of these factors is weighted according to how distressing it is for the agent. For human agents in the EmerGEnT system, fire is the most distressing, followed by temperature, pressure and wetness. However, these weights (*W1*, *W2*, *W3*, *W4*) can be tuned to reflect different priorities of different agents. For example, an alien might find water far more dangerous than heat. The comfort function returns a real value between zero and one, with a lower value representing a more comfortable cell.

```
Comfort = Min(((fire*W1) + (temp*W2) + (pressure*W3)
 + (wetness*W4)) ,1)
```

The comfort function provides an efficient alternative to the environment sending the agent multiple messages about its state, such as "it's hot" or "it's raining". Instead, the relevant factors are weighted and combined into a single value that gives the agent an estimate of the safety and comfort of its current location. The purpose of the comfort value is twofold. First, it provides a means for the agents to determine how comfortable they are in the current cell and to react accordingly. Second, it provides a means for the agents to assess surrounding cells and find a suitable destination. These two tasks are discussed in the following sections.

6.3.2 Level of Reaction

The comfort function returns a real value between zero and one, which allows the agent to react with varying degrees of distress, providing for more diverse and interesting behaviour (see Table 6.2). A comfort value of less than 0.1 represents a comfortable cell and the agent does not react. A comfort value of greater than 0.1 but less than 0.3 represents a cell that is uncomfortable and the agent reacts calmly and moves to a more comfortable cell. A comfort value of greater than 0.3 but less than

0.6 represents a cell that is distressing and the agent runs from the cell. Finally, a comfort value of greater than 0.6 represents a cell that is painful, which causes the agent to panic and run from the cell. The agent's level of reaction is denoted by its speed of movement, as well as its animation and sound. Scaling the agents' reactions allows the agents to react in varying ways to different situations, while greatly simplifying the process of determining how the agents will react. Instead of the agents considering each element in the environment individually, the comfort function determines the agents' level of discomfort and the agents respond accordingly by choosing the reaction level that corresponds to their comfort value.

Table 6.2. Agent reaction levels. Agents react with varying degrees of distress to provide more diverse behaviour

Value	*Level*	*Reaction*
< 0.1	comfortable	no reaction
0.1 - 0.3	uncomfortable	calmly moves to more comfortable cell
0.3 - 0.6	distressing	runs from the cell
> 0.6	painful	panics and runs quickly from cell

6.3.3 Choosing a Destination

If the agents are not comfortable in their current cell then they must locate and move to a more comfortable cell. Each agent reassesses its situation each timestep, by calculating the comfort value for the cell it is standing in or passing through and finding a destination cell based on the comfort of its neighbour cells. As long as the agent is not comfortable, it will keep reassessing its situation and finding a new destination, which means that agents can change destination while they are moving towards their current destination, if they find a better destination. Also, as the state of the environment is continuously changing, the destination the agent found last cycle may no longer be a comfortable cell. In choosing a destination, the agents evaluate the comfort values of the cells in a neighbourhood of a given size and choose the cell with the lowest comfort value.

6.4 Agent Experiments

Three experiments were conducted to investigate and tune the behaviour of the agents in the EmerGEnT system, in terms of efficiency, effectiveness and observable behaviour. The first experiment aimed to determine the most appropriate neighbourhood size that should be used by the agents when choosing a destination. The second experiment followed on from the first experiment and investigated combining different sized neighbourhoods to gain the benefits of both reactive and goal-directed behaviour. The third and final experiment involved combining desirability, in the form of a goal, with the comfort-based reactive behaviour of the agents. This section discusses the aims, methods, results and conclusions of each experiment.

6.4.1 Method

Several conditions were investigated in each experiment and ten trials with ten agents were run in each condition. Each experiment was conducted on a ten by ten grid of cells in the EmerGEnT system. The criteria that were used to evaluate the performance of the agents included whether or not the agents converged on a solution (i.e. agents located and reached comfortable cells), the number of cycles the EmerGEnT system ran before the agents converged, how efficiently the agents found a solution, what (if any) strategies or patterns agents exhibited and the number of local optima (comfortable cells) on which the agents converged.

The initial state of each trial was randomly generated, including the position of the agents, the position of rain and the number and position of explosions. After the trials were started, notes were made in relation to the criteria of efficiency, strategy and patterns and the EmerGEnT system was stopped as soon as the agents converged for the first time. At this point, the number of cycles the agents took to converge was noted, as was the number of optima that the agents converged on (an optima was considered to be a single cell). If the agents failed to converge (which usually occurred if the agents died before finding an optimum) then it was noted that the agents did not converge. A screen shot was taken of the final state of the EmerGEnT system in each trial, showing the state of the system, the number of cycles and the position of the agents.

6.4.2 Experiment 1: Determining Neighbourhood Size

The aim of the first experiment was to determine the optimal neighbourhood size, in terms of agent performance and behaviour, that agents should evaluate when choosing a destination (i.e. where to move to maximise comfort). Four conditions were investigated, including three conditions where the agents used neighbourhood sizes of one (n=1), two (n=2) and three (n=3) to choose destinations (see Figure 6.2). The final condition involved the evaluation of the entire grid to find a global optimum.

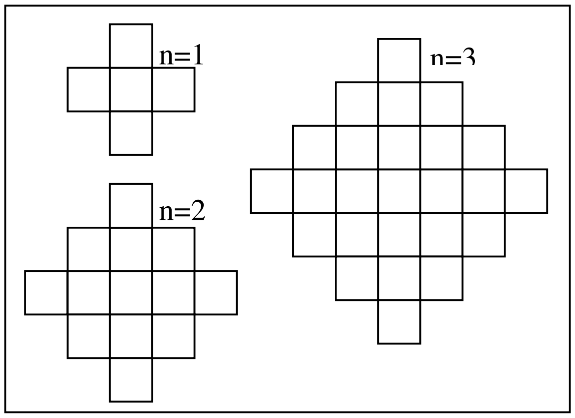


Figure 6.2. Neighbourhood sizes. Area that is evaluated for neighbourhood sizes n=1, 2 and 3

6.4.2.1 Results and Discussion

Each of the neighbourhood sizes that were evaluated in the first experiment had various advantages and drawbacks. The agents with a neighbourhood size of one performed the best at avoiding immediate danger. However, their short sight meant that they often ran towards more dangerous situations or became stuck in larger hazards as they were unable to find a way out. With a neighbourhood size of two, the agents were better at choosing safe destinations and appeared more organised, but still expressed the problems associated with short sight. The agents with a neighbourhood size of three were exceptional at picking particularly desirable cells and appeared organised as many agents moved to similar locations. However, the problems for these agents were almost the opposite of the previous agents, as they performed the best at choosing a destination but were unable to avoid immediate hazards in getting to their destination. They would often put themselves in great danger (e.g. run through fire) to get to a safe destination cell.

Moving through hazards to reach a safe goal was a far more severe problem when the agents were simply moving towards a global optimum. The agents would move across the whole map to get to their destination, rather than looking for a local optimum or a "good enough" cell in local proximity. The global optimum agents also took significantly more cycles to reach a goal cell than the agents in each of the previous conditions[5] (see Figure 6.3), as the global optimum was continually changing. Using a global optimum would be more reasonable if the goal cell was particularly desirable, such as fulfilling a mission or going home. The global optimum works well if the agents are in close proximity to the global optimum, but does not work at all if they are far away. Therefore, it would be much more logical for the agents to find a local optimum, as in the previous conditions where they were evaluating their local neighbourhood.

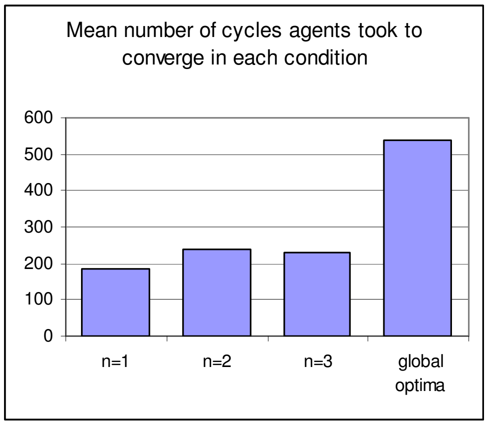


Figure 6.3. Finding a destination. The agents took significantly more cycles to converge in the global optimum condition than each of the local optima conditions

There were desirable traits expressed by the agents with short sight (avoiding immediate danger) and the agents with longer sight (finding a local optimum). As a result, a logical solution is to endeavour to combine these two approaches to enable the agents to move to a nearby, safe cell while avoiding hazards along the way. Consequently, the next experiment investigates how these two approaches can be combined and whether they yield an improvement in behaviour.

[5] n=1 (t=5.280, p<.001), n=2 (t=4.669, p<.001), n=3 (t=4.865, p<.001)

6.4.3 Experiment 2: Optimising Agent Navigation

The previous experiment demonstrated that neither purely reactive nor goal-directed behaviour was desirable for the agents in the EmerGEnT system. The reactive agents were able to avoid immediate danger but often ran into another hazard and the goal-directed agents chose a suitable goal but ran through hazards to reach it. Therefore, the aim of the second experiment was to determine if a combination of these two approaches is more effective than either approach individually and what combination of reactive and goal-directed behaviour is the most suitable for the EmerGEnT system.

The agents in the second experiment combined the reactive and goal-directed behaviours of the agents in the first experiment by first selecting a goal (the most comfortable cell) from the area around the agent (n=3). Second, the agents then evaluated the cells in their immediate neighbourhood (n=1) and chose which way to move, based on the comfort of the immediate cells and how close the immediate cells advanced the agent towards the goal. There were three conditions evaluated in the second experiment. The first condition weighted the comfort of each cell in the agent's immediate area with equal importance to the proximity of each cell to the agent's goal cell in the local area. The second condition weighted the proximity of the immediate cells to the goal cell more importantly (75%) and the comfort of the immediate cells less importantly (25%). Finally, the third condition weighted the proximity of the immediate cells to the goal cell less importantly (25%) and the comfort of the immediate cells more importantly (75%).

6.4.3.1 Results and Discussion

The agents in the second experiment displayed definite advantages over the agents in the first experiment. The agents in the evenly weighted and goal-directed conditions appeared far more intelligent, as they moved towards a goal rather than running back and forth randomly. Also, these agents appeared more realistic, as they moved around hazards on the way to their goal rather than simply running in a straight line, which made the agents in the previous experiment appear very flat and synthetic. Also, the agents in the evenly weighted condition displayed more depth as they did not always react in the same way, sometimes they would appear organised and at other times they would appear more independent, with their behaviour being heavily dependent on the current situation. The agents in the evenly weighted condition took the least amount of time to converge on safe cells (see Figure 6.4). The agents in the goal-directed condition behaved in a similar way to the agents in the evenly weighted condition, but became stuck more often and still ran through hazards. The agents in the reactive condition had the least desirable behaviour as they often appeared to move randomly, did not appear organised and often became stuck. Therefore, the second experiment suggests that the most suitable combination of reactive and goal-directed behaviour for the agents in the EmerGEnT system is approximately equal, where it is more desirable to err on the side of goal-directed than on reactive behaviour.

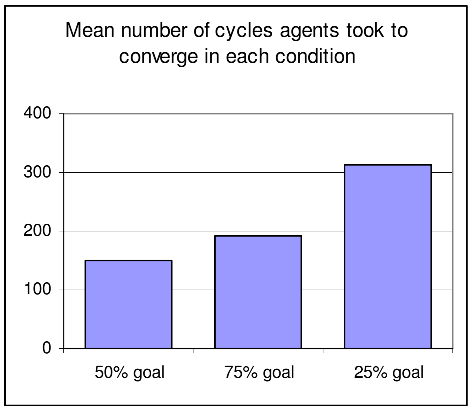


Figure 6.4. Optimising agent navigation. Mean number of cycles agents took to converge in Experiment 2

6.4.4 Experiment 3: Combining Comfort and Desire

The first and second experiments gave rise to agents that efficiently, intelligently and realistically react to the environment by moving from danger to safety. However, in a computer game situation, it is also likely that agents will have greater goals or desires that they need to fulfil, apart from simply surviving and reacting sensibly to the environment. For example, marines in a strategy game might be on a mission to kill the enemy in a particular cell or a villager in a role-playing game might want to stay near its house or shop. Drawing on the notion of "desirability" values from influence maps, goal areas could be given high desirability values for the agents. Additionally, desirability values could then be propagated out to surrounding areas to indicate that these areas are more desirable as they are near the goal. Therefore, the aim of the third experiment was to combine the desire to reach a greater goal with the agents' current behaviour of reacting to the environment and avoiding hazards. The third experiment combined an influence map to propagate the desirability of the cells with the cellular automata to determine the comfort of the cells. The question being investigated in the third experiment was what combination of desire and comfort gives the agents the optimal behaviour, in terms of avoiding hazards and reaching their goal.

The scenario for the third experiment was that ten agents have been given the order of getting to a single goal (e.g. marines sent to attack an enemy tank). The method used in the third experiment was to propagate the desire out from the goal position, with a propagation constant of 0.7 (i.e. the desirability value is multiplied by 0.7 for each step out from the goal). This value was chosen as it allows the influence to spread over the entire map of ten by ten cells, with a high concentration of desirability near the goal and low levels away from the goal. Figure 6.5 shows the desirability on a map with a single goal and desirability combined with comfort (darker is less desirable). Next, the agents calculated the comfort values for their local neighbourhood (n=3). Subsequently, the agents selected the best cell in this neighbourhood, based on the desirability value combined with the comfort value of each cell, which became their goal.

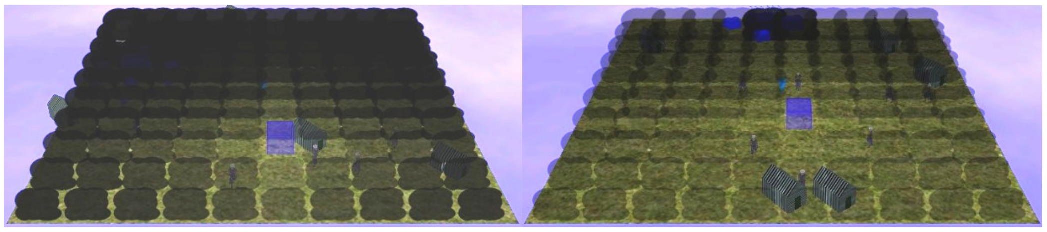


Figure 6.5. Desirability visualisation. Desirability on a ten by ten map with a single goal and propagation constant of 0.7 (left) and combined with comfort values (right)

The three conditions that were investigated in the third experiment were designed to test different influences of comfort and desirability on the agent's choice. The three conditions were evenly weighted (50% desirability – 50% comfort), goal-oriented (75% desirability – 25% comfort), and self-preserving (25% desirability – 75% comfort). After the agent has chosen the best cell in its local neighbourhood, based on comfort and desirability, it then chose which cell to move to in its immediate neighbourhood (as in Experiment 2). The best cell in the immediate neighbourhood was chosen based on its comfort value (50%) and its closeness to the chosen cell in the local neighbourhood (50%), which is the condition that was identified as optimal in the second experiment.

6.4.4.1 Results and Discussion

The first three conditions demonstrated that an equal weighting of desirability and comfort gave the agents the most acceptable observable behaviour, in terms of organisation, avoiding hazards and navigating the environment realistically and intelligently. The agents converged reasonably efficiently (Mean = 326 cycles), but only about half of the agents found the goal (Mean = 4.3 agents) as they opted for comfort over the goal.

When the weighting was tipped towards comfort or desirability, the agents' behaviour appeared random, less organised and less intelligent. The goal-directed agents converged in a reasonable period of time (Mean = 253 cycles), but only about half the agents found the goal (Mean = 4.2 agents). The self-preserving agents required significantly more cycles to converge (Mean = 420 cycles) than the agents in the previous conditions[6]. There was no significant difference between the number of agents that found the goal (Mean = 3.2 agents) in the self-preserving condition and in the previous conditions (see Figure 6.6).

[6] ($t=2.443$, $p<.05$)

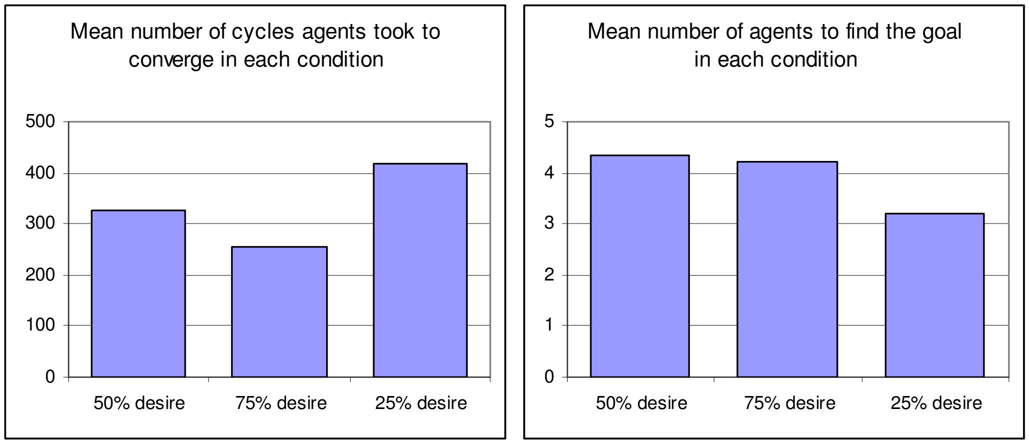


Figure 6.6. Combining comfort and desire. Mean number of agents to find the goal and mean number of cycles agents took to converge in each condition

Propagation Constant

The agents in each of the first three conditions were not very successful at finding the goal. Therefore, a fourth trial was run with equal weighting to optimise behaviour, but with a greater propagation constant to increase desirability values around the goal. Increasing the propagation constant (0.8) resulted in greater differentiation between cells on the influence map and further improvements in observable agent behaviour. The most noticeable change with a propagation constant of 0.8 was the improvement in the agents' behaviour, in terms of organisation, intelligence and rational behaviour. The agents were consistently able to move in organised and intelligent ways, exhibiting interesting and rich behaviour. In one situation, the agents were moving towards a goal that was blocked by rain and they waited for the rain to pass before moving towards the goal, rather than running through the rain or getting stuck. The increased differentiation between cells on the influence map provided the agents with a clear view of the best way to navigate through the environment. However, the agents were still not successfully able to find the goal.

Multiple Goals

A fifth and final condition was tested to determine how the agents would perform with multiple goals instead of one. The agents in the multiple goals condition were much better at finding the goals (Mean = 5.9 agents), but did not appear as intelligent or realistic as the agents in the previous condition. The agent were more likely to find the goals in this condition as there were more goals, but mostly because the desirability from the goals was cumulative, allowing more cells to have higher values and the influence to propagate further.

The third experiment produced an agent model that successfully integrates goal-directed behaviour (based on agent desires) with situation awareness (based on comfort), which enabled the agents to both react to the environment in an intelligent, realistic and organised way while simultaneously satisfying their desire to reach a goal.

6.4.5 Outcomes of Agent Experiments

The outcome of the first two experiments was a model for agents that dynamically respond to the environment in an intelligent and realistic way, based on concepts from cellular automata and influence maps. The outcome of the third experiment was an extension of this model that also integrates goal-directed behaviour to enable the agents to respond to the environment while pursuing a goal. An advantage of the model developed through these experiments is extensibility, in that it can be extended to incorporate any aspects in the game world that are relevant to the agents' behaviour (e.g. other agents, terrain, events). It would also be possible to incorporate other models for agent behaviour, such as flocking, so that the agents also take into consideration the movement of other agents around them. The simplicity and flexibility of this model means that it can be used to govern the behaviour of almost any agent in any circumstance. The contribution of this research is a design that allows agents to dynamically react to the changing situation of their environment, as well as an intelligent pathfinding algorithm that allows agents to find a safe path to a goal, based on aspects of their environment. A summary of the agent model that was developed and tuned through the experiments in the previous section can be seen in Figure 6.7 (see Appendix D for full pseudo-code).

First, the desirability of the goal is propagated by iterating through each cell on the map and setting the propagated influence value in each cell, depending on its *distance from the goal*, as well as the *goal's desirability* and *propagation constant*:

$$\text{Desirability} = \text{goal's desirability} * (\text{propagation constant})^{\text{distance from goal}}$$

In each cycle, each agent calculates the comfort value for each cell in its local neighbourhood (n=3). The *comfort* value is a weighted sum of the environmental factors in the cellular automata, including *temperature*, *pressure*, *fire* and *wetness*:

$$\text{Comfort} = \text{Min}(((\text{fire}*W1) + (\text{temp}*W2) + (\text{pressure}*W3) + (\text{wetness}*W4)), 1)$$

Subsequently, the agent finds the optimal cell in its local neighbourhood. The "*goodness*" of each cell in the local neighbourhood is the sum of 50 percent of the *comfort* of that cell and 50 percent of the *desirability* of that cell.

Goodness of local cell = 50% comfort + 50% desirability

The cell that the agent identifies as the optimal cell in its local neighbourhood becomes its goal. Subsequently, the agent assesses its immediate neighbourhood (n=1) to find the most optimal cell. The *"goodness"* of each cell is in its immediate neighbourhood is the sum of 50 percent of the *comfort* of that cell and 50 percent of the *proximity* of that cell to the agent's goal in the local neighbourhood.

Goodness of immediate cell = 50% comfort + 50% proximity to local goal

After the agent determines its destination cell in the immediate neighbourhood, it moves towards that cell at a pace dependent on its current comfort. Each cycle, the agent re-evaluates its local and immediate neighbourhood and updates its goal and destination.

Figure 6.7. Reactive agent model. A summary of the agent model developed and tuned through the agent experiments

6.5 Scripting versus Emergence

Introducing entities that have a choice of how to react to the situation amplified the variation and unpredictability of the system. Rather than the simple physical exchange that existed in the previous chapters, a new level of depth was introduced to the EmerGEnT system by agents that actively change their own state. Agents in the environment are not confined to the state of their current cell. Instead, they are free to react to the changing environment in ways that optimise values that are important to them, such as comfort and desirability. In doing so, the agents actively change the state of the environment, as they carry effects between cells in new and dynamic ways. For example, if a tank catches on fire and reacts by rolling into a group of trees, then those trees will in turn catch on fire, whereas the outcome would have been different without the active role of the agent. Furthermore, as the agents have the same set of physical and structural properties as objects and cells, different agents are also affected in varying ways by the same circumstances and subsequently react in different ways in these scenarios. This section examines four scenarios and discusses the contrasting outcomes and considerations for agents, objects, and environments in scripted and emergent systems with respect to these scenarios. The four scenarios are heat and fire, rain and water flow, pressure and explosions, and the integrated system including each of the previous components.

6.5.1 Scenario 1: Heat & Fire

The first scenario presents a case study for heat and fire, in which a fire is started in a military base that contains a variety of buildings (e.g. metal bunkers, wooden barracks) and agents (e.g. tanks, people). As discussed in the previous chapter, the interactions of fire with each of the different types of objects and terrain are emergent in the EmerGEnT system (see Figure 6.8). Similarly, the interactions of fire with the different types of agents in the scenario are emergent. The observable behaviour of the fire is emergent due to the high and low-level properties of the entities in the scenario, combined with the rules for the interactions of heat and fire. Therefore, the observable behaviour of fire in the heat and fire scenario depends on the position, composition, number and variety of objects, agents and terrain in the scenario. Furthermore, the behaviour and effects of the fire emerge and change as a function of the situation. Additionally, the agents in the heat and fire scenario react dynamically to the changing situation and in varying ways, depending on their low-level properties (a tank has a higher threshold for discomfort than a human) and high-level properties (a tank can move faster than a human and a boat cannot move on land). Finally, the ways that the agents react to the heat and fire scenario feedback to actively change the state of the scenario. For example, a tank that is on fire will propagate the fire as the tank moves into cells that the fire might not have reached otherwise.

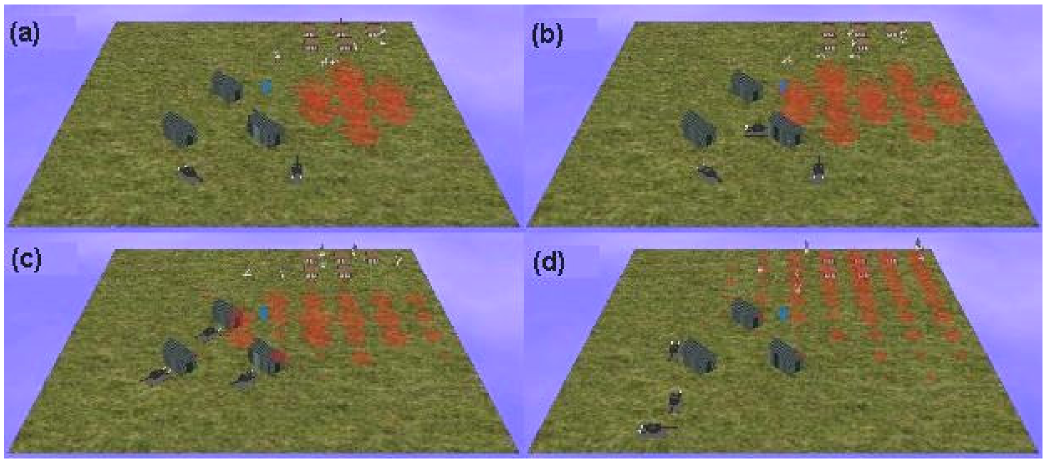


Figure 6.8. Heat and fire scenario in the EmerGEnT system. A fire is started in a military base (a) and spreads throughout the base (b-d) affecting different agents (tanks, people) and buildings (bunkers, barracks) in different ways

It would be difficult and time-consuming to script the observable behaviour of the fire, the effects of the fire on the agents, objects and terrain, as well as the resulting reactions of the agents to the situation, especially with any level of realism. As discussed in previous chapters, it is prohibitively complex to script the interactions of fire with game objects and environments to the level of detail implemented in the EmerGEnT system. Each attribute of the objects and terrain that need to be considered add a new level of complexity to the problem. Furthermore, the layout of the objects, agents and terrain in the heat and fire scenario magnifies the problem. Finally, when the reaction of the agents to the fire also needs to be considered (even without the agents' actions feeding back into the scenario), the problem becomes simply impossible. Therefore, the options for scripting fire in a game scenario are to implement flat, unrealistic, uniform fire or to spend tedious hours specifically scripting the multitude of combinations that are possible, which would fall far short of what is simulated in the EmerGEnT system due to time and logistical constraints.

6.5.2 Scenario 2: Fluid & Wetness

The second scenario presents a case study for fluid and wetness, in which rain falls on a hillside and runs down the hill and floods the village in the valley below. The village contains different types of buildings, villagers, vehicles and a boat. As discussed in the previous chapters, the interactions of water with different types of objects, agents and terrain are emergent in the EmerGEnT system (see Figure 6.9), as well as interactions of water with different terrain contours. The water from the rain runs down the hill and floods the valley, due to the emergent interactions of the water with the contours of the hill and valley. Also, some objects and agents (such as houses and cars) are filled with water, due to their high-level properties and other agents can float on the water (such as boats), due to their ability to move on water. Also, the different objects, agents and terrain become wet from the water to varying degrees, depending on their composition. For example, a wooden house absorbs more water than a metal shack, making it harder to ignite and more water-damaged. Again, small changes in the fluid and wetness scenario can give rise to significantly different observable behaviour. For example, changing the contours of the hill could mean that the village will not flood. Also, different agents respond to the water in different ways, as a flood is far more dangerous to a villager than it is to a boat.

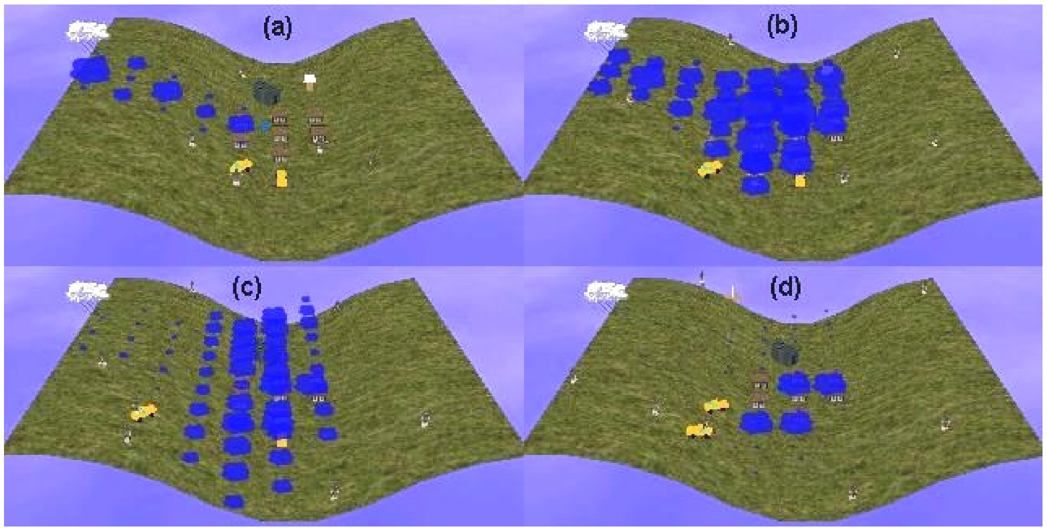


Figure 6.9. Fluid and wetness scenario in the EmerGEnT system. Rain runs down a hillside (a) and floods the village in the valley below (b-c) and dries out over time (d). The flood affects and interacts with the agents (trucks, people and a boat) in different ways

Again, scripting the behaviour of the fluid and wetness scenario specifically would be time-consuming, difficult and impractical. Changes to the contours of the terrain have a significant effect on the series of events in the scenario, by determining whether the water will flood the valley, whether it will pass through the valley and the magnitude of the flood. Subsequently, the various entities in the valley are affected by the flood water in varying ways, which would be prohibitively expensive to script specifically. Furthermore, the agents add another layer of complexity to the problem. Again, there are two possible approaches to scripting the fluid and wetness scenario. The first is to have a limited number of pre-scripted scenarios that can take place and the second is to attempt to script the scenario with some level of realism and flexibility, which is prohibitively time-consuming and complex.

6.5.3 Scenario 3: Pressure & Explosions

The third scenario presents a case study for pressure and explosions, in which a bomb explodes in a military base that contains different types of buildings (e.g. metal bunker, wooden barracks) and agents (e.g. people, tanks). As discussed in previous chapters, the observable effects of the interactions of the explosion (i.e. pressure) with the objects, agents and terrain in the pressure and explosions scenario are emergent in the EmerGEnT system (see Figure 6.10), as are the secondary effects of the explosion (e.g. fire started as a result of heat generated by explosions). The objects, agents and terrain are affected in varying ways by the explosion, depending on the strength of their composing material, their high-level properties (e.g. objects with volume can be filled with pressure), size and number of explosions, as well as the position, size, number and type of entities in the scenario. The result is a dynamic, emergent chain of interactions that arise from the rules for interaction, properties of the entities and the initial and evolving state of the scenario. Furthermore, the actions of the agents again add an extra layer of complexity to the EmerGEnT system. The agents react in varying ways to the pressure and explosions scenario and as a function of the changing state of the scenario. In doing so, they actively change the system state and propagate effects in ways that might not have occurred naturally.

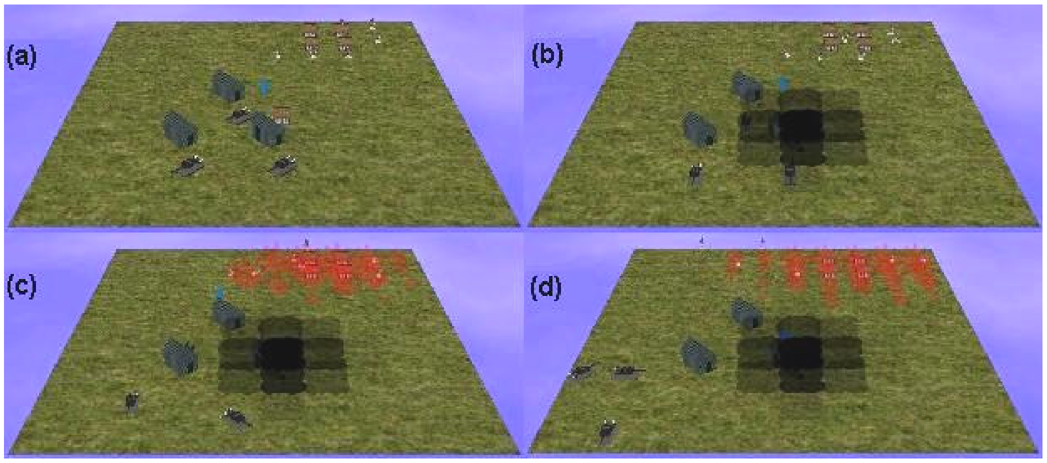


Figure 6.10. Pressure and explosions scenario in the EmerGEnT system. A bomb explodes in a military base (b), causing a fire (c) that spreads through the base (d)

The same problems exist for specifically scripting the pressure and explosions scenario as are discussed in the previous scenarios and in previous chapters. The EmerGEnT system takes into consideration many different factors, such as the position and magnitude of the explosion, proximity of game objects and agents to the explosion, high and low-level properties of objects, agents and terrain that result in varied effects from the explosion and so on. Again, the actions and reactions of the agents, which vary by type and situation of agents, add another layer of complexity to the problem. Therefore, the pressure and explosions scenario could be scripted to be executed in a prescribed way or the system could attempt to consider various attributes of the situation and run a precoded script. In either case, the script is preset, rigid and cannot be interacted with by the player or changed as a result of the situation.

6.5.4 Scenario 4: Integrated System: Heat, Fluid & Pressure

The fourth scenario is a combination of the previous three scenarios. Rain falls on a hillside and runs down the hill, flooding the village in the valley below. Subsequently, the flood washes away and an unrelated explosion occurs in the village. In the EmerGEnT system (see Figure 6.11), the initial flood accumulates in the valley as a result of the interactions of the water with the contours of the hill. The flood water then interacts with the buildings, vehicles and people in the village in varying ways, depending on their low and high-level properties. The flood washes away due to the contours of the landscape and also dries out over time. When the explosion occurs, the entities in the village are affected by the high pressure in different ways, depending on their composition. Also, under the right conditions, a fire starts in the village as a result of the heat released by the explosion. The fire then spreads through the village, affecting different buildings and agents in different ways, depending on their composition, their wetness from the flood, position, size and various other factors.

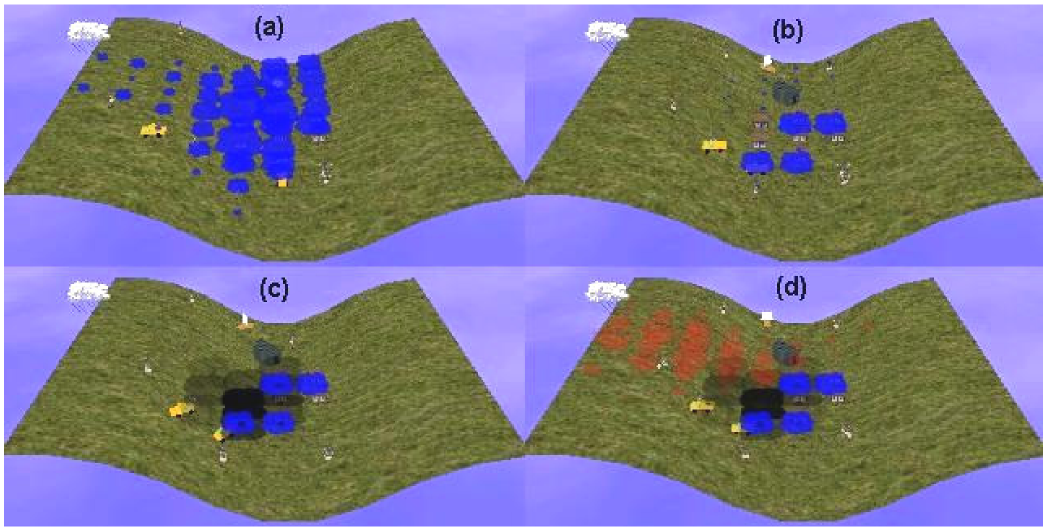


Figure 6.11. Integrated scenario in the EmerGEnT system. Rain runs down a hillside and floods a village (a) and dries out over time (b), followed by an explosion in the village (c) and a fire caused by the explosion (d)

As discussed in previous chapters, the complexity, power and flexibility of the EmerGEnT system becomes most apparent in the integrated scenario. The simple low-level rules of each system (pressure, heat and water) interact to give rise to interesting, rich and complex high-level, observable behaviour. The low-level properties of the agents, objects and terrain, the high-level properties of agents and objects and the contours of the terrain combine with the system rules to create a living, evolving and complex system. Changes to the initial and continuing state of the integrated scenario give rise to different outcomes and observable behaviour of the system.

Each event that occurs in the EmerGEnT system (rain, flood, explosion, fire) is not the result of pre-planning and scripting. Instead, each event occurs because the conditions of the integrated scenario are right for each of these events to happen, as would be the case in the real world. The rules of how things work in the EmerGEnT system (e.g. hot things burn, water flows downhill) combine and interact to mould the series of events that occur in integrated scenario. With the addition of the agents (entities that have a choice of how to react in a given situation), the complexity of the system is deepened further. As the system state changes, the agents choose how to react to their environment (affected by their high and low-level properties and goals), which in turn feeds back into the state of the system.

The interactions and actions of the agents give another level of complexity to the observable behaviour of the EmerGEnT system, bringing the environment to life with thinking, responding, interacting entities. A script to run the events described in the integrated scenario would need to be very specific to the scenario. Any change to the initial setup or any change to effects throughout the scenario would require multiple changes to be made to the script. It would not be possible to achieve the level of detail, interactivity and dynamic behaviour as is described in the integrated scenario in a specifically coded system.

6.6 Discussion and Conclusions

The aim of the study presented in this chapter was to design, implement and test reactive agents in the EmerGEnT system and to assess the extent to which the behaviour of these agents is appropriate, intelligent, realistic and emergent. The approach taken was to design, implement and test agents that use influence maps, in conjunction with the cellular automata in the EmerGEnT system, to dynamically respond to the environment. The resulting agent model works by first populating the agents' influence map with values based on the comfort of the cells and the agents' desire to move to a goal (if they have one). Subsequently, the agents find the best cell in the influence map in their local neighbourhood and decide which way to move (north, south, east or west), based on the comfort of the cell in each direction and the proximity of this cell to their local optima.

Three structured experiments were conducted to determine the design that would achieve optimal behaviour for the agents. The first experiment aimed to determine the appropriate neighbourhood size that agents should evaluate when choosing a destination. It was concluded that it would be desirable to combine the ability to find local optima of agents with larger neighbourhoods with the ability to avoid immediate threats of agents with smaller neighbourhoods. Consequently, the aim of the second experiment was to find the most appropriate combination of immediate area (reactive) and greater area (goal) evaluation. From the second experiment, it was concluded that an equal weighing of reactive and goal-directed behaviour gives rise to the most appropriate agent behaviour in the EmerGEnT system. Finally, the third experiment aimed to determine how well the agents perform with a goal, as many agents in games are required to do more than react to the environment. From the third experiment, it was demonstrated that optimal differentiation between cells is important to the success of an agent's behaviour.

The EmerGEnT system, including cells, objects and agents, exhibited the same advantages over scripted systems as were discussed in Chapters 4 and 5, as well as a new level of complexity and emergence added by the autonomous agents. As discussed in Chapters 4 and 5, the agents, objects and cells have low-level properties, related to their physical composition, which determine how they will interact with heat, pressure and water. Also, as discussed in Chapter 5, the agents and objects have high-level properties, related to their physical structure, which further constrain the ways in which they are able to interact with heat, pressure and water. Therefore, the interactions that occur in any given scenario are dependent on, and emerge as a function of, a variety of factors, including number, type and position of entities in the environment, terrain and external effects (e.g. rain, wind).

The addition of agents to the EmerGEnT system's complex and emergent environment adds another layer of complexity as the agents are able to actively change the state of the environment by choosing how to react to a given situation. Furthermore, the differences between individual and types of agents, such as composition, structure, goals, personality and so on, means that different agents will choose to react in different ways in the same situation. The result of each component of the system (objects, agents and cells) working together is a complex, rich and living world that provides a suitable environment for interesting and emergent gameplay to take place. The depth, complexity and life of the EmerGEnT system could not be replicated in a static system, based on scripting or finite states.

In conclusion, the agent study answered the questions of how agents that appear intelligent to the player, by reacting sensibly to the game environment, can be incorporated into an active game environment and to what extent these agents give rise to emergent behaviour and gameplay. First, reactive agents can be incorporated into an active environment by using cellular automata to provide the agents with a measure of comfort in their current situation, as well as a map for deciding where they might move to maximise their comfort. As this design closely resembles an influence map, it is also possible to integrate goal-directed behaviour and potentially personality, group movement and various other behaviours into the agent model. Within the active environment, the agents are simply extensions of the existing objects and are therefore subject to the same rules of interacting in the environment. Second, the agent study provided evidence that the reactive agents can further the emergent behaviour and gameplay discussed in previous chapters by adding a new level of complexity to the game world. As the agents are able to choose how to react to the environment, they are able to actively change the state of the world in ways that might not have occurred without their intervention. Whereas current agents in games do not demonstrate an awareness of their situation or react appropriately to events in their immediate surroundings, the reactive agents maintain a model of the comfort of their environment and react according to the changing state of their situation. The reactive agent model developed in this study allows agents to dynamically react to the changing situation of their environment and to intelligently find a path to a goal, increasing their visible level of intelligent, realistic and non-deterministic behaviour.

Part III

Evaluating the Facilitation of Player Enjoyment in the EmerGEnT System

7

Evaluation of Player Enjoyment

Player enjoyment is the single most important goal for computer games. The EmerGEnT system was developed to enhance player enjoyment by addressing the issues associated with interacting in and developing game worlds by facilitating emergent behaviour and gameplay in games. The EmerGEnT system has been evaluated to determine the extent to which it facilitates emergent behaviour and gameplay (see Chapters 4 to 6), but is yet to be evaluated in terms of player enjoyment. Therefore, the next step is to assess the extent to which the EmerGEnT system can potentially facilitate player enjoyment in comparison to conventional scripted systems and to identify the ways in which it can facilitate player enjoyment.

The aim of the study reported in this chapter was to evaluate the EmerGEnT system with respect to player enjoyment, which was accomplished by evaluating it against the GameFlow criteria (see Sweetser & Wyeth, in press, on the CD). GameFlow consists of eight elements – concentration, challenge, skills, control, clear goals, feedback, immersion and social interaction. Each element consists of a set of criteria for achieving enjoyment in games. GameFlow is an extension of flow (Csikszentmihalyi, 1990), a widely used model of enjoyment that is based on the idea that optimal experience or "flow" is the same the world over. Flow is an experience "so gratifying that people are willing to do it for its own sake, with little concern for what they will get out of it, even when it is difficult or dangerous" (Csikszentmihalyi, 1990). The EmerGEnT system was quantitatively evaluated against each of the criteria of GameFlow to determine the extent to which and how it can potentially facilitate enjoyment in games.

7.1 Enjoyment and Flow

Flow is based on Csikszentmihalyi's (1990) extensive research into what makes experiences enjoyable. Csikszentmihalyi's research consisted of long interviews, questionnaires and other data collection over a dozen years from several thousand respondents. Csikszentmihalyi started with people who spend large amounts of time and effort on activities that are difficult, but provide no external rewards (e.g. money or status), such as composers, chess players and rock climbers. Later studies were conducted with ordinary people with ordinary lives, asking them to describe how it felt when their lives were at their fullest and when what they did was most enjoyable. His research was conducted in many countries (e.g. USA, Korea, Japan, Thailand, Australia, Europe and on a Navajo reservation) and he found that optimal experience, or flow, is the same all over the world. He also found that very different activities are described in similar ways when they are being enjoyed and that enjoyment is the same irrespective of social class, age or gender.

Flow consists of eight elements:

- a task that can be completed
- ability to concentrate on the task
- concentration is possible because the task has clear goals
- concentration is possible because the task provides immediate feedback
- ability to exercise a sense of control over actions
- deep but effortless involvement that removes awareness of worries and frustrations of everyday life
- concern for self disappears but sense of self emerges stronger afterwards
- sense of duration of time is altered

The combination of these elements causes a sense of deep enjoyment that is so rewarding that people feel that expending a great deal of energy is worthwhile simply to be able to feel it (Csikszentmihalyi, 1990). Additionally, an important precursor of flow is a match between the person's perceived skills and the challenges associated with the task, with both being over a certain level.

Most flow experiences occur with activities that are goal-directed, bounded by rules and that require mental energy and the appropriate skills. For example, reading is one of the most frequently enjoyed activities throughout the world (Csikszentmihalyi, 1990). Reading has a goal and requires concentration of attention and that the reader knows the rules of the written language. The skills involved in reading are literacy, as well as the ability to turn words into images, empathise with fictional characters, recognise historical and cultural contexts, anticipate plot twists, critique and evaluate. Activities such as games, sports, art and literature have been developed throughout history for the express purpose of enriching life with enjoyable experiences (Csikszentmihalyi, 1990). The key element of flow is that the activity is an end in itself (or autotelic) – it must be intrinsically rewarding. This rings true in games because people play games (computer or other) for the experience itself, as there is no external reward. Finally, every flow activity provides a sense of discovery, a creative feeling of transporting the person into a new reality, which is a familiar concept for game players.

Flow has been applied extensively by researchers to assess enjoyment in a wide variety of domains. Previous applications of flow include workflow (Vass, Carroll & Shaffer, 2002), which was developed to support creativity in problem solving. Flow has also been used in a framework for constructing engaging ecommerce websites (Jennings, 2000), to assess enjoyment in an interactive music environment (Pachet & Addressi, 2004) and for assessing information systems (Artz, 1996).

7.2 GameFlow: A Model of Player Enjoyment

A comprehensive review of the literature on usability and user-experience in games was conducted to determine how the elements of flow are manifested in computer games (Sweetser & Wyeth, in press[7]). A model of enjoyment in games was constructed, based on the elements of flow and the evidence of flow experiences in

[7] A study independent of the research reported in this thesis.

games from the literature. The result was the development of the GameFlow model, which consists of eight core elements – concentration, challenge, skills, control, clear goals, feedback, immersion and social interaction. Each of these elements consists of a varying number of criteria and relate to Cziksentmilalyi's (1990) elements of flow as shown in Table 7.1.

Table 7.1. Elements of flow. The mapping of the elements from games literature to the elements of flow

Games Literature	*Flow*
The Game	A task that can be completed
Concentration	Ability to concentrate on the task
Challenge	Perceived skills should match challenges and both must
Player Skills	exceed a certain threshold
Control	Allowed to exercise a sense of control over actions
Clear goals	The task has clear goals
Feedback	The task provides immediate feedback
Immersion	Deep but effortless involvement, reduced concern for self and sense of time
Social Interaction	n/a

The first element of flow, a task that can be completed, is not represented directly in the GameFlow elements as it is the game itself. The remaining elements of GameFlow are all closely interrelated and interdependent. In summary, games need to keep the player's concentration through a high work load, but the tasks must be sufficiently challenging to be enjoyable. The player must be skilled enough to undertake the challenging tasks, the tasks must have clear goals so that the player can complete the tasks and the player must receive feedback on their progress towards completing the tasks. If the player is sufficiently skilled and the tasks have clear goals and feedback then they will feel a sense of control over the task. The resulting feeling for the player is total immersion or absorption in the game, which causes them to lose awareness of everyday life, lose concern for themselves and have an altered sense of time. The final element of player enjoyment, social interaction, does not map to the elements of flow but is featured highly in user-experience literature on games. People play games for interaction with other people, regardless of the task, and will even play games that they do not like or if they don't like games at all.

In this section, each element of flow is described and its manifestation in games is discussed. For each element, the GameFlow model includes an overall goal and a set of central criteria that can be used to design and evaluate games with respect to player enjoyment (see Table 7.2).

Table 7.2. GameFlow criteria for player enjoyment in games

Element	*Criteria*
Concentration Games should require concentration and the player should be able to concentrate on the game	- games should provide a lot of stimuli from different sources - games must provide stimuli that is worth attending to - games should quickly grab the player's attention and maintain their focus throughout the game - the player shouldn't be burdened with tasks that don't feel important - games should have a high workload, while still being appropriate for the player's perceptual, cognitive and memory limits - players should not be distracted from tasks that they want / need to concentrate on
Challenge Games should be sufficiently challenging and match the player's skill level	- challenges in games must match the player's skill level - games should provide different levels of challenge for different players - the level of challenge should increase as the player progresses through the game and increases their skill level - games should provide new challenges at an appropriate pace
Player Skills Games must support player skill development and mastery	- players should be able to start playing the game without reading the manual - learning the game should not be boring, it should be part of the fun - games should include online help so the player doesn't need to exit the game - players should be taught to play the game through tutorials or initial levels that feel like playing the game - games should increase player skills at an appropriate pace as they progress through the game - players should be rewarded appropriately for their effort and skill development - game interfaces and mechanics should be easy to learn and use
Control Players should feel a sense of control over their actions in the game	- players should feel a sense of control over their character or units and their movements and interactions in the game world - players should feel a sense of control over the game interface and input devices - players should feel a sense of control over the game shell (starting, stopping, saving etc) - players should not be able to make errors that are detrimental to the game and should be supported in recovering from errors - players should feel a sense of control and impact onto the game world (like their actions matter and they are shaping the game world) - players should feel a sense of control over the actions that they take and the strategies that they use and that they are free to play the game the way that they want (not simply discovering actions and strategies planned by the game developers)
Clear Goals Games should provide the player with clear goals at appropriate times	- overriding goals should be clear and presented early - intermediate goals should be clear and presented at appropriate times
Feedback Players must receive appropriate feedback at appropriate times	- players should receive feedback on their progress to their goals - players should receive immediate feedback on their actions - players should always know their status or score
Immersion Players should experience deep but effortless involvement in the game	- players should become less aware of their surroundings - players should become less self-aware and less worried about everyday life or self - players should experience an altered sense of time - players should feel emotionally involved in the game - players should feel viscerally involved in the game
Social Interaction Games should support and create opportunities for social interaction	- games should support competition and cooperation between players - games should support social interaction between players (chat etc) - games should support social communities inside and outside the game

7.2.1 Concentration

For a game to be enjoyable, it needs to require concentration and the player must be able to concentrate on the game (see Table 7.2). The more concentration a task requires, in terms of attention and workload, the greater the absorption in the task. When all of a person's relevant skills are needed to cope with the challenges of a situation, that person's attention is completely absorbed by the activity and no excess energy is left over to process anything other than the activity (Csikszentmihalyi, 1990). Games must quickly grab the player's attention and maintain their focus throughout the game (Pagulayan, Keeker, Wixon, Romero & Fuller, 2003; Lazzaro, 2004). Games can captivate player attention by providing something worth attending to (Brown & Cairns, 2004), such as detailed game worlds that draw the player into the game (Johnson & Wiles, 2003). It is important to increase the player's workload, while still being appropriate for the player's perceptual, cognitive and memory limits (Lazzaro & Keeker, 2004). Also, the player shouldn't be burdened with tasks that don't feel important (Fullerton, Swain & Hoffman, 2004). Finally, distractions from major game tasks during play should be minimised, by reducing non-game related interactions during play (e.g. setting options) and reducing the game interface to maximise the amount of screen taken up with game action occurring (Johnson & Wiles, 2003).

7.2.2 Challenge

Challenge is consistently identified as the most important aspect of good game design. Games should be sufficiently challenging, match the player's skill level, vary the difficulty level and keep an appropriate pace (see Table 7.2). An important precursor of flow is a match between the person's perceived skills and the challenges associated with an activity, with both skills and challenges being over a certain level (Johnson & Wiles, 2003; Sharafi, Hedmen & Montgomery, in press). If the challenges are greater than the skills then the result is anxiety and if the challenges are less than the skills the result is apathy (Johnson & Wiles, 2003). Games should have a variable difficulty level (Federoff, 2002) to meet all players with the correct level of challenge (Pagulayan et al, 2003). The difficulty level in games should also be varied, gradually increasing to maintain the interest of the player and provide more challenge as they develop mastery (Desurvire, Caplan, & Toth, 2004; Pagulayan et al, 2003). Pace is also an important aspect of challenge. The rate that players experience new game challenges and details can be paced to maintain appropriate levels of challenge and tension throughout the game (Pagulayan et al, 2003).

7.2.3 Player Skills

For games to be enjoyable, they must support player skill development and mastery (see Table 7.2). In order for the player to experience flow, their perceived skills must match the challenge provided by the game and both challenge and skills must exceed a certain threshold. Therefore, it is necessary that the player develops their skills at playing the game to truly enjoy the game. Players should be taught to play games through interesting and absorbing tutorials (Federoff, 2002) that allow the players to become involved quickly and easily (Desurvire et al, 2004). An alternative or accompaniment to tutorials is for players to learn as they play. When learning as they play, players learn and practice skills as part of accomplishing things they need and want to accomplish (Gee, 2004). Rewards are also an important part of learning to play a game. Players must be rewarded appropriately for continued play and the effort invested in a game should equal the rewards of success (Brown & Cairns, 2004).

Players should have enough information to start playing the game upon initially turning it on (Desurvire et al, 2004) and should not need or be expected to use a manual to play (Desurvire et al, 2004; Federoff, 2002; Gee, 2004). Games should also include online help so that the player doesn't need to stop playing the game to get help (Johnson & Wiles, 2003; Federoff, 2002). Players can also be given help in the form of hints (Federoff, 2002) or context sensitive help while playing (Desurvire et al, 2004), on demand or just in time (Gee, 2004; Sweetser & Dennis, 2003). Player learning can also be supported by games that are easy to use and learn.

7.2.4 Control

In order to experience flow, players must be allowed to exercise a sense of control over their actions (see Table 7.2). Players should be able to adequately translate their intentions into in-game behaviour (Pagulayan et al, 2003) and feel in control of the actual movements of their character and the manner in which they explore their environment (Federoff, 2002). The player should be able to move their character intricately, effectively and easily through the world and easily manipulate the world's objects, which become tools for carrying out the player's goals (Gee, 2004). Players should also feel a sense of control over the game interface and game controls. The game controls should be basic enough to learn quickly and the player should be able to customise the controls (Federoff, 2002, Adams, 2004). The game shell should be easy to use, allowing the player to start the kind of game that they want (Pagulayan et al, 2003), turn the game on and off (Desurvire et al, 2004) and save the game in different states. The player should not be able to make mistakes that stop the game from working (Adams, 2004) and games should help players to recognise, diagnose and recover from errors (Federoff, 2002).

It is important that players perceive a sense of impact onto the game world (Desurvire et al, 2004). Players should feel as though their actions and decisions are co-creating the world they are in and the experiences they are having (Gee, 2004). Players should feel a sense of control over their character (Desurvire et al, 2004) and be free to play games and solve problems in the way that they want (see Chapter 3). In short, the player should feel like they are playing the game, not being played by it (Kane, 2003).

7.2.5 Clear Goals

Game should provide the player with clear goals at appropriate times (see Table 7.2). Games must have an object or goal (Federoff, 2002), but to achieve flow these goals must also be clear (Csikszentmihalyi, 1990; Johnson & Wiles, 2003). Games should present the player with a clear overriding goal early in the game (Federoff, 2002), which is often done through an introductory cinematic that establishes the background story (Pagulayan et al, 2003). The goal should be conveyed to the player in a clear and straightforward way (Pagulayan et al, 2003). Also, each level should have multiple goals (Federoff, 2002) and games often use "briefings" to describe a "mission" that outlines immediate goals of the current part of the game and to suggest some of the obstacles that the player might face (Pagulayan et al, 2003).

7.2.6 Feedback

Players must receive appropriate feedback at appropriate times (see Table 7.2). During flow, concentration is possible because the task provides immediate feedback (Csikszentmihalyi, 1990). Games should use scores to tell players where they stand and players should always be able to identify their score and status in the game (Federoff, 2002). In-game interfaces and sound can be used to deliver necessary status feedback (Pagulayan et al, 2003; Federoff, 2002). Games should also provide immediate feedback for player actions (Desurvire et al, 2004; Johnson & Wiles, 2003).

7.2.7 Immersion

Players should experience deep but effortless involvement in a game (see Table 7.2). Immersion, engagement and absorption are concepts that are frequently discussed and highly important in game design and research. The element of flow that describes immersion is deep but effortless involvement, which can often result in loss of concern for self, everyday life and an altered sense of time (Csikszentmihalyi, 1990). Deep but effortless involvement is commonly reported by game-players and people who observe them (Johnson & Wiles, 2003). Players become less aware of their surroundings and less self-aware than previously (Brown & Cairns, 2004). Many game-players report devoting entire nights or weekends to playing games without being concurrently aware of doing so or consciously deciding to do so (Johnson & Wiles, 2003). Enjoyable games transport the player into a level of personal involvement emotionally and viscerally (Desurvire et al, 2004), drawing the player into the game and affecting their senses through elements such as audio and narrative (see Chapter 3).

7.2.8 Social Interaction

Games should support and create opportunities for social interaction (see Table 7.2). Social interaction is not an element of flow and can often even interrupt immersion in games as real people provide a link to the real world that can knock players out of their fantasy game worlds. However, it is clearly a strong element of enjoyment in games as people play games for social interaction, whether or not they like games or the game they are playing (Lazzaro, 2004). Therefore, social interaction is not a property of the task as are the other elements of flow, but the task is a means to allow social interaction. To support social interaction, games should create opportunities for player competition, cooperation and connection (Lazzaro, 2004; Pagulayan et al, 2003). Game experiences should be structured to enhance player to player interaction and should create enjoyment of playing with others inside and outside of the game (Lazzaro, 2004).

7.3 Evaluating EmerGEnT with GameFlow

The EmerGEnT system was evaluated with the GameFlow criteria. The aim of this evaluation was to determine to what extent and in what ways the EmerGEnT system facilitates player enjoyment in games. The EmerGEnT system was evaluated against each GameFlow element and the system's level of support for the criteria of each element was classified as "direct", "indirect" or "unsupported".

7.3.1 Concentration

Concentration is predominantly a game design issue, as it relates to the amount and variation of content that the game developer provides to attract the player's attention and maintain their concentration. The EmerGEnT system indirectly supports this criterion, however, by allowing more interactions with the game environment and creating more secondary effects to capture the player's attention (e.g. explosions can cause fires). Also, the use of the global approach to game design in the EmerGEnT system simplifies the creation of game content, making it easier for game developers to create more, varied content to populate the game world. In a conventional scripted system, the game developers would need to create each piece of content by hand, as well as the interactions between content, player interactions with content and game events. The need to plan and create everything in the game by hand can result in a reduced amount of game content in current games, giving the player less stimuli for concentration. The EmerGEnT system facilitates player enjoyment in terms of concentration by supporting game developers in creating content and allowing more interactions and effects to take place, which can give rise to more, varied stimuli that can maintain the player's focus and increase the player's workload (see Table 7.3).

Table 7.3. Concentration in EmerGEnT

Concentration Criteria	*Support*
- games should provide a lot of stimuli from different sources	Indirect
- games must provide stimuli that is worth attending to	Unsupported
- games should quickly grab the player's attention and maintain their focus throughout the game	Indirect
- the player shouldn't be burdened with tasks that don't feel important	Unsupported
- games should have a high workload, while still being appropriate for the player's perceptual, cognitive and memory limits	Indirect
- players should not be distracted from tasks that they want / need to concentrate on	Unsupported

7.3.2 Challenge

The challenge criteria are also predominantly dependant on game design as they are related to the tasks and obstacles provided by the game developers. However, the EmerGEnT system can also indirectly support challenge as the player has many more options for action and more potential strategies to use. The players are able to extend their own repertoire at their own pace by creating new strategies and modifying old strategies to increase performance or fit new situations. In contrast, conventional scripted systems often require the game developer to hard code specific strategies for the player to use to solve problems. Therefore, the challenge level of the strategy is preset, as well as the task. With the use of the EmerGEnT system, the game developers can focus more on the tasks, as the strategies emerge as a product of the environment. The EmerGEnT system indirectly supports each of the challenge criteria as the player has many more possibilities for strategies, it is more likely that the strategy will match their skill level as they have formulated it and the player can extend and refine their strategies as they become more skilled (see Table 7.4).

Table 7.4. Challenge in EmerGEnT

Challenge Criteria	*Support*
- challenges in games must match the player's skill level	Indirect
- games should provide different levels of challenge for different players	Indirect
- the level of challenge should increase as the player progresses through the game and increases their skill level	Indirect
- games should provide new challenges at an appropriate pace	Indirect

7.3.3 Player Skills

There are two sides to the player skills criteria, teaching the player to play the game and the player being able to intuitively play the game. Teaching the player to play the game is related to game design and is accomplished through tutorials, help, manuals and play. Therefore, the criteria related to teaching the player to play are not supported by the EmerGEnT system. However, the criteria related to the mechanics of the game being easy to learn and use and the intuitiveness of the game are directly supported by the EmerGEnT system. The mechanics of the EmerGEnT system are easy to learn and use as the game world has consistent rules and properties. As discussed in Chapter 2, consistent game worlds facilitate player learning as the player is better able to build a mental model of how the game world works. Additionally, the EmerGEnT system is easier to learn and use as the physical rules and properties resemble real world rules and properties. As discussed in Chapter 2, this correlation between the real and game worlds gives the player an inherent understanding of how the game world works. For example, the player knows that a forest will burn as wood is flammable. Similarly, the EmerGEnT system supports the player being able to play the game without reading the manual as the game world is intuitive. These criteria are only partially supported by the EmerGEnT system as the game interface is also an important component of being able to learn and use the game. In summary, the EmerGEnT system supports the player skills criteria of intuitiveness, learnability and usability due to the reflection of the real world and consistency in the game world rules (see Table 7.5).

Table 7.5. Player skills in EmerGEnT

Player Skills Criteria	*Support*
- players should be able to start playing the game without reading the manual	Direct
- learning the game should not be boring, it should be part of the fun	Unsupported
- games should include online help so the player doesn't need to exit the game	Unsupported
- players should be taught to play the game through tutorials or initial levels that feel like playing the game	Unsupported
- games should increase player skills at an appropriate pace as they progress through the game	Unsupported
- players should be rewarded appropriately for their effort and skill development	Unsupported
- game interfaces and mechanics should be easy to learn and use	Direct

7.3.4 Control

The control criteria that are related to the player feeling a sense of control over the game interface, input devices and game shell are not relevant to the game engine and are not supported by the EmerGEnT system. Similarly, the criterion that is related to the player not being able to make errors and being supported in recovering from errors is related to game design and is not supported by the EmerGEnT system. The criterion that relates to the player feeling a sense of control over their character and its movements and interactions in the game world is partially related to narrative, in terms of the control that the player has over defining their character and deciding their fate. Additionally, this criterion is partially related to the degree of control that the player has over the input devices that are used to control the player's character. Finally, this criterion is related to the game engine and mechanics, in terms of the degree of freedom the player has in performing the actions and interactions that they want and expect to be able to perform. The EmerGEnT system facilitates this criterion as it allows more interactions to take place and not just the interactions that have been predefined by the game developers. The interactions that are possible in the EmerGEnT system are dependent on the properties of the game elements and the rules for their interactions. Therefore, the more comprehensive the affordances and rules for interactions, the more freedom the player has in interacting in the way that they want.

The control criterion that relates to the player feeling a sense of control and impact onto the game world is partially related to the choices that the player is given and the decisions that they are required to make and the extent to which these decisions affect the game world, including narrative, characters and events. This criterion also relates to decisions on a smaller scale, such as the individual actions of the player having an effect on the game world. These small scale effects are facilitated by the EmerGEnT system as the player's actions have resulting reactions, which have secondary effects and so on. The EmerGEnT system facilitates a more reactive game world in which the reactions can have strategic (in the case of a strategy game) or significant impact. For example, burning down a forest costs the opponent a source of wood and compromises their defences.

The final control criterion is related to the player feeling a sense of control over the actions that they take and the strategies that they use and the player being free to play the game in the way that they want. This criterion can be seen as a continuation of the criterion relating to control of individual interactions and is supported by the EmerGEnT system. If the player is able to control individual interactions then they are able formulate and use their own strategies, due to the degree of reactivity of the game world and the freedom of interacting in the world. The EmerGEnT system facilitates control over strategies as the player does not discover strategies set by the game developer, they make their own strategies. Additionally, the strategies that the player devises and expects to work do not fail as a result of their attempted interactions not working or not being foreseen by the game developer. The EmerGEnT system's level of support for the control criteria is shown in Table 7.6.

Table 7.6. Control in EmerGEnT

Control Criteria	*Support*
- players should feel a sense of control over their character and its movements and interactions in the game world	Direct
- players should feel a sense of control over the game interface and input devices	Unsupported
- players should feel a sense of control over the game shell (starting, stopping, saving etc)	Unsupported
- players should not be able to make errors that are detrimental to the game and should be supported in recovering from errors	Unsupported
- players should feel a sense of control and impact onto the game world (like their actions matter and they are shaping the game world)	Direct
- players should feel a sense of control over the actions that they take and the strategies that they use and that they are free to play the game the way that they want (not simply discovering actions and strategies planned by the game developers)	Direct

7.3.5 Clear Goals

Goals are usually related to the game's narrative or specific conditions of victory. Therefore the clear goals criteria are not relevant to the game engine and are not supported by the EmerGEnT system (see Table 7.7).

Table 7.7. Clear goals in EmerGEnT

Clear Goals Criteria	*Support*
- overriding goals should be clear and presented early	Unsupported
- intermediate goals should be clear and presented at appropriate times	Unsupported

7.3.6 Feedback

Giving the player feedback on their progress and score is related to game design and is not supported by the EmerGEnT system. However, in the EmerGEnT system, the player's actions and interactions with the game environment and elements has various reactions, which provides implicit feedback to the player on their actions (see Table 7.8). In many scripted games, the game developers must manually encode the game world's reactions to the player's actions. Therefore, if no reactions have been encoded for a specific element for a specific action then the player's action will have no effect and hence no feedback is given to the player. This lack of reactivity may be deliberate for some game elements, such as pieces of scenery than cannot be interacted with. In this case, the player receives no feedback on whether their actions are working and why. Conversely, the lack of reactivity may not be deliberate and may occur because the developer missed a specific instance of a game element or specific reactions for that game element. This problem is a result of a specific approach to creating a game world and can be more damaging for the player's learning and immersion as it causes inconsistencies.

Table 7.8. Feedback in EmerGEnT

Feedback Criteria	*Support*
- players should receive feedback on their progress to their goals	Unsupported
- players should receive immediate feedback on their actions	Direct
- players should always know their status or score	Unsupported

7.3.7 Immersion

Immersion in games is usually associated with elements such as narrative, sound and graphics and is therefore not supported by the EmerGEnT system (see Table 7.9). However, immersion can occur as a result of deep concentration or involvement in a game. If the EmerGEnT system is used to facilitate concentration and other elements of GameFlow then it is likely that the player will become immersed in the game.

Additionally, it is difficult to evaluate the EmerGEnT system in terms of how it would facilitate immersion as immersion is heavily related to not only game design, but the game as a whole product. Therefore, even if the EmerGEnT system were successfully used to facilitate immersion, it is possible that other aspects of the game could prevent the player from becoming immersed.

Table 7.9. Immersion in EmerGEnT

Immersion Criteria	*Support*
- players should become less aware of their surroundings	Unsupported
- players should become less self-aware and less worried about everyday life or self	Unsupported
- players should feel emotionally involved in the game	Unsupported
- players should feel viscerally involved in the game	Unsupported

7.3.8 Social Interaction

The GameFlow social interaction criteria are different from the other elements of GameFlow as the game is no longer the task. The social interaction becomes the task and the game becomes the medium for the social interaction. As a result, most of the criteria for social interaction are not only separate to the game engine, but are separate to the gameplay as well, such as supporting game communities and social interaction outside of the game. The remaining social interaction criteria are related to supporting cooperation, competition and interaction between players through various gameplay mechanisms and are therefore not related to the game engine and not supported in the EmerGEnT system (see Table 7.10).

Table 7.10. Social interaction in EmerGEnT

Social Interaction Criteria	*Support*
- games should support competition and cooperation between players	Unsupported
- games should support social interaction between players (chat etc)	Unsupported
- games should support social communities inside and outside the game	Unsupported

7.4 Discussion and Conclusions

The aim of the study reported in this chapter was to evaluate the EmerGEnT system with respect to player enjoyment, to determine the extent to which and how it facilitates player enjoyment in games. The method used in this study was to conduct an evaluation of the EmerGEnT system using the criteria for GameFlow, an extension to the commonly accepted model of enjoyment, flow. The EmerGEnT system was evaluated against the criteria for each of the elements of GameFlow, including concentration, challenge, player skills, control, clear goals, feedback, immersion and social interaction.

Overall, the evaluation indicated that the EmerGEnT system directly supports six, indirectly supports seven and does not support 22 out of the 35 criteria of GameFlow (a summary is shown in Table 7.11). The EmerGEnT system directly supports criteria in player skills, control and feedback and indirectly supports criteria in concentration and challenge. The system does not support any of the criteria in clear goals, immersion and social interaction.

Table 7.11. GameFlow in EmerGEnT. The number of GameFlow criteria that were directly, indirectly or unsupported by the EmerGEnT system.

Element	*Direct*	*Indirect*	*Unsupported*
Concentration		3	3
Challenge		4	
Player Skills	2		5
Control	3		3
Clear Goals			2
Feedback	1		2
Immersion			4
Social Interaction			3
Total	6	7	22

The elements of concentration and challenge are indirectly supported by the EmerGEnT system. Concentration is predominantly a game design issue as it relates to the amount and variety of content provided by the game developer. However, the EmerGEnT system indirectly supports concentration by allowing more and a greater variety of interactions with the game world, creating more interactions and effects within the game world and simplifying content creation for game developers. Similarly, challenge is predominantly a game design issue as it relates to the tasks and obstacles provided by the game developer. However, the EmerGEnT system indirectly supports each of the challenge criteria as the player has many more possibilities for strategies, the player's strategy is more likely to match their skill level as they have formulated it themselves and the player can extend and refine their strategies as they become more skilled.

Components of the elements of player skills, control and feedback are directly supported by the EmerGEnT system. There are two sides to player skills, teaching the player to play the game and the game being easy to learn and use. The EmerGEnT system does not support teaching the player to play the game as this is a game design issue. However, it directly supports the game being easy to use and learn as the game rules reflect real world rules and are consistent. Control is partly associated with game design issues, such as the game interface, input devices, game shells, errors and narrative, which are not supported by the EmerGEnT system. However, the EmerGEnT system directly facilitates control as the player has more freedom in performing actions and interactions that they want and expect to be able to perform. Also, the player's actions have an impact and the player feels a sense of control over their actions and strategies and can play the way that they want. Giving the player feedback on their progress and score is related to game design and is not supported by the EmerGEnT system. However, the EmerGEnT system directly supports feedback as the game world immediately reacts to player actions, providing implicit feedback.

The elements of clear goals, immersion and social interaction are not supported by the EmerGEnT system. The clear goals criteria are not supported by the EmerGEnT as they are related to game design and rely on narrative and victory conditions. Immersion is not supported by the EmerGEnT system as it relates more to game design elements and is a product of a complete game system. However, it is possible that the EmerGEnT system could facilitate immersion through concentration. Finally, the social interaction criteria are not supported by the EmerGEnT system as they are related to game design, as well as player interactions outside the game world.

In conclusion, there is a clear split in the criteria between elements of player enjoyment that are related to game design and elements that are related to the game world. The EmerGEnT system is not a tool for creating narrative, interfaces, goals or tasks and therefore cannot directly support criteria that are related to game design, such as clear goals, immersion and social interaction. Conversely, the EmerGEnT system is able to directly facilitate elements of player enjoyment that relate to the game world and player interactions within the game world, such as player skills, control and feedback. Furthermore, the EmerGEnT system can indirectly facilitate aspects of player enjoyment related to game design that are dependent on interactions in the game world, such as concentration and challenge.

The EmerGEnT system directly or indirectly facilitates approximately one third of the criteria of player enjoyment, where conventional scripted systems provide no support other than what the game developer hard-codes. Consequently, the EmerGEnT system fulfils its aim of supporting player enjoyment through emergent behaviour and gameplay and has the potential to provide significant improvements over scripted systems. The importance of game design and gameplay was made apparent by the evaluation, including narrative, tasks, goals and interface. The creation of these elements mostly relies on the game developer’s creativity, design ability and thorough testing. However, further tools could be designed to assist in supporting or maximising these aspects of enjoyment, but these aspects are out of the scope of the EmerGEnT system. In short, the EmerGEnT system facilitates player enjoyment by supporting concentration, challenge, player skills, control and feedback, as it allows more intuitive, consistent and emergent interactions with the game world.

8 General Discussion and Conclusions

The goal of the research reported in this thesis was to investigate an emergent approach to designing game worlds and the resulting implications for game developers and the enjoyment of game players. The aims of this research were to define the issues associated with player enjoyment in current game worlds from the players' perspective, to investigate the potential and validity of using cellular automata to facilitate emergence in game worlds, and to determine how an emergent game system based on cellular automata affects developing and playing games. Multiple approaches were taken to achieve these aims, which resulted in the thesis being divided into three major components: (i) identifying the player-centred issues of interacting in game worlds, (ii) designing, implementing and testing the EmerGEnT system, and (iii) evaluating the facilitation of player enjoyment in the EmerGEnT system. This discussion argues the thesis that developing emergent game worlds based on cellular automata is valid and has the potential to provide an alternative to the current scripted approach and allow the development of more enjoyable games by facilitating emergent behaviour and gameplay.

8.1 Part I: Identifying the Player-Centred Issues of Interacting in Game Worlds

The player-centred studies (see Chapter 3) aimed to identify the aspects of current game worlds that affect player enjoyment from the players' perspective. Two studies were conducted, a focus group to identify and define the issues and a questionnaire to determine how the issues affect different groups of players. The outcome was a set of five issues that affect player enjoyment in game worlds: consistency, freedom of expression, intuitive interactions, immersion and physics. This research provides the first empirical evidence of the issues that affect player enjoyment when interacting in game worlds. Previously, the only knowledge of interacting in game worlds was in the form of anecdotal evidence from game developers. Considerations for the studies are that the results are only based on self-report and the samples were skewed towards experienced players. Further research is needed to delve deeper into the identified issues, particularly practical studies that do not rely solely on self-report and that examine different groups of players.

From the player-centred studies, it is apparent that gameplay (i.e. the player interacting with the game) is central to enjoyment and realism of the game. Four of the five issues (intuitiveness, consistency, freedom of expression and physics) identified in the player-centred studies relate to how the player interacts in the game world and how the game world reacts to the player. Each of these issues is inherent or supported in emergent game worlds, whereas they are unsupported or difficult to achieve in scripted games. Emergent games are inherently consistent and support game developers in providing greater freedom of expression, intuitive interactions and better physics. Conversely, scripted game worlds have many inconsistencies, limit player freedom and creating intuitive interactions and realistic physics depends solely on good design and manual effort. The remaining issue, immersion, relates to the scripted and designed elements of the game world, such as audio and narrative and is therefore the same in emergent and scripted games. It was evident from the player-centred studies that players are dissatisfied with many of the static, unintuitive, inconsistent and unrealistic elements of current scripted games and that they are looking for more interactivity, realism and control in game worlds. Consequently, the validity of striving to create emergent game worlds is supported by this research as emergence provides the opportunity to enhance player enjoyment by creating game worlds that allow consistency, freedom, intuitiveness and realistic physics.

8.2 Part II: Designing, Implementing and Testing the EmerGEnT System

The second stage of the project (see Chapters 4 to 6) aimed to investigate an emergent approach to designing game worlds, including environments (Chapter 4), objects (Chapter 5) and agents (Chapter 6). Each component of the EmerGEnT system (environment, objects and agents) was evaluated with four possible strategy game scenarios. One scenario was used to evaluate each of the major systems (heat, pressure and fluid flow). The final scenario was used to evaluate the integrated design, including each of the major systems. In each tested scenario, the EmerGEnT system displayed advantages related to its ability to dynamically determine and accommodate the specific state of the game world (e.g. number, type and position of entities, terrain and external effects), due to the underlying properties of the cells, objects and agents. The properties of materials allowed new materials to transfer heat and burn in reasonable ways that were not predetermined. The rules, height field of cells and affordances of objects and cells allowed fluid flow over contours and with object structures that were not predefined. Explosions occurred spontaneously (not triggered) and second order fire effects occurred spontaneously. Interactions occurred in cells by the rules of heat, pressure and fluid and these simple interactions summed to give emergent effects (e.g. it was not specified that water puts out fire, but water reduces heat and raises the flashpoint and therefore prevents or stops burning).

It was shown that an integrated game system design (including cells, objects and agents), based on cellular automata, can facilitate emergent environmental effects and complex behaviour in a limited domain. The EmerGEnT system presented in this thesis modelled heat, fire, rain, fluid flow, pressure and explosions in a game environment. It was shown that the use of a cellular automata, as well as property-based materials, objects, agents and rules can give rise to behaviour that is not specifically scripted into the system. Specifically, the behaviour of heat, fire, fluid flow, pressure and explosions was shown to respond dynamically to the changing game world and give rise to second order effects that were not directly specified. An important contribution was that the EmerGEnT system successfully modelled an emergent game world (i.e. the environment itself, as well as objects and agents), rather than emergent game objects only. Emergent worlds allow many more interactions and greater complexity than emergent objects alone.

8.2.1 Implications for Cellular Automata in Games

The EmerGEnT system constituted the first investigation of the application of cellular automata to real-time games. The research provided evidence that cellular automata are suitable algorithms to form the basis of emergent game worlds. The grid-based structure of cellular automata allows them to be easily integrated into game systems, which are often divided into grids. Also, the computational complexity of two-dimensional cellular automata proved appropriate for use in a game engine, such as the Auran Jet. However, the EmerGEnT system was not tested within a complete game and it is therefore not possible to conclude whether its computational complexity is acceptable for use within a game. It is likely that the limitations of a real game environment would require the EmerGEnT system to be significantly more efficient, which was outside the scope of this project.

The cellular automata used in the EmerGEnT system proved able to facilitate emergent behaviour of environmental systems (water, heat and pressure) and effects (explosions, fire) with cells of different terrain, objects of different material and structure, and reactive agents of different type, material and structure. Cellular automata were shown to be able to facilitate emergence in games in terms of the behaviour of an environmental system and the corresponding effects on cells, objects and agents. It was also shown that property-based objects and influence maps can be easily integrated with cellular automata to allow intelligent and reactive object behaviour and agent decision-making, as well as increasing the complexity and emergence of the game world.

8.2.2 Emergence as an Approach to Game Design

Various issues need to be considered when developing emergent game systems, including level of creative control for game developers, effort in designing, implementing and testing, effort in modifying and extending, issues related to uncertainty and quality assurance (knowing the system won’t break) and ease of feedback and direction to players. The EmerGEnT system studies (Chapters 4 to 6) allowed these issues to be examined in more detail, adding insight and depth to the current literature.

8.2.2.1 Level of Creative Control

Scripted systems give the game developer complete creative control, as the game developer decides what will happen and when. However, emergent systems allow a more approximate control, in that the game developer guides the player, providing boundaries for gameplay rather than dictating specifically what will happen. In an emergent system, the developer can set goals, but cannot specify how the player will get there. The EmerGEnT system studies did not investigate the level of creative control as the system did not include creative content. However, the method of developing rules and properties to approximate the desired behaviour provided insight into the issues that would be present when trying to control creative content. Getting a simple environment to behave in a desired way was difficult and involved significant tuning. With a full scale world, there would be sufficient problems in getting the world to behave reasonably, even without the added complication of controlling the narrative flow. Considerable future work is required to determine how narrative can be used effectively in emergent systems.

8.2.2.2 Effort in Designing, Implementing and Testing

As discussed throughout Chapters 4 to 6, both scripted and emergent systems have considerations for effort in development. The EmerGEnT system involved substantial initial effort in planning the rules and properties that would govern the behaviour of the system. It was difficult to decide how certain behaviour should be modelled and what rules and properties best capture the behaviour. Subsequently, the system required substantial testing and tuning to get the rules to generate behaviour that was desirable or acceptable.

Scripted systems also require considerable effort in planning, as well as implementing and testing. As discussed in Chapters 4 to 6, scripted systems involve every game element to be set up manually. Not only the goals need to be set, but the ways to achieve the goals must also be specified. In an emergent system, the game problems can be determined and the player can find their own solution. However, in a scripted system, the problem and solution must both be set and the player must find the developer's preset solution.

Both types of systems have considerations for the effort required in development, but the emergent approach would definitely be favourable as the size of the game grows and it becomes impossible to predict, plan and code everything. Games are now very large, in terms of the size of the worlds and the amount of content, and they will continue to grow. Scripting everything is already infeasible and some game developers have found that the initial outlay of effort to get an emergent game world working is a superior solution to creating the entire world manually (e.g. Valve's Half-Life 2). But so far, these games have been limited to game companies with extensive resources, experience and time, and only the objects (rather than the environments) have been emergent.

8.2.2.3 Effort in Modifying and Extending

The effort in modifying and extending was one of the major benefits of the EmerGEnT system, as well as emergent systems in general. The use of properties for different materials and objects made it simple to add new materials and objects to the system. The system dynamically retrieves the properties of the material or object and feeds them into the rules. Consequently, the system was data driven and adding new materials or objects and making changes to game scenarios did not involve changing any internal code or rules. Ease of modification and extension is a very important benefit as it allows developers to easily add more content, create expansion packs for their games (currently a big source of revenue for games), quickly release patches to fix bugs in the games and allow players to make modifications and create additional content. On the other hand, modifying and extending a scripted system is difficult and time-consuming. As mentioned previously, once the initial work was done in the EmerGEnT system, setting up new scenarios was a simple process that involved dropping in types of objects, agents and terrain and setting any desired events and letting the system run.

8.2.2.4 Uncertainty and Quality Assurance

Uncertainty is the predominant purpose of emergent systems, as it gives rise to new and unexpected behaviour. Ironically, it is also the main drawback of emergence in games. When human players are introduced into an emergent system, they have the ability to use the system in ways it was not meant to be used, change things in the game that the developer had not expected and play the game in ways that could not be foreseen. Furthermore, human players seem to have a perverse drive to intentionally push the game to its limits, exploit its weaknesses and to make it break. Consequently, game developers' fears of using emergent systems are justified, in that if a game has loopholes, exceptions or problems then the players will not only find them, but will actively seek them out and exploit them.

No unwanted uncertainty arose in the EmerGEnT system. However, it was a relatively small, controlled system and there were no humans interacting in the environment. The larger an emergent system (more entities, rules etc), the more complex it will become and the more variations in behaviour that will be exhibited. With the size of current games (e.g. role-playing games), the use of an emergent approach would require a different approach to game development. It would not be possible to predict every way the players will interact with the game and ensure that it works. Conducting extensive player testing would catch most of the problems and it would be necessary to focus on problems that will break the game or make it less fun.

8.2.2.5 Ease of Feedback and Direction to Players

Giving feedback and direction to players is simple in scripted systems. The player is simply moving through pieces of the game in a prescribed way. They are given goals, narrative and objectives and then play for a while. The player is led through a prescribed story, with pre-planned intervals of gameplay with very specific objectives. It is easy to give the players feedback and direction because it will be mostly the same for every player and the required feedback is easily identified as the possible and correct actions are known.

Giving feedback and direction is a much bigger problem in emergent games. In emergent game systems, the range of possible interactions and actions by the player is far more extensive and there is a lot more uncertainty. Consequently, the problem in giving feedback and direction in emergent games arises because players require a lot more feedback in these games, while at the same time feedback is much more difficult to give. This issue was not investigated in the EmerGEnT system as there were no players interacting and no goals or objectives.

8.2.3 Levels of Emergence – the Scripting-Emergence Continuum

In the literature review (Chapter 2), the idea of a continuum between scripted and emergent systems was proposed. Game systems do not need to be entirely scripted or completely emergent. There are many possible levels between these extremes. The balance between emergence and scripting will determine the degree of the creative control the game developer possesses and the level of freedom and variation for the player. A truly emergent system will have little to no creative control and complete freedom for the player. As discussed throughout this thesis, scripted systems provide complete creative control for the developer, but no freedom for the player. However, between these two extremes there are scripted systems with emergent elements (e.g. Half-Life 2), which allow much more creative control for the developers than in entirely emergent systems and greater freedom and variation for the player in interacting in the game world. This middle-ground provides a more balanced game in which the developer can tell a story, design challenges and tasks and maintain a reliable and enjoyable game, while the player still has freedom, control and variation.

The ways that emergence can be incorporated into games depends on the genre of the game and the level of creative control that is required. Emergence can be incorporated into games via emergent objects, agents or entire game worlds. Alternatively, game narrative could be made emergent (as a function of the player's interactions in the game world), as could conversations with game characters and game quests, objectives and puzzles. Role-playing games could include emergence in the form of characters that have general rules for behaviour, conversation and goals, rather than specifically scripted dialogue. First-person shooter games could include emergent objects, enemies and buildings. Finally, as shown in this thesis, strategy games can include emergent environmental effects, as well as active and reactive buildings, units and terrain.

8.2.4 Narrative in Emergent Game Worlds – A New Genre

It was proposed in the introduction (Chapter 1) that better games would result from allowing open, emergent gameplay within scripted narrative. As discussed in the previous section, emergent elements can be integrated into current scripted games to provide more flexibility and variation in gameplay, while still following a scripted narrative. However, the scripted structure of current games limits the extent to which these games can be made emergent. Although the gradual advent of emergence in games might result in enjoyable games in the short term, it is likely that there will be a need for fully emergent games in the future. The need for fully emergent games will necessitate the creation of a new game genre that will allow emergence and narrative to coexist in games.

The research discussed in this thesis shows that emergence in game worlds is possible. However, incorporating narrative into emergent game worlds is not simple. The game's storyline, goals and problems will need to be designed in a way that emergent gameplay won't violate the narrative flow of the game. Alternately, the narrative itself could be emergent and be generated by the gameplay, so that the player constructs their own story as they interact in the emergent game world. However, it would still be necessary to give the player goals and conditions for winning, otherwise it would not be a game and they would have no drive to keep playing. A game with emergent narrative would require very strong artificial intelligence as the story would be a product of the player's interactions in the world and with the game characters. Each character would need a comprehensive world model and extensive range of interactions with the player. It is almost certain that the use of emergent game worlds would involve the game developer letting go of some of their creative control and prohibit the use of linear storylines. There would also be problems involving the use of narrative elements, such as cutscenes and dialogue. It is also possible that game worlds will only ever be semi-emergent, holding with the current game model. However, the advent of fully emergent game worlds seems to be a natural progression of game development that will be driven by player demand.

8.3 Part III: Evaluating the Facilitation of Player Enjoyment in the EmerGEnT System

The final component of the project involved evaluating the EmerGEnT system against a set of criteria for player enjoyment in games. This evaluation allowed the EmerGEnT system's role in facilitating player enjoyment to be clearly defined, outlining exactly how and where the EmerGEnT system fits in designing games that players will enjoy. It was found that incorporating an emergent game environment, such as the EmerGEnT system, into a games engine would allow the direct or indirect facilitation of approximately one third of the criteria of player enjoyment in the GameFlow model. Conversely, conventional scripted systems provide no support for game developers to facilitate player enjoyment. The game developers must hard-code all interactions and scenarios manually, only getting out exactly what they put in.

The EmerGEnT system facilitates player enjoyment by supporting concentration, challenge, player skills, control and feedback, as it allows more intuitive, consistent and emergent interactions with the game environment. The system does not directly support criteria that are related to game design (clear goal, social interaction and immersion), but is able to directly facilitate elements of enjoyment that relate to the game environment and interactions within the game environment (player skills, control and feedback). Additionally, the system can indirectly facilitate aspects of player enjoyment related to game design that are dependent on interactions in the game world (concentration and challenge) by supporting game developers in creating emergent behaviour and gameplay.

As tests were not conducted with the system as part of a complete game, it was not possible to directly measure the impact that an emergent game world has on the issues of player enjoyment. However, the potential effect on players was indirectly measured by assessing the EmerGEnT system with respect to the affordances it would give game developers in creating game environments. Ultimately, the ideal evaluation of the EmerGEnT system in terms of player enjoyment would be to actually incorporate it into a game, so that player enjoyment could be measured directly from the players. However, at this early stage, the method used was appropriate and useful as it provided insight into how the EmerGEnT system could potentially facilitate player enjoyment in games.

8.3.1 Gameplay versus Game Design

The evaluation study identified which aspects of player enjoyment are related to game design (i.e. elements crafted by game developers) and which are related to gameplay (i.e. player interactions with and in the game world). The aspects of player enjoyment that are related to game design are not affected by the game world being emergent or scripted. Every element of player enjoyment is in some way related to game design. Clear goals, immersion and social interaction are solely related to game design, whereas concentration, challenge, control, player skills and feedback are only partly related to game design. Unlike the game design issues, the aspects of player enjoyment related to gameplay are affected by the game world being scripted or emergent. The aspects of player enjoyment that are related to gameplay can directly affect the player's enjoyment (player skills, control, feedback) or they can support the game developer in creating more enjoyable games (challenge, concentration).

The evaluation study not only shows that both game design and gameplay are highly important for player enjoyment, but that they are tightly interwoven. For a game to be exceptional, the game developer must design a game with story, conflict, detail, goals, problems and a good interface, and the player must be able to interact freely in this world, feeling central and influential. It is not possible, however, to conclude how much each element contributes to player enjoyment (e.g. is challenge more important than control?). Also, it is likely that each element affects individual players differently and that different game genres will manifest the elements in different ways.

Emergence definitely has a place within the current model of games, as it can enhance player enjoyment in elements where scripting is weak (e.g. giving the player more control). However, the game design (i.e. scripted) aspect of current games is central to player enjoyment and compromising on well-placed scripted elements is likely to reduce the enjoyability of a game. Therefore, the most enjoyable games in the near future will involve a good balance of scripting and emergence. However, it is possible that with the development of a new emergent game genre that the elements of player enjoyment will shift or be manifest in new, unforseen ways.

CONCLUSIONS

The EmerGEnT system serves as a proof of concept that emergent game worlds based on cellular automata can provide an alternative to currently scripted game worlds and facilitate the development of more enjoyable games, by giving rise to emergent behaviour and the potential to be incorporated into complete games to provide emergent gameplay.

Cellular automata can facilitate emergence in game worlds in terms of the behaviour of environmental systems and the corresponding effects on terrain, objects and agents. Also, games can have varying levels of emergence, which can be thought of as a scripting-emergence continuum. The most enjoyable games are likely to fall between the two extremes of the continuum. Emergence can also be incorporated into games in different ways. In this thesis, it was shown that game worlds can be emergent, including the environment, objects and agents. It was also theorised that game narrative could be emergent, in terms of interactions with characters or gameplay-generated narrative.

Players are dissatisfied with the static, unintuitive and unrealistic worlds in current games and emergence provides the opportunity to enhance player enjoyment with game worlds that allow consistency, freedom, intuitiveness and realistic physics. Player enjoyment is affected by both gameplay and narrative (or game design) and emergence has the potential to improve player enjoyment in terms of gameplay, by increasing player control, challenge, skills, concentration and feedback. As games incorporate more emergent components or become fully emergent, getting the narrative and emergence to work together will be a critical problem. It is likely that the most enjoyable games will result from carefully-designed, scripted game elements integrated seamlessly into emergent game worlds. The future of game development lies is finding the means to integrate and balance scripting and emergence and different combinations thereof to produce a variety of game genres that result in more valuable game playing experiences.

FURTHER WORK

The EmerGEnT system demonstrated the potential of emergent game worlds, within a limited domain and a controlled environment. The next step would be to incorporate an emergent system into a complete game environment. This would allow a better understanding of the limitations imposed on emergent systems in games (e.g. computational time available), the amount of tuning required and the complications that would arise from integrating an emergent system into a game. Additionally, once the emergent system is successfully incorporated into a complete game, it would be possible to introduce players into the system. This would serve two purposes: (1) the effect of the human factor on the emergent system could be gauged (e.g. how much complexity and uncertainty is introduced by human players?) and (2) the effect of emergent gameplay on player enjoyment could be directly measured.

Not only is it important to integrate an emergent system into a complete game, it is also necessary to test varying levels of emergence in games. As discussed, emergence is starting to make its way into games, such as emergent object behaviour or emergence through sheer world size and number of options. It was also discussed that there is likely to be a limit on the amount of emergence that can be incorporated into the current game model, without compromising on player enjoyment or sacrificing developers' creative control. Therefore, it is necessary to investigate different ways to incorporate emergence into games (e.g. through agents, environments, narrative), investigate different levels of scripting and emergence in games and the effects on developers and players, and find the limitations of emergence in games. The ultimate goal would be to determine how emergence can be best used in games to maximise player enjoyment and whether it will be necessary to develop a new game model that supports full emergence, while ensuring integrity and enjoyment.

References

Adams, E. (2004) The Designer's Notebook: Bad Game Designer, No Twinkie! *Gamasutra*, June 11, 2004. Retrieved online 1 February, 2005, at http://www.gamasutra.com/features/20040611/adams_01.shtml

Artz, J. (1996) Computers and the Quality of Life: Assessing Flow in Information Systems. *Computers and Society* 26 (3), pp. 7-12.

Bar-Yam, Y. (1997) *Dynamics of Complex Systems*. Reading, Mass.: Addison-Wesley.

Barros, F., and Mendes, M. (1997) Forest fire modelling in the DELTA environment. *Simulation Practice and Theory* 5, pp. 185-197.

Berger, L. (2002) Scripting: Overview and Code Generation. In S. Rabin (Ed.), *AI Game Programming Wisdom*. Hingham, Mass.: Charles River Media, Inc., pp. 505-510.

Brown, E., and Cairns, P. (2004) A Grounded Investigation of Game Immersion. *Extended Abstracts of the 2004 Conference on Human Factors in Computing Systems*. New York, NY: ACM Press, pp. 1297-1300.

Church, D. (2002) Simulation, Emulation, and the Game Design/Development Process. Presented at the *Australian Game Developers Conference, Melbourne, Australia*, 6-8 December, 2002. Retrieved online 13 April, 2005, at http://www.agdc.com.au/about/archive_2002_schedule.php#Sunday

Consolini, G., and De Michelis, P. (2001) A revised forest-fire cellular automaton for the nonlinear dynamics of the Earth's magneotail. *Journal of Atmospheric and Solar-Terrestrial Physics* 63, pp. 1371-1377.

Csikszentmihalyi, M. (1990) *Flow: The Psychology of Optimal Experience*. New York: Harper Perennial.

Desurvire, H., Caplan, M., and Toth, J.A. (2004) Using Heuristics to Evaluate the Playability of Games. *Extended Abstracts of the 2004 Conference on Human Factors in Computing Systems*. New York, NY: ACM Press, pp. 1509-1512.

Drennan, P., Viller, S. and Wyeth, P. (2004) Engaging Game Characters: Informing Design with Player Perspectives. Entertainment Computing - ICEC 2004: Third International Conference, *Lecture Notes in Computer Science*, 3166, pp. 355-358.

Federoff, M. (2002) Heuristics and Usability Guidelines for the Creation and Evaluation of Fun in Video Games. Unpublished thesis, Indiana University, Bloomington. Retrieved online 1 February, 2005, at http://www.melissafederoff.com/thesis.html

Fedkiw, R., Stam, J., and Wann Jensen, H. (2001) Visual Simulation of Smoke. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 15-22.

Forsyth, T. (2002) Cellular Automata for Physical Modelling. In D. Treglia (Ed.), *Game Programming Gems 3*. Hingham, Mass.: Charles River Media, Inc., pp. 200-214.

Frasca, G. (2003) Ludologists love stories, too: notes from a debate that never took place. *Proceedings of Levelup 2003*, DIGRA, pp. 92-99.

Fullerton, T., Swain, C., and Hoffman, S. (2004) Improving Player Choices. *Gamasutra*, March 10, 2004. Retrieved online 1 February, 2005, at http://www.gamasutra.com/features/20040310/fullerton_01.shtml

Garneau, P. (2002) Emergence: Making Games Deeper. Retrieved online 24 July, 2003, at http://www.pagtech.com/Articles/Emergence.html

Gee, J.P. (2004) Learning by Design: Games as Learning Machines. *Gamasutra*, March 24, 2004. Retrieved online 1 February, 2005, at http://www.gamasutra.com/gdc2004/features/20040324/gee_01.shtml

Hargrove, W., Gardner, R., Turner, M., Romme, W, and Despain, D. (2000) Simulating fire patterns in heterogeneous landscapes. *Ecological Modelling* 135, pp. 243-263.

Haykin, S. (1994) *Neural Networks: A Comprehensive Foundation*. Maxwell Macmillan International.

Hecker, C. (2000) Physics in Computer Games. *Communications of the ACM* 43 (7), pp. 34-37.

Holland, J. (1998) *Emergence: from Chaos to Order*. Oxford: Oxford University Press.

Jennings, M. (2000) Theory and Models for Creating Engaging and Immersive Ecommerce Websites. *Proceedings of the 2000 ACM SIGCPR Conference on Computer Personnel Research*, pp. 77-85.

Johnson, S. (2001) *Emergence: the Connected Lives of Ants, Brains, Cities and Software*. New York: Scribner.

Johnson, D., and Wiles, J. (2001) Computer Games with Intelligence. *Australian Journal of Intelligent Information Processing Systems* 7, pp. 61-68.

Johnson, D., and Wiles, J. (2003) Effective affective user interface design in games. *Ergonomics* 46 (13/14), pp. 1332-1345.

Juul, J. (2000) What computer games can and cannot do. Presented at the *Digital Arts and Culture Conference*, Bergen, 2-4 August, 2004. Retrieved online 2 February, 2005, at http://www.jesperjuul.dk/text/WCGCACD.html

Juul, J. (2004) Working with the Player's Repertoire. *International Journal on Intelligent Games and Simulation* 3 (1), pp. 54-61.

Kane, B. (2003) Postcard from GDC 2003: 34 Ways to Put Emotions into Games. *Gamasutra*. March 8, 2003. Retrieved online 1 February, 2005, at http://www.gamasutra.com/gdc2003/features/20030308/kane_emotion_01.htm

LaMothe, A. (1999) *Tricks of the Windows Game Programming Gurus*. Indianapolis, Ind.: SAMS.

Lazzaro, N. (2004) Why we Play Games: Four Keys to More Emotion without Story. Retrieved online 1 February, 2005, at http://www.xeodesign.com/whyweplaygames/xeodesign_whyweplaygames.pdf

Lazzaro, N., and Keeker, K. (2004) What's My Method? A Game Show on Games. *Extended Abstracts of the 2004 Conference on Human Factors in Computing Systems*. New York, NY: ACM Press, pp. 1093-1094.

Leonard, T. (2003) Building an AI Sensory System: Examining the Design of Thief: The Dark Project. *Gamasutra*, March 7, 2003. Retrieved online 2 February, 2005, at http://www.gamasutra.com/gdc2003/features/20030307/leonard_pfv.htm

Mateas, M (2002) Interactive Drama, Art and Artificial Intelligence. Ph.D. Thesis. Technical Report CMU-CS-02-2006, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2002. Retrieved online at 2 February, 2005, at http://www-2.cs.cmu.edu/~michaelm/publications/CMU-CS-02-206.pdf

Mitchell, M. (1998) *An Introduction to Genetic Algorithms*. Cambridge, Mass.: MIT Press.

McLean, J. (2002) Conversations from GDC Europe: Bill Fulton, Zeno Colaco, Harvey Smith. *Gamasutra*, September 11, 2002. Retrieved online 2 February, 2005, at http://www.gamasutra.com/features/20020911/mclean_01.htm

Pachet, F., and Addressi, A.R. (2004) Music: When Children Reflect on their own Playing Style: Experiments with Continuator and Children. *Computers in Entertainment* 2 (1), pp. 14.

Pagulayan, R., Keeker, K., Wixon, D., Romero, R., and Fuller, T. (2003) User-Centered Design in Games. In J.A. Jacko and A. Sears (Eds.), *The Human-Computer Interaction Handbook: Fundamentals, Evolving Techniques and Emerging Applications*. Mahwah, NJ: Lawrence Erlbaum Associates., pp. 883-905.

Perla, P., Markowitz, M., Nofi, A., Weuve, C., Loughran, J., and Stahl, M. (2000) Gaming and Shared Situation Awareness. Technical Report CRM

D0002722.A2/Final, Centre for Naval Analyses, November 2000. DOD Document.

Poiker, F. (2002) Creating Scripting Languages for Nonprogrammers. In S. Rabin (Ed.), *AI Game Programming Wisdom*. Hingham, Mass.: Charles River Media, Inc., pp. 520-529.

Rabin, S. (2000) Designing a General Robust AI Engine. In M. DeLoura (Ed.), *Game Programming Gems*. Hingham, Mass.: Charles River Media, Inc., pp. 221-236.

Rabin, S. (2002) Implementing a State Machine Language. In S. Rabin (Ed.), *AI Game Programming Wisdom*. Hingham, Mass.: Charles River Media, Inc., pp. 314-320.

Rabin, S. (2004) Common Game AI Techniques. In S. Rabin (Ed.), *AI Game Programming Wisdom 2*. Hingham, Mass.: Charles River Media, Inc., pp. 3-14.

Reynolds, C. (1987) Flocks, Herds, and Schools: A Distributed Behavioural Model. *Computer Graphics* 21 (4), pp. 25-34.

Russel, S. and Norvig, P. (2003) *Artificial Intelligence: A Modern Approach* (Second Edition). New Jersey: Prentice Hall.

Sharafi, P., Hedman, L., and Montgomery, H. (in press) Using information technology: engagement modes, flow experience, and personality orientations. *Computers in Human Behaviour*. Retrieved online 9 April, 2004, at http://www.sciencedirect.com

Smith, H. (2002) Systemic Level Design. Presented at the Game Developers Conference, San Jose, CA, March 21-23, 2002. Retrieved online 13 April, 2005, at http://www.gdconf.com/archives/2002/

Smith, H. (2001) The Future of Game Design: Moving Beyond Deus Ex and Other Dated Paradigms. Retrieved online July 23, 2003, at http://www.planetdeusex.com/witchboy/ articles/thefuture.shtml.

Stam, J. (2000) Interacting with Smoke and Fire in Real Time. *Communications of the ACM* 43 (7), pp. 77-83.

Stam, J. (2003) Flows on Surfaces of Arbitrary Topology. *ACM Transactions on Graphics* (TOG) 22 (3): Proceedings of SIGGRAPH 2003, pp. 724-731.

Stripinis, D. (2001) The (Not So) Dark Art of Scripting for Artists. *Game Developer Magazine*, pp. 40-45.

Sweetser, P. (2004a) Strategic Decision-Making with Neural Networks and Influence Maps. In S. Rabin (Ed.), *AI Game Programming Wisdom 2*. Hingham, Mass.: Charles River Media, Inc., pp. 439-446.

Sweetser, P. (2004b) How to Build Neural Networks for Games. In S. Rabin (Ed.), *AI Game Programming Wisdom 2*. Hingham, Mass.: Charles River Media, Inc., pp. 615-625.

Sweetser, P. (2004c) How to Build Evolutionary Algorithms for Games. In S. Rabin (Ed.), *AI Game Programming Wisdom 2*. Hingham, Mass.: Charles River Media, Inc., pp. 627-637.

Sweetser, P. & Dennis, S. (2003). Facilitating Learning in a Real Time Strategy Computer Game. *Entertainment Computing: Technologies and Applications* (eds. Ryohei Nakatsu and Junichi Hoshino). Kluwer Academic Publishers, Boston., pp. 49-56.

Sweetser, P., Johnson, D., Sweetser, J. and Wiles, J. (2003) Creating Engaging Artificial Characters for Games. *Proceedings of the Second International Conference on Entertainment Computing*. Pittsburgh, PA: Carnegie Mellon University., pp. 1-8.

Sweetser, P. and Wyeth, P. (in press) GameFlow: A Model for Evaluating Player Enjoyment in Games. To be published in *ACM Computers in Entertainment* 3 (3).

Tozour, P. (2001) Influence Mapping. In M. Deloura (Ed.), *Game Programming Gems 2*. Hingham, MA: Charles River Media, Inc., pp. 287-297.

Treuille, A., McNamara, A., Popović, Z., and Stam, J. (2003) Keyframe Control of Smoke Simulations. *ACM Transactions on Graphics* (TOG) 22 (3): Proceedings of SIGGRAPH 2003, pp. 716-723.

Vass, M., Carroll, J., and Shaffer, C.A. (2002) Supporting Creativity in Problem Solving Environments. *Proceedings of the Fourth Conference on Creativity and Cognition*, pp. 31-37.

Walker, R. (2004) The Design and Production of Half-Life 2. Presented at the *Australian Game Developers' Conference*, Melbourne, 2-4 December, 2004. Retrieved online 13 April, 2005, at http://www.agdc.com.au/about/arch04_schedule_sat.php

Woodcock, S. (2000) Game AI: The State of the Industry. *Gamasutra*, November 1, 2000. Retrieved online 2 February, 2005, at http://www.gamasutra.com/features/20001101/woodcock_01.htm

Woodcock, S. (2003) Games Making Interesting Use of Artificial Intelligence Techniques. Retrieved online 2 February, 2005, at http://www.gameai.com

Games

Bethseda (2002) The Elder Scrolls III: Morrowind. Available online 2 February, 2005, at http://www.morrowind.com

Electronic Arts (2000) The Sims. Available online 2 February, 2005, at http://thesims.ea.com/

Microsoft (2002) Age of Mythology. Available online 2 February, 2005, at http://www.microsoft.com/games/ageofmythology/egypt_home.asp

Smart, D. (1996) BattleCruiser: 3000AD. Available online 2 February, 2005, at http://www.3000ad.com/home.shtml

Valve (1998) Half-Life. Available online 2 February, 2005, at http://www.planethalflife.com/half-life/

Valve (2004) Half-Life 2. Available online 2 Feburary, 2005, at http://www.planethalflife.com/half-life2/

Appendix A

Questionnaire on Interaction in Games

Interaction in Games Questionnaire

Penny Sweetser, School of ITEE, University of Queensland.

This questionnaire will take approximately 5-10 minutes to complete. It is part of a research project for my PhD. If you have any questions then feel free to contact me on the above email. You are free to exit at any time, but please **press the submit button** at the bottom of the screen so that the data is saved. Thank you for your time.

Section 1

Age

Gender Male

Nationality

Approximately how experienced would you rate yourself at playing computer games?

Very Inexperienced						Very Experienced
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7

What is your favourite type of game? (if you have more than one favourite, choose the one you have played most recently)

Role-playing Games   If *other*, please specify:

Please answer the rest of the questionnaire with respect to this type of game.

Approximately how frequently do you play *this type of game*?

○ Daily   ○ Weekly   ○ Fortnightly   ○ Monthly   ○ Less than Monthly

Comments on frequency (optional):

What games *of this type* have you played?

Section 2: Questions 1-7 (of 28)

1. How do objects that look the same but that cannot be used in the same way affect your enjoyment of your preferred type of game (eg one barrel can be broken but another cannot)?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

2. How does *not* being able to use game objects because they are only scenery affect your enjoyment of your preferred type of game (eg a piece of furniture cannot be moved)?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

3. How does *not* being able to use game objects in the way that you would expect to be able to use them affect your enjoyment of your preferred type of game (eg a bucket can be kicked but not picked up)?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

4. How does being able to use objects in the same way as you would in the real world affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

5. How does being able to use objects in the same way as you have in previous games affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

6. How does having a wide variety of possible ways to use objects in the game environment affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

7. How does having new and unique ways of using objects affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

Any other comments:

Section 3: Questions 8-14 (of 28)

8. How does a moving and powerful soundtrack affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

9. How do sound effects that set the mood (eg build up suspense) affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

10. How do sound effects that cause an emotional response (eg fear or happiness) affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

11. How do life-like graphics affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

12. How do graphics with obvious inconsistencies (eg a body sticking through a wall) affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

13. How does an introduction that sets the scene for the game affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

14. How does a strong storyline affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

Any other comments:

Section 4: Questions 15-21 (of 28)

15. How does only having one way to perform a task or solve a problem affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

16. How does needing to figure out what the game developer wanted you to do to perform a task or solve a problem affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

17. How does being able to perform a task or solve a problem in your own way affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

18. How does needing to use trial and error to perform a task or solve a problem affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

19. How does *not* being able to perform a task or solve a problem within a reasonable period of time affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

20. How does *not* being able to perform a task or solve a problem at all affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

21. How does needing to get help from the internet or another person in order to solve a problem affect your enjoyment of your preferred type of game?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

Any other comments:

Section 5: Questions 22-28 (of 28)

22. How does being affected by the laws of gravity affect your enjoyment of your preferred type of game (eg you can jump in a realistic way or take falling damage)?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

23. How do objects that follow the laws of gravity in a realistic way affect your enjoyment of your preferred type of game (eg bouncing, falling, rolling down hills)?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

24. How does momentum affect your enjoyment of your preferred type of game (eg being pushed backwards when hit)?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

25. How does water that behaves in a realistic way affect your enjoyment of your preferred type of game (eg flows, wets, cools, current)?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

26. How do objects that are affected by water in a realistic way affect your enjoyment of your preferred type of game (eg weapons cannot work underwater)?

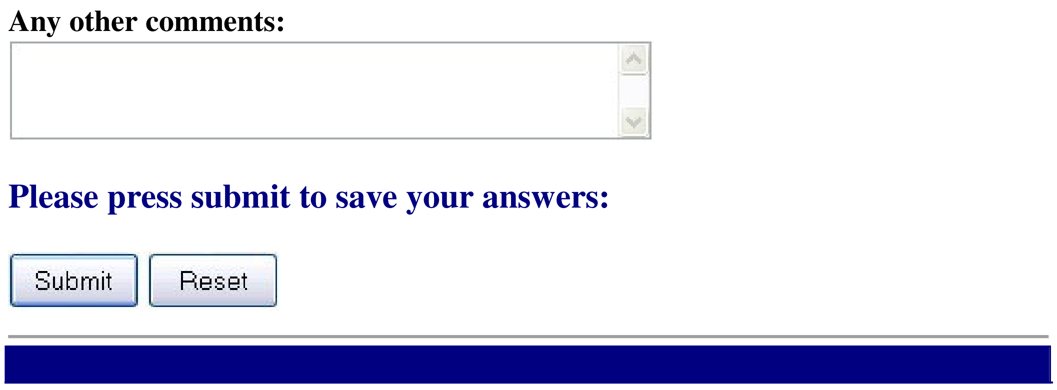
Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

27. How does fire that behaves in a realistic way affect your enjoyment of your preferred type of game (eg burns, ignites, heats)?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9

28. How do objects that are affected by fire in a realistic way affect your enjoyment of your preferred type of game (eg flammable objects burn or explosives explode when they come in contact with fire)?

Much less enjoyable				No effect				Much more enjoyable
○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9



Any other comments:

Please press submit to save your answers:

Appendix B

Pseudo-code for Environment in EmerGEnT System

Heat

```
// get the heat capacities of the cell and the neighbour
HCCell = material(cell).SHC * cell.Mass;
HCNeigh = material(neigh).SHC * neigh.Mass;
// calculate the difference in temp between the cell and neighbour
EnergyFlow = cell.Temp – neigh.Temp;
// convert from heat to energy
EnergyFlow *= HCCell;
// multiply by a constant for cell update speed
EnergyFlow *= ConstantEnergyFlowFactor;
// heat doesn’t flow against wind
if (neigh isn’t against wind)
{
        // cell has higher heat than neigh
        if (EnergyFlow > 0)
        {
                // heat flow into neighbour
                neigh.Temp += EnergyFlow / HCNeigh;
                // heat flows from cell
                cell.Temp -= EnergyFlow / HCCell;
        }
        // detect and kill oscillations
        if ((EnergyFlow > 0) && (neigh.Temp < cell.Temp))
        {
                // find average temp of cell and neigh
                TotalEnergy = (HCCell * cell.Temp) + (HCNeigh * neigh.Temp);
                AverageTemp = TotalEnergy / (HCCell + HCNeigh);
                // set cell and neigh to average temp
                cell.Temp = AverageTemp;
                neigh.Temp = AverageTemp;
                // increase heat flow with wind
                if (neighbour is with wind)
                {
                        cell.Temp /= (1 + (windspeed * wind_const));
                        neigh.Temp *= (1 + (windspeed * wind_const));
                }
        }
}
```

```
// neighbour lower than cell
if ((neigh.Height < cell.Height) &&
  // cell has the max fluid it will hold, modified by the slope – easier to flow downhill
  (cell.Fluid > (material(cell).MaxFluid * (neigh.Height / cell.Height))))
{
        // flow equals the difference between fluid in cell and neigh divided by four
        flow = (cell.Fluid – neigh.Fluid) * 0.25;
        // flow is increased proportionally to slope
        flow = flow * (cell.Height / neigh.Height) * flow_const;
        // flow cannot be less than zero
        if (flow < 0) flow = 0;
        // update cell and neigh with flow
        cell.Fluid -= flow;
        neigh.Fluid += flow;
        // cell.Fluid cannot be less than zero
        if (cell.Fluid < 0) cell.Fluid = 0;
}
// neighbour higher than cell
else if ((neigh.Height > cell.Height) &&
  // cell has the max fluid it will hold, modified by slope – harder to flow uphill
  (cell.Fluid > (material(cell).MaxFluid * (neigh.Height / cell.Height)))){
        // flow equals difference between the fluid in cell and neigh divided by four
        flow = (cell.Fluid – neigh.Fluid) * 0.25;
        // flow is decreased proportionally to slop
        flow = flow * (cell.Height / neigh.Height) / flow_up_const;
        // flow cannot be less than zero
        if (flow < 0) flow = 0;
        // update cell and neigh with flow
        cell.Fluid -= flow;
        neigh.Fluid += flow;
        // cell fluid cannot be less than zero
        if (cell.Fluid < 0) cell.Fluid = 0;
}
// neighbour on same level
// fluid in cell must exceed max fluid cell can hold
else if (cell.Fluid > material(cell).MaxFluid)
{
        // flow equals difference between cell and neigh divided by four
        flow = (cell.Fluid – neigh.Fluid) * 0.25;
        // flow cannot be less than zero
        if (flow < 0) flow = 0;
        // update cell and neigh with flow
        cell.Fluid -= flow;
        neigh.Fluid += flow;
        // cell fluid cannot be less than zero
        if (cell.Fluid < 0) cell.Fluid = 0;
}
```

Pressure

```
// if there is more pressure in cell than in neighbour
if (cell.Pressure > neigh.Pressure)
{
        // calculate the pressure ratio between cell and neigh
        pressure_ratio = cell.Pressure / neigh.Pressure;
        // if pressure ratio is more than explosion ration then explode
        if (pressure_ratio > explosion_ratio)
        {
                // release heat proportional to pressure ratio
                cell.Temp += (explosion_const * pressure_ratio) * 0.25;
        }
        // calculate pressure difference between cell and neigh
        PressureFlow = cell.Pressure – neigh.Pressure;
        // pressure diffuses to neighbour
        neigh.Pressure += PressureFlow * 0.25;
        cell.Pressure -= PressureFlow * 0.25;
        // detect and remove oscillations
        if ((PressureFlow > 0) && (neigh.Pressure < cell.Pressure))
        {
                // calculate the average pressure of cell and neigh and distribute evenly
                TotalPressure = cell.Pressure + neigh.Pressure;
                AveragePressure = TotalPressure / 2;
                cell.Pressure = AveragePressure;
                neigh.Pressure = AveragePressure;
        }
}
```

Fire

```
// temperature is the difference between the temp of the cell and the flashpoint of the
// material in the cell and the wetness of the cell
Temp = cell.Temp – (material(cell).FlashPoint + cell.Wetness);
// damage the cell
if (Temp > 0) cell.Damage = ((Temp * material(cell).BurnRate)  - cell.Wetness) *
    burn_const;
// convert to actual burning value
if (Temp > (material(cell).MaxBurn * 2)) Burn = material(cell).MaxBurn;
else if (Temp > 0) Burn = (1.0 – ((0.25 * Temp) / material(cell).MaxBurn)) * Temp;
// burn cannot exceed MaxBurn
if (Burn > material(cell).MaxBurn) Burn = material(cell).MaxBurn;
// reduce burn by amount of damage in cell (less fuel to burn when damaged)
Burn -= cell.Damage;
// burn cannot be less than zero
if (Burn < 1) Burn = 0;
// Heat the cell up from the burning
cell.Temp += Burn * material(cell).BurnTemp;
cell.Burn = Burn;
```

Appendix C

Pseudo-code for Objects in EmerGEnT System

Heat (obj)

```
// Find current heat capacities
HCObj = material(obj).SHC * obj.Mass;
HCCell = material(cell).SHC * cell.Mass;
EnergyFlow = cell.Temp  - obj.Temp;

// Energy flowing from cell to object
if (EnergyFlow > 0){
        // Convert from heat to energy
        EnergyFlow *= HCCell;
        // A constant according to cell update speed
        EnergyFlow *= ConstantEnergyFlowFactor;
        cell.NewTemp -= (EnergyFlow / HCCell);
        obj.NewTemp += (EnergyFlow / HCObj);

        // Detect and kill oscillations
        if (cell.NewTemp < obj.NewTemp){
                TotalEnergy = (HCObj * obj.NewTemp) + (HCCell * cell.NewTemp);
                AverageTemp = TotalEnergy / (HCObj + HCCell);
                obj.NewTemp = AverageTemp;
                cell.NewTemp = AverageTemp;
        }
}

// Energy flowing from object to cell
else if (EnergyFlow < 0){
        EnergyFlow *= -1;
        // Convert from heat to energy
        EnergyFlow *= HCObj * ConstantEnergyFlowFactor;
        cell.NewTemp += (EnergyFlow / HCCell);
        obj.NewTemp -= (EnergyFlow / HCObj);

        // Detect and kill oscillations
        if (obj.NewTemp < cell.NewTemp){
                TotalEnergy = (HCObj * obj.NewTemp) + (HCCell * cell.NewTemp);
                AverageTemp = TotalEnergy / (HCObj + HCCell);
                obj.NewTemp = AverageTemp;
                cell.NewTemp = AverageTemp;
        }
}
```

Fluid Flow (obj)

```
// will flow into object if cell has any fluid and if object is not full of water
if ((cell.Fluid > 0) and (obj.Fluid < (material(obj).MaxFluid * obj.Mass/cell.Mass))
        // and object affords flowing
        and AffordsFlow(obj))
{
        // should fill obj with same proportion of fluid as in cell
        flow = (cell.Fluid - obj.Fluid) * 0.25 * (obj.Mass/cell.Mass);
        if (flow < 0) flow = 0;
        cell.NewFluid -= flow;
        obj.NewFluid += flow;
        if (cell.NewFluid < 0) cell.NewFluid = 0;
}

// will flow from obj to cell if obj is over-full
excess = obj.Fluid – material(obj).MaxFluid;
if ((excess > 0)
        // object affords flowing
        && AffordsFlow(obj))
{
                // objects are smaller than cells
                flow = excess * (obj.Mass / cell.Mass);
                cell.NewFluid += flow;
                obj.NewFluid -= flow;
        }
}
```

Fire (obj)

```
//Burning temperature
Temp = obj.Temp - (material(obj).FlashPoint + obj.Wetness);
//Damage the cell
if (Temp > 0){
        obj.NewDamage += ((Temp * material(obj).BurnRate) - obj.Wetness) * const;
}
//Convert to actual burning value
if (Temp > (material(obj).MaxBurn * 2)) Burn = material(obj).MaxBurn;
else if (Temp > 0) Burn = ((1.0 - ((0.25 * Temp) / material(obj).MaxBurn)) * Temp);
if (Burn > material(obj).MaxBurn) Burn = material(obj).MaxBurn;
Burn -= obj.Damage;
if (Burn < 1) Burn = 0;
// Heat the object up from the burning
obj.NewTemp += (Burn * material(obj).BurnTemp) * BurnHeatConst;
obj.Burn = Burn;
```

Pressure (obj)

```
// high absolute pressure in cell immediately damages object
if (cell.Pressure > high_pressure){
        obj.NewDamage += (cell.Pressure * pressure_damage_const);
}

// cell pressure is higher than object pressure and object affords flowing
if ((cell.Pressure > obj.Pressure) and AffordsFlow(obj))
{
        // flow of pressure: cell to object
        PressureFlow = (cell.Pressure - obj.Pressure) * obj.Mass/cell.Mass;
        obj.NewPressure += PressureFlow;
        cell.NewPressure -= PressureFlow;
}

// high pressure in object -- causes explosion
if (obj.Pressure > cell.Pressure){

        // ratio of object pressure to cell pressure - modified by obj material
        pressure_ratio = (obj.Pressure / cell.Pressure);

        // if pressure difference is great enough then explode
        if ((pressure_ratio > (explosion_ratio * material(obj).Strength))
                and AffordsExploding(obj)){
                cell.NewTemp += (explosion_const * pressure_ratio);
                PressureFlow = obj.Pressure - cell.Pressure;
                cell.NewPressure += PressureFlow;
                obj.NewPressure -= PressureFlow;
        }

        // flow of pressure: object to cell
        else
        // object affords flowing out of
        if (AffordsFlow(o)){
                PressureFlow = (obj.Pressure - cell.Pressure) * obj.Mass/cell.Mass;
                obj.NewPressure -= PressureFlow;
                cell.NewPressure += PressureFlow;
        }
}
```

Appendix D

Pseudo-code for Agents in EmerGEnT System

GetComfort(x, y)

```
// Comfort = (temp * const) + (burn * const) + (pressure * const) + (wetness * const)
// burn > temp > pressure > wetness
comfort = (cells[x,y].Temp * tempconst) + (cellx[x,y].Burn * burnconst) +
          (cells[x,y].Pressure * pressconst) + (cellx[x,y].Wetness * wetconst);
if (comfort > 1.0) comfort = 1.0;
return comfort;
```

React(ai)

```
comfort = GetComfort(ai.x, ai.y);

// neighbourhood size = 3
n = 3;

// if comfortable – stand still
if (comfort < 0.1) StopMove();

// if uncomfortable – move
else if (comfort < 0.3){
        // set speed – 1, animation walk
        speed = 1.0f;
        // move – to dest
        GoalDest(comfort, ai, speed, n);
}
// if distressed – move quickly
else if (comfort < 0.6){
        // set speed – 2, animation run
        s = 2.0f;
        // move – to dest
        GoalDest(comfort, ai, s, n);
}
else {
        // set speed – 3, animation run
        s = 3.0f;
        // move – to dest
        GoalDest(comfort, ai, s, n);
}
```

CalcDesire(goalx, goaly)

```
for area around goal
    // distance = city block distance of x,y from goal
    distance =  abs(x - goalx) + abs(y - goaly);

    // desire of x,y is multiplied by 0.7 for each step out from goal
    Desire[x][y] += pow(0.7, distance);
}
```

GoalDest(comfort, ai, speed, n)

```
for agents local neighbourhood (n=3)
{
    // only calculate if not previously calculated and store
    if (Comfort[x][y] == NULL)  Comfort[x][y] = GetComfort(x,y);

    this_comfort = Comfort[x][y];

    // add in desire from influence map
    this_comfort = (Comfort[x][y] * 0.5) + ((1 - Desire[x][y]) * 0.5);

    // set destination to the most comfortable cell in neighbourhood
    if (this_comfort < comfort){
        comfort = this_comfort;
        destx = x;
        desty = y;
    }
}
// find destination within immediate neighbourhood (n=1)
ImmDest(comfort, ai, speed, 1, destx, desty);
```

ImmDest(comfort, ai, speed, n, goalx, goaly)

```
comfort = (comfort * 0.5) + ((abs(goalx - ai.x) + abs(goaly - ai.y)) / 8.0f);

for local neighbourhood around agent (n=1){

    // 50% weighting = divide by 8
    goal_dist = (abs(goalx - x) + abs(goaly - y)) / 8.0f;

    // 50% weighting = multiply by 0.5
    this_comfort = (Comfort[x][y] * 0.5) + goal_dist;

    // set immediate destination to most comfortable cell in neighbourhood
    if (this_comfort < comfort){
        comfort = this_comfort;
        destx = x;
        desty = y;
    }
}
// move to destination
Move(a, destx, desty, s);
```

Appendix E

Contents of Accompanying CD

This icon indicates that there is further information or a demonstration on the accompanying CD.

The accompanying CD includes two folders: *Papers* and *Demos*.

The *Papers* folder includes a set of papers that are relevant to this thesis and a list of the references for these papers.

The *Demos* folder includes demonstrations of the EmerGEnT system and a README file. The README includes a list of the demonstrations on the CD and instructions for using the EmerGEnT system.