Creating a Virtual Training Environment for Traffic Accident Investigation for the Dubai Police Force

Ahmed BinSubaih

Doctor of Philosophy

Department of Computer Science

University of Sheffield

October 2007

Abstract

This research investigates the use of gaming technology (especially game engines) in developing virtual training environments, and comprises of two main parts. The first part of the thesis investigates the creation of an architecture that allows a virtual training environment (i.e. a 'game') to be portable between different game engines. The second part of the thesis examines the learning effectiveness of a virtual training environment developed for traffic accident investigators in the Dubai police force. The second part also serves to evaluate the scalability of the architecture created in the first part of the thesis.

Current game development addresses different aspects of portability, such as porting assets and using middleware for physics. However, the game itself is not so easily portable. The first part of this thesis addresses this by making the three elements that represent the game's brain portable. These are the game logic, the object model, and the game state, which are collectively referred to in this thesis as the game factor, or G-factor.

This separation is achieved by using an architecture called game space architecture (GSA), which combines a variant of the model-view-controller (MVC) pattern to separate the G-factor (the model) from the game engine (the view) with on-the-fly scripting to enable communication through an adapter (the controller). This enables multiple views (i.e. game engines) to exist for the same model (i.e. G-factor). The principal findings from the evaluation process reveal that GSA is capable of servicing the same G-factor to multiple game engines and that it supports modifiability. They also reveal that GSA adds little development overhead. The ability of GSA to scale to real world applications is demonstrated by the development of a virtual training environment.

The second part of the thesis examines the development of a virtual training environment for traffic accident investigators in the Dubai police force. Their training needs were identified in a field study conducted in the summer of 2004 to assess the current training methods of lectures and on-the-job training. The virtual environment was then developed by combining game design and instructional design to ensure the learning objectives were integral to the gameplay. To evaluate the learning effectiveness of the virtual environment an experiment was conducted in February and March of 2006 for fifty-six police officers from the Dubai police force. They were divided into two groups: novices (0 to 2 years experience) and experienced investigators (with more than 2 years experience). The experiment revealed significant performance improvements in both groups, with the improvement reported in novices significantly higher than the one reported in experienced investigators.

Acknowledgements

I would like to thank the Commander-in-Chief of Dubai Police, Lieutenant General Dhahi Khalfan Tamim for his constant encouragement ever since finishing my master's degree to pursue a PhD. I am also grateful to the Dubai police force for sponsoring this work. I would like also to thank my fellow officers in the Dubai police force for their support without which the field study and the experiments would have been very difficult to carry out.

They have put up with my constant interruptions and did not mind me following them with a camera. They have gone out of their duty to help this endeavour. The serious game developed as part of this thesis would not have reached its current state without the constant refinement and without the expertise of the subject matter experts especially Brigadier Mohammed Al Zaffin, Lieutenant Colonel Mohammed Al-Ali, Major Khalid Kinsham, Major Omar Ashour, Captain Maher Bin Haider, and many others from the Dubai Traffic Department. During the field study, my first stop was to work at Bur Dubai police station and it was an experience that was made very insightful with the expertise, professionalism, and support of officers there. This list is by no means a comprehensive one. In fact, more than 100 officers contributed in one way or another. This work will always be indebted to you. I am also grateful to my friends in Sheffield who helped test the serious game. These are Khalfan Al Shoaili, Nasser Al Yarubi, Ahmed Al Shokaili, Ayoub Al Awadi, and Ahmed Bin Humaidan.

I would like to express my sincere appreciation to my supervisor Dr. Steve Maddock. He was very influential in motivating me to do a PhD and the reason for selecting the University of Sheffield for another educational journey after already having done my undergraduate studies here. His endless support, encouragement, and invaluable guidance have made it all possible. I would also like to thank my second supervisor Dr. Daniela Romano for her motivation and her useful suggestions and feedback. Also thanks to my original second supervisor Dr. Gordon Manson and the panel for their comments and constructive criticism. This work is also shaped by comments from reviewers during the publication of the material in conferences, in a journal, and in a book chapter whose names remain unknown. Thank you all so much.

Many thanks to all the members of the graphics group (past and present) for their help along the way. They have made the journey more tolerable and knowledgeable. Thanks to now Dr. Mike Meredith for his technical support and participating in the exchanges of jibes often about our social life or lack of it. Thanks to now Dr. James Edge for discussing my ideas, proof reading, and encouraging me to publish. Thanks to now Dr. Manuel Sanchez for sharing his knowledge about game development. Many thanks also to Dr. Alan Watt for keeping an eye on me when in fact someone needs to keep an eye on him! Many thanks also to the Sheffield Academy of

Martial Arts where it was nice to get rid of all the frustration even though it sometimes meant being used as a punching bag. It was all worth it to teach me to keep my guards up and persevere. Many thanks to Sifu Graham Abdulla, his brother Adrian, and others at the academy for making me part of the Lau Gar Kung Fu family. What I have learned from you guys will always stay with me.

This thesis would not have been possible without the endless support from my family and friends. I would specially like to thank my family for being patient despite the long gaps between visits. What made this even more difficult for me was missing on my nephews and nieces growing up. Also thank you for putting up with me forgetting sometimes how to behave in a conservative culture. This is partly to be blamed on the English Sitcom (Dad's Army, Steptoe and Son, Only Fools and Horses, Hancock's Half Hour, On the Buses, Till Death Us Do Part, Love Thy Neighbour, etc) and partly for spending 11 years in the UK. The journey started from learning English in 1994 in Torquay to this day with few years in between spent working in the Dubai police force. When I go back home I probably need a crash course (or a serious game) on culture.

Research Publications

Refereed Publications

BinSubaih, A., Maddock, S., and Romano, D. (2008). Developing a serious game for police training. In *Handbook of Research on Effective Electronic Gaming in Education*, Ferdig, R.E. (Ed.) Information Science Reference (In press).

BinSubaih, A., and Maddock, S. (2008). Game Portability Using a Service-Oriented Approach. International Journal of Computer Games Technology (IJCGT) (In press).

BinSubaih, A., and Maddock, S. (2007). G-factor portability in game development using game engines. In *Proceedings of 3rd International Conference on Games Research and Development 2007* (CyberGames 2007), pages 163-170.

BinSubaih, A., Maddock, S., and Romano, D. (2006). A serious game for traffic accident investigators. Special Issue of International Journal of Interactive Technology and Smart Education on "Computer Game-based Learning.", 3(4):329-346.

BinSubaih, A., and Maddock, S. (2006). Using ATAM to evaluate a game-based architecture. *Workshop on Architecture-Centric Evolution (ACE 2006), hosted at the 20th European Conference on Object-Oriented Programming ECOOP 2006.*

BinSubaih, A., Maddock, S., and Romano, D. (2006). An architecture for portable serious games. *Doctoral Symposium, hosted at the 20th European Conference on Object-Oriented Programming ECOOP 2006.*

BinSubaih, A., Maddock, S., and Romano, D. (2005). Comparing the use of a tabletop and a collaborative desktop virtual environment for training police officers to deal with traffic accidents: Case study. In *Proceedings of International Conference on Engineering Education*, 2:94-100.

BinSubaih, A., Maddock, S., and Romano, D. (2005). Game logic portability. In *Proceedings of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology ACE 2005, Computer Games Technology session*, pages 458-461.

BinSubaih, A., Maddock, S., and Romano, D. (2005). OntRAT: ontology-based rules acquisition tool. In *Proceedings of m-ICTE2005, 3rd International Conference on multimedia and Information & Communication Technologies in Education*, pages 978-983.

BinSubaih, A., Maddock, S., Romano D.M. (2005). Tabletop vs CVE Police Training (Poster). PRESENCE 2005, In *Proceedings of 8th Annual International Workshop on Presence*, London, England, September 21-23, 2005.

BinSubaih, A., Maddock, S., and Romano, D. (2004). A collaborative virtual training architecture for investigating the aftermath of vehicle accidents. *In Proceedings of Middle Eastern Simulation and Modelling Conference*, pages 170-178.

BinSubaih, A., Maddock, S., and Romano, D. (2004). A domain-independent multiplayer architecture for training. In *Proceedings of International Workshop in Computer Game Design and Technology*, pages 144-151.

BinSubaih, A., Maddock, S., and Romano, D. (2004). An architecture for domain-independent collaborative virtual environments. In *Proceedings of 5th Annual European GAME-ON Conference*, pages 84-88.

Technical Report

BinSubaih, A., Maddock, S., and Romano, D. (2007). A survey of 'game' portability. Department of Computer Science Technical Report CS-07-05, 2007, University of Sheffield.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

This thesis is composed of two main parts. The first part of the thesis (chapters 2 - 4) investigates portability between game engines by developing an architecture that allows a 'game' (i.e. a virtual training environment) to exist independently of a game engine. The second part of the thesis (chapters 5 & 6) presents the development of a virtual training environment for traffic accident investigators in the Dubai police force.

Game engines are increasingly being used in developing virtual training environments. This means developers can concentrate on the virtual training environment rather than the core technology, which is supplied by the engine (e.g. Quake game engine). The problem, however, with using game engines is that they make the game dependent on the engine it was developed on. This ties the game's future to the engine's future. This lack of future security has become known as "the RenderWare Problem" (Carless, 2007) after the acquisition of RenderWare by Electronic Arts (EA) and its removal from the market. One way to address this problem is to make it possible to easily port a game between game engines.

There are many different aspects of portability supported in current game development. For example, hardware and software abstractions have facilitated the ability to play a game on different hardware and on different operating systems. These abstractions have also facilitated the ability to use level data assets such as 3D models, sound, music, and texture across different game engines. In addition, current game development makes use of components such as artificial intelligence (AI), networking, and physics across engines. The aspect of portability that has not yet received similar attention is porting the game from one game engine to another (e.g. porting Unreal Tournament to the Quake engine or porting Doom to the Unreal engine). The three game elements this thesis argues need to be made portable are the game logic, the object model, and the game state, which together represent the game's brain. The game logic dictates the game behaviour. The object model describes the classes for the objects in a game (e.g. players and non-player characters (NPCs)). The game state holds the working memory of the game at any moment, which contains game objects (e.g. players and NPCs), time, and logged interactions. This thesis will collectively refer to these three elements as the game factor (or G-factor). The ability to port these elements represents the next logical step that follows on from the game-independent engine evolution (Lewis & Jacobson, 2002), and is referred to in this work as the engine-independent game approach.

The complexity involved in porting the G-factor elements is due to practices employed when using a game engine, such as intertwining the game deeply in the engine's code, or providing proprietary languages or classes to create the game logic and the object model. The solution presented in the first part of this thesis is a novel approach based on using a variant of the model-view-controller (MVC) pattern. The MVC pattern divides a system into three logical parts: model (i.e. data), view (i.e. interface),

and controller (i.e. processes events) (Buschmann et al., 1996). This pattern is used to separate the G-factor from the game engine to allow the G-factor to exist independently of the game engine. The G-factor becomes the model and the game engine becomes the view. The MVC pattern allows multiple views (i.e. game engines) to exist for the same model (i.e. G-factor). The architecture developed based on this approach is called the game space architecture (GSA). The success of GSA in being portable, modifiable, and able to perform at an acceptable rate is evaluated using two types of evaluation: unstructured and structured. The unstructured evaluation uses three simple games and a real world application. The structured evaluation uses the architecture trade-off analysis method (ATAM). The findings from the evaluation process are presented in chapter 4.

The second part of the thesis (chapters 5 & 6) describes the development of a virtual training environment for traffic accident investigators. The objective of the development is to demonstrate the ability of GSA to support a real world application. The use of gaming technology to create virtual training environments falls under what is becoming known as serious games (Zyda, 2005; Stone, 2005). Despite the term being described as an oxymoron, since the term game is not perceived as learning (Wexler et al, 2007), the serious games term has received acceptance in many domains (Michael & Chen, 2005). In this thesis, this term is preferred because of its close association with gaming technology compared to more general terms such as immersive learning simulation (Wexler et al, 2007).

An important step in developing any serious game is to understand how learning occurs while playing games and more crucially how the serious game can be based on sound learning and instructional principles. This is needed to ensure that learning is integrated in the gameplay. To achieve this a multidisciplinary solution is required which can involve instructional designers, game designers, and subject matter experts. However, the process of achieving effective learning integration in the game is not yet fully understood, and research is hampered by the lack of practical demonstrations of how effective instructional design is when used alongside game design. Furthermore, the process is undermined by the separation between the two camps – game design and instructional design – as described by Becker (Becker, 2006c). For example, the game design camp views games as already employing sound principles and there is no need for instructional designer involvement. These issues will be addressed in chapter 5 and will form the foundation for the learning and instructional principles that will be used in chapter 6 to develop a serious game for traffic accident investigators (SGTAI).

The learning potential of serious games has been shown in a number of domains such as the military, healthcare, education, and emergency services (see section 5.2), but less so in police training. For example, police training in the Dubai police force still depends mainly on classrooms and field training. The police areas in which simulation technology has shown its practicality have often relied on video-based simulations (Bennell & Jones, 2003) for tasks such as improving shooting and driving skills. The use of serious games in police training is currently limited and there is a lack of empirical studies showing their effectiveness (see section 5.2.3 for examples). That is despite the fact that serious games have been

shown to be particularly effective at dealing with learning that requires practice, especially when the real environment is unsafe. Traffic accident environments are intense, unpredictable, and unsafe, and thus serious games should be useful for training.

In Dubai, a traffic accident happens every three minutes and one person dies every 36.6 hours (RTA, 2005). The training provided for traffic investigators in the Dubai police force uses these real environments to provide hands-on practice. To understand the effect these environments have on traffic accident investigation training, a field study was conducted in the summer of 2004. The field study included travelling to accidents and conducting interviews and experiments. The findings from the field study (see section 6.2.1) were then used to identify the learning objectives for SGTAI. These objectives were then integrated into the gameplay using experiential learning principles and instructional principles (see chapter 6). The effectiveness of SGTAI was tested in an experiment conducted in 2006 for 56 police officers from the Dubai police force. The hypothesis made in this experiment was that a serious game can be an effective learning environment and could help address the problems with the current training methods used by the Dubai police force. The experiment measured the learning effectiveness across novices and experienced investigators. The findings from the experiment (see chapter 6) show the success of SGTAI in transferring learning and show its suitability to address the challenges facing the current training employed by the Dubai police force.

1.1 Aims and Objectives

The two main parts of the thesis address the two aims of this research. The first aim is to create an architecture that supports a game development approach capable of making the G-factor portable between game engines. This will be achieved through the following objectives:

- Investigate the typical game development practices promoted by game engines and examine how they affect G-factor portability.
- Examine what makes a game dependent on a particular game engine and propose solutions to address the dependency.
- Develop an architecture to address the dependency issues and evaluate its ability to make the G-factor portable while remaining modifiable and able to perform at an acceptable rate.

The second aim is to develop a serious game for traffic accident investigators in the Dubai police force, which also serves to evaluate the architecture. This will be achieved through the following objectives:

- Determine how a serious game can be built for a training environment.
- Analyse the current training practices used by the Dubai police force and investigate how a serious game can support them.
- Develop the serious game using the new architecture to examine its ability to scale to real world applications.

- Evaluate the learning effectiveness of the training environment for novice and experienced investigators through an empirical study.

1.2 Research Methodology

The two parts of the thesis employ the same methodology which consists of three main stages: a quantitative and qualitative analysis stage, an iterative design process stage, and a quantitative and qualitative evaluation process stage. The quantitative method relies primarily on numbers and involves the use of experiments, surveys, and testing, whereas the qualitative method relies on the reasons behind a social phenomena and involves the use of case studies, focus groups and documents and artefact studies (De Villiers, 2005). De Villiers argues that the practices used to collect data for both methods overlap and often studies rely on qualitative data to formulate the quantitative questions.

For the first part of the thesis, the analysis stage consisted of two surveys. The first survey of game engines used methods such as examining sources like software development kits (SDK), published materials, forums, and emailing game engines providers. This survey produced quantitative and qualitative data showing the techniques provided for setting the G-factor elements. The second survey of projects that use game engines produced quantitative and qualitative data. The quantitative data revealed the practices used to set the G-factor elements and the qualitative data highlighted the reasons for using game engines and the issues encountered during development. The method used for collecting both types of data relied on published material. The design stage followed an iterative development process. The first iteration developed an architecture in which one element of the G-factor (i.e. the game logic) was moved outside the engine and placed in the hands of a human controller (or a dungeon master) whose role was to monitor the game and trigger events. This was tested in a preliminary experiment which used a multiplayer serious game for traffic accident investigators (BinSubaih et al., 2005a). The second iteration of the architecture created a process to hold the G-factor elements outside the game engine.

For the second part of the thesis, the analysis stage consisted of interviews, observations, role-playing sessions, and examining the course material and accident files. These provided qualitative data highlighting the issues facing the current training methods employed by the Dubai police force. This stage also used quantitative and qualitative data from a preliminary experiment to compare the use of a multiplayer serious game against the use of tabletop training. The iterative design process stage involved the development of a multiplayer accident scenario in the first iteration. This was then modified in the second iteration to a single player environment and added features to assist trainers in evaluating trainees. It also added features to create artefacts that can be used in creating a social ecology (Herz & Macedonia, 2002) outside the game that promotes discussion and continues learning. The second iteration also benefited from the use of focus groups. The third stage conducted a controlled experiment to evaluate the learning effectiveness of SGTAI. The experiment used a pre-test-post-test control group design in which the participants were divided equally based on their performance on the pre-test into

two groups: a control group and a treatment/trained group. After carrying out the training for the trained group both groups were then tested again using the post-test. The difference in performance was used as a quantitative measure of the learning effectiveness. The trained group participants were also given a Presence questionnaire (Slater, 1999) and were asked to specify their likes and dislikes, and ways to improve the environment. In addition, their navigational behaviour was observed.

1.3 Thesis Structure

Chapter 2, G-factor Portability in Game Engines, provides an overview of the historical evolution of game development and the aspects of portability supported by game engines. What limits the ability to port the G-factor is then presented in a structured review that contains the findings from two surveys. The first survey identifies the methods game engines provide for creating the G-factor elements and how they affect the G-factor portability. The second survey reveals how the projects that utilize game engines tend to specify the G-factor elements and the issues developers come across when using game engines. A sample game is then presented to highlight the issues with a typical game development approach.

Chapter 3, A New Architecture for Serious Games, describes a new game space architecture (GSA). The decisions made to develop the architecture are presented and the trade-offs made are discussed. The development of the sample game from chapter 2 is then contrasted to the development approach when using GSA.

Chapter 4, Evaluating the Game Space Architecture, evaluates GSA by following two types of evaluation: unstructured and structured. The unstructured evaluation throws challenges (e.g. testing scalability by developing a real world application for police training) at the architecture to find out how GSA can cope with them. The structured evaluation uses the architecture trade-off analysis method (ATAM).

Chapter 5, Serious Games and Learning, provides an overview of serious games covering what the concept means, the domains that have benefited from their use, the traits that makes them effective for educational purposes, and the challenges facing their design and development. Also discussed are the learning theories that have been found to be able to explain how learning occurs in serious games.

Chapter 6, A Serious Game for Traffic Accident Investigators, demonstrates how GSA can be used to develop a serious game for traffic accident investigators (SGTAI). The learning principles upon which SGTAI is built are described. The experiment conducted in 2006 in the Dubai police force for 56 officers is detailed. An experiment divided officers into two groups: novices and experienced investigators. The learning effectiveness for each group is measured and the findings are analysed and discussed. Also discussed are the effectiveness of SGTAI in achieving its learning objectives, the limitations, the lessons learned, and the implications for policy makers, educators, and researchers.

Chapter 7, Conclusions and Future Work, provides an overview of the conclusions and directions for future work. Table 1.1 gives an overview of the appendices.

Table 1.1: The appendices.

Appendix	Description
Appendix A: A Survey of Projects that have used Game Engines	Lists the table of projects that made use of game engines.
Appendix B: Adapter	Illustrates the adapter codes for the traffic investigation challenge, Moody NPCs game, and SGTAI.
Appendix C: Building SGTAI	Presents the implementation of SGTAI using GSA and the Torque game engine.
Appendix D: Pre- and Post-Tests	Describes the tests used in the experiment presented in chapter 6.

1.4 Main Thesis Contributions

The main contributions of this thesis are:

- An analysis of the issues that make a game dependent on the engine it was developed on. This is backed by the findings from qualitative and quantitative surveys of game engines and projects that use game engines. These findings have been compiled in a technical report (BinSubaih et al., 2007).
- A demonstration of the ability to make the G-factor (game logic, object model, and game state) portable through the development of a new architecture. The novel design approach combines a variant of the model-view-controller (MVC) pattern to separate the G-factor (i.e. model) from the game engine (i.e. view), with on-the-fly scripting to enable communication through an adapter (i.e. controller). This work has been published in (BinSubaih et al., 2005b; BinSubaih et al., 2006a; BinSubaih & Maddock, 2007).
- An analysis of the issues currently facing the training methods employed by the Dubai police force in the traffic investigation field. This has been presented internally to the Dubai police force and has been published in (BinSubaih et al., 2005a).
- A novel environment for training traffic accident investigators in the Dubai police force which enables investigators to virtually experience accident investigation without the restrictions facing the current training methods. SGTAI has been published in (BinSubaih et al., 2006b, BinSubaih et al., 2008).
- A demonstration of the ability of SGTAI to improve learning for novice and experienced investigators in the Dubai police force. The empirical study has been published in (BinSubaih et al., 2006b) and the way instructional design has been used alongside game design to ensure learning is embedded in the gameplay will be published in (BinSubaih et al., 2008).

2. G-factor Portability in Game Engines

2.1 Introduction

The shift in game development from developing games from scratch to using game engines was first introduced by 'Quake'[1] and marked the advent of the game-independent engine development approach (Lewis & Jacobson, 2002). In this approach the game engine became "the collection of modules of simulation code that do not directly specify the game's behaviour (game logic) or game's environment (level data)" (Wang et al., 2003). This makes the game engine reusable for (or portable to) different game projects. However this shift produces a game which is notoriously dependent on the game engine. For example, why can't a player play 'Unreal'[2] on the Quake engine or Quake on the Unreal engine?

Hardware and software abstractions have facilitated the ability to play a game on different hardware and on different operating systems (in some cases with some modifications). These abstractions have also facilitated the ability to use data assets such as 3D models, sound, music, and textures across different game engines. This ability should also be extended to allow for the game itself to be portable. The goal is to make the game engine's brain portable, where the brain holds the game state and the object model and uses the game logic to control the game. The game state holds the working memory of the game at any moment, and contains game objects (e.g. players and non-player characters (NPCs)), time, and logged interactions. The object model describes the classes for the objects in a game (e.g. players and interactions). The game logic dictates the game behaviour. In this thesis, these three things are collectively referred to as the G-factor. The portability of the G-factor is the next logical step in the evolution of game development. Following Lewis and Jacobson's terminology (Lewis & Jacobson, 2002), an approach that promotes G-factor portability can be called the game engines independent game development approach. Figure 2.1 illustrates the evolution of game development and highlights the issues facing each approach.

A benefit of making the G-factor portable would be to encourage more developers to make use of game engines, since a particular game engine's future capability (or potential discontinuation, as was the fate of Adobe Atmosphere which was used for 'Adolescent Therapy – Personal Investigator' (Coyle & Matthews, 2004)) would not be a worry as a different game engine could easily be substituted. This problem has recently been referred to as "the RenderWare Problem" (Carless, 2007) after the acquisition of RenderWare engine by Electronic Arts (EA) and its removal from the market. Rewriting the G-factor from scratch every time a game is migrated from one engine to another is similar to the undesired practice of developing games from scratch which was deemed unfeasible and resulted in the advent of game engines.

---

[1] http://www.idsoftware.com/games/quake/quake4 (accessed 23/8/2007).

To identify the extent of the portability problem in game development in general, and game engines in particular, two surveys will be presented in this chapter. The first will consider game engines. The objective of this survey is to illustrate the effects the development practices encouraged by game engines have on the G-factor elements. The second survey will examine how portable the G-factor is for projects that use game engines.

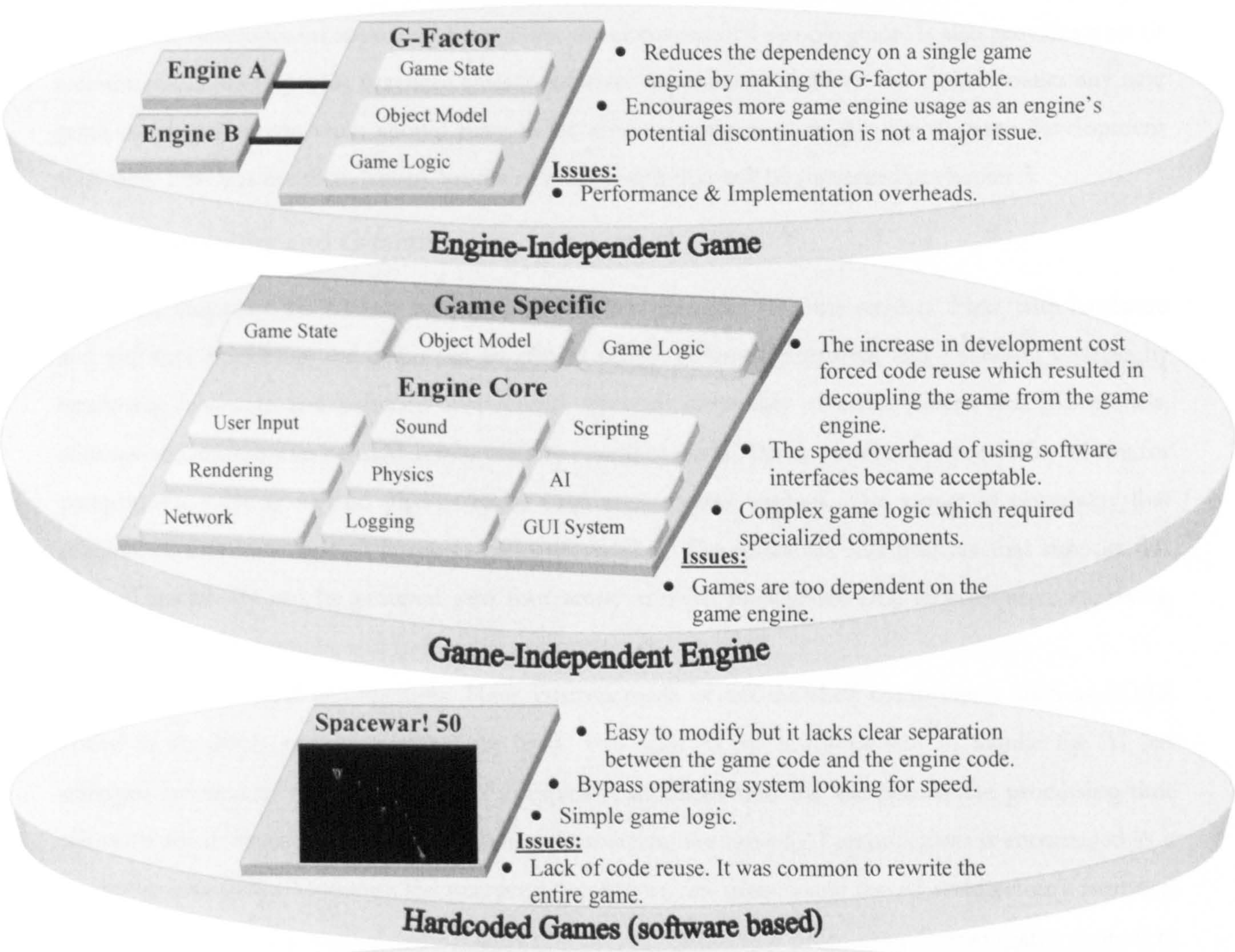


Figure 2.1: Game development evolution.

---

[2] http://www.epicgames.com (accessed 23/8/2007).

Section 2.2 describes the aspects of portability in relation to game engines and the techniques that have been tried to aid G-factor portability. Section 2.3 describes the governing variables and how they affect the G-factor implementation and how they can be used to create a categorization for game engines based on how they promote portability. Section 2.3.4 presents the findings of the first survey which is conducted to identify the effects the development practices encouraged by game engines have on the G-factor elements. Section 2.4 presents the second survey which examines the common practices followed by projects using game engines to illustrate how portable the G-factor is for these projects. Section 2.5 demonstrates the over-dependency of the G-factor on the game engine associated with a typical game development approach through the development of a sample game. It also provides a set of recommended development practices to ease the over-dependency and lists the characteristics any new game development approach should have to be able to compete with the typical game development approach. This lays the foundations for the new approach that will be presented in chapter 3.

2.2 Portability and G-factor

Figure 2.2 illustrates the current aspects of portability addressed in game engines. First, with hardware and software portability the game can be played across different platforms and operating systems by employing hardware and software abstractions. Second, portability of assets means that 3D models, textures and sounds can be used across different game engines. Third, middleware portability allows for components such as AI and physics to be used across game engines. The aspect of portability that requires further investigation is the G-factor portability. The initiatives and projects that support this kind of portability can be grouped into four areas: artificial intelligence (AI) architectures, interfaces, standards and file formats, and frameworks or protocols.

The first area is AI architectures. Here, custom made or off-the-shelf components such as SOAR (Laird et al., 2002) and AI.Implant[3] are used. The need to use a component to handle the AI has emerged because of the increase in AI complexity in games and the increase in the processing time allocated for it. From a software engineering perspective, the use of AI architectures is encouraged as it promotes reusability. However, the practice of specifying the game using the AI architecture's format is not what is eventually wanted since this merely moves it from one proprietary format (game engine) to another (AI architecture). Nevertheless it is a step in the right direction of moving the game away from the game engine's format. The architectures that promote portability more than others are those that allow complete removal of the game from the game engine, such as TIELT (Aha & Molineaux, 2004). Others that only partially remove the game are obviously less portable, such as Mimesis (Young et al., 2004) and MissionEngine (Vilhjalmsson & Samtani, 2005). The AI architectures promote the use of their own proprietary format which is similar to what game engines do. Furthermore suggesting a monolithic architecture as a complete entity is not what is needed. Instead initiatives must examine the causes of the

[3] http://www.biographictech.com (accessed 5/5/2007).

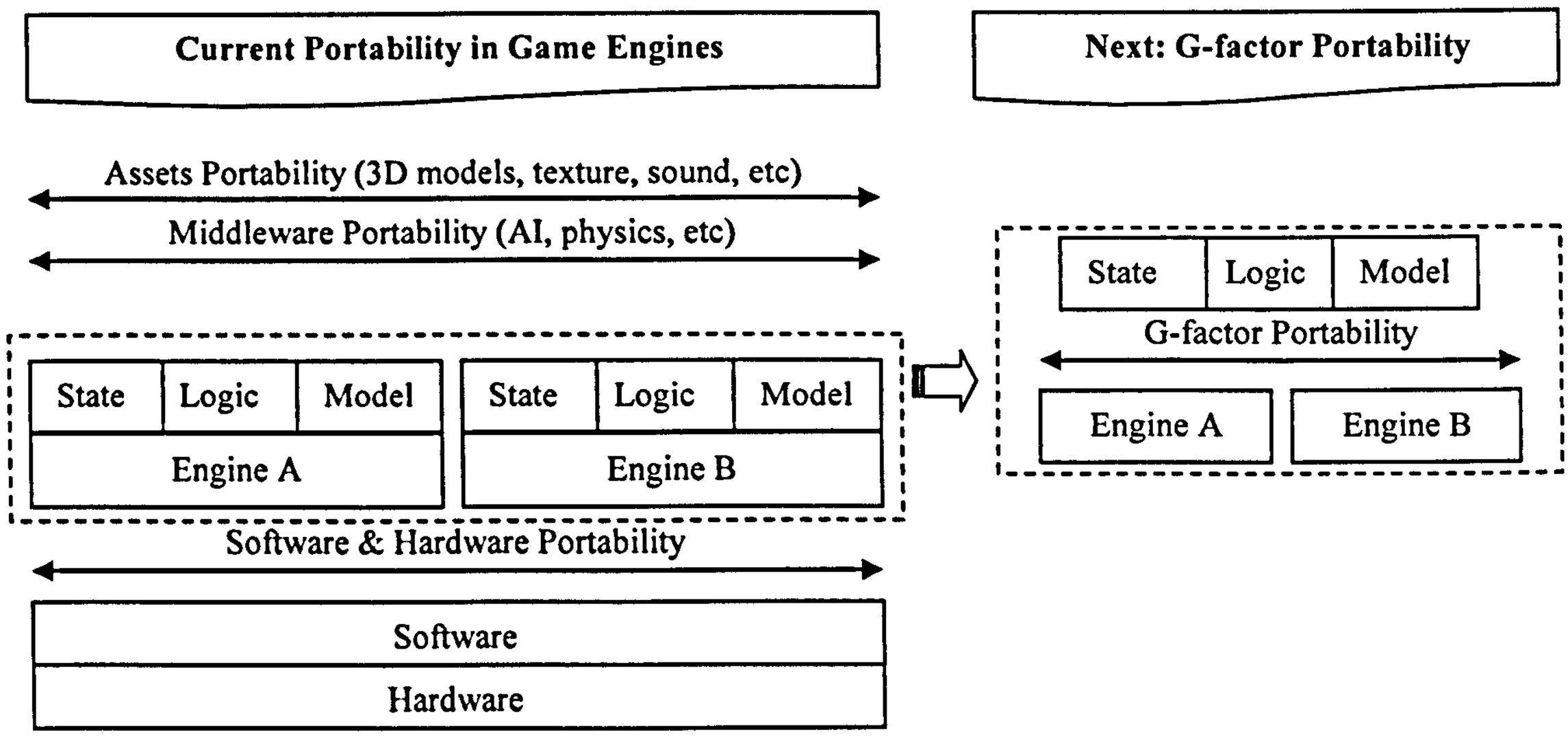


Figure 2.2: Portability in game engines.

G-factor portability problem and provide practical solutions that can be employed even if their architecture or middleware is not chosen. Section 2.5.2 will show the causes of the tight-coupling between the G-factor and the game engine and chapter 3 will describe the practices used in this thesis to address these causes.

The second area is interfaces. Here, the aim is to provide access to external programs, and in game engines two types of interfaces are found: specific and common. These provide access to the G-factor elements. A number of interfaces have been developed to provide access to specific game engines. For example the interfaces that have been used to access Unreal are Gamebots (Adobbati et al., 2001) and GOLOG Bots (Jacobs et al., 2005). Similarly, Quakebot (Laird, 2001) can be used to access Quake, FlexBot (Khoo et al., 2002) can be used to access Half-Life, and Shadow Door (Hussain & Vidaver, 2006) can be used for Neverwinter Nights. These provide interfaces for specific game engines. Other projects are attempting to provide common interfaces to game engines such as the initiative by the International Game Developers Association (IGDA) for world interfacing (Nareyek et al., 2005) and OASIS (Berndt et al., 2005). Such interfaces, however, may have more success in the serious games community rather than in the fast-evolving games industry.

The third area is the standards and file-based formats such as VRML/X3D[4]. These still lack the maturity needed for game development. For instance VRML lacks the rendering capability required. It also suffers from speed and security issues (Jankovic, 2000).

The fourth area is the frameworks or protocols that aid interoperability between different simulations like the High Level Architecture (HLA) (Smith, 1998) and Java Adaptive Dynamic Environment (JADE)

[4] http://www.web3d.org/x3d/ (accessed 5/5/2007).

(Oliveira et al., 2003). Despite the fact that this category focuses more on the interoperability between simulations and less on how the game is linked to the simulation it is mentioned here to illustrate that portability exists at different levels. HLA for instance promotes it at the simulation and object level and JADE promotes interoperability at the functionality level. HLA identified the simulation functionality that is generally required across all systems and thus should not only be part of a single simulation system but available for others. To achieve this it moved the general simulation functionality from the simulation system to the HLA infrastructure and thus made the simulation functionality accessible to other simulation systems (Smith, 2000). An example of the functionality provided is object management which is used to share object instances between different simulations. JADE was designed to address the monolithic nature of current Virtual Environment (VE) systems. Oliveira et al. argue that in current VE systems it is not possible to replace or increment the necessary functionality. JADE proposes to host Modules without concern for their functionality which is the responsibility of the VE developer. A Module can encapsulate an entire system or a block of code and thus can be reused by others. The first challenge facing frameworks and protocols is that they require the projects to comply with their infrastructure to be able to interoperate with other systems. The other challenge facing them is to create a generalizable infrastructure to support any kind of environment (Kapolka, 2003).

This section has presented the different aspects of portability that are supported by game engines and has analyzed what has been done so far to address G-factor portability. The next two sections present two surveys to help better understand G-factor portability and to highlight what is required from a development approach that aims to promote G-factor portability.

2.3 Survey of G-factor in Game Engines

The objective of this survey is to discover the development practices encouraged by game engines through examining the tools they provide (e.g. a scripting language) to specify the G-factor. The aim is also to create a categorization that groups engines by the way they promote G-factor portability.

Categorizations of current game engines listed by researchers include ones based on the player's point of view (Stang, 2003) or based on the game genre (e.g. action, strategy, sports, simulation, etc). Another categorization is one proposed by Young et al. (Young et al., 2004) that is based on the integration between intelligent reasoning capabilities and game engines. Young et al.'s categorization is divided into three groups: mutually specific, AI specific, and game specific. In the mutually specific category the essence is on creating new functionalities using specific intelligent reasoning tools or techniques (such as planning algorithms) for a specific game engine. This can be described as having a one-to-one relationship between the new AI functionality and the game engine. In the second category, AI specific, a set of AI functionalities can be used across a range of game engines – a one-to-many relationship between the AI functionalities and game engines. The game specific category allows more than one AI element to be used on a specific game engine. This is a many-to-one relationship between the AI elements and the game engine.

The categorization presented in this section focuses on the relationship between the G-factor and the game engine in order to design an architecture that is able to support portability. The architecture also needs to support two important quality attributes in any game development approach, namely: modifiability and performance. Three variables are selected which relate to these three aspects.

The first variable selected to govern the relationship between the G-factor and the game engine is 'location'. This concerns whether the engine forces the G-factor to be located inside it or outside it, which is correspondingly dictated by whether the engine follows a hard-coded or data-driven approach. The location variable was selected because it supports the portability aim of moving the game outside the engine. It also corresponds to providing a clearer separation between the game and the engine which the evolution of game development has followed (see Figure 2.1) and is recommended by game developers (see section 2.3.1). Section 2.3.1 describes the location variable and highlights the suitability of using a data-driven approach to provide better portability and modifiability.

The second variable selected to govern the relationship concerns the type of support the engine provides for setting the object model (i.e. the classes for the objects in the game) which can either be static or dynamic (see section 2.3.2). This variable was selected because an engine that supports a dynamic object model would promote better portability since the object model is independent from the engine's implementation language. In addition, a dynamic object model should provide better modifiability since it is not intertwined in the code and thus easier to understand and modify.

Finally, the third variable selected to govern the relationship concerns whether the engine allows the game logic to be set by precompiled scripting languages or compiled and interpreted on-the-fly scripting languages. The scripting constraint variable was selected because of the need to maintain good modifiability. Section 2.3.3 describes this variable and its role in supporting portability. This third variable, along with the first two, has an impact on performance. The measures taken to lessen this will be discussed in the following sections.

2.3.1 Game Location

Location refers to whether the engine promotes hard-coded or data-driven approaches to create the G-factor. The hard-coded approach does not meet the current dynamic game design requirements since embedding the game too deeply in the code is very restrictive as it shields it from the designers and artists (Keith, 2003; Schertenleib, 2006). Keith also reports another problem with this approach which is the over dependency on the object hierarchies for behaviour which makes the code fragile and very difficult to maintain. He gives the example of "moving AI behaviour around the hierarchy until you end up with AI behaviors in base classes or a great deal of cut-and-pasted code." This problem was also mentioned by Bilas (Bilas, 2002) who noted that the line between the content and the engine keeps moving as the requirements get fuzzier and advised a change to a data-driven development approach, warning that resistance would only cause regular refactoring.

The data-driven approach allows the data to be defined by configuration files and/or scripts (Schertenleib, 2006) and these are then fed into the program to dictate its flow. The need for the game engines to be extremely flexible is the reason why it is crucial to have a data-driven design focus where the game is controlled by data which resides outside the engine (Tong, 2003). The advantages alongside the aforementioned flexibility are: extensibility and improved process (Fermier, 2002). The disadvantages are performance, its too powerful (Tapper, 2003), there is more work up-front (Leonard, 1999), over-engineering and lack of ownership (Fermier, 2002), and difficulty in debugging (Wilson, 2003).

The advantages of the data-driven approach outweigh the disadvantages as reported by the developers of a number of commercial games. The developers of 'Gabriel Knight 3' (budget over $4.5 million, development time almost 3 years) reported that the initial hard-coding of the story sequence of the game in C++ meant that engineers were creating content instead of working on the engine and also that the tiniest changes to the game required recompilation which "made the development process unbelievably inefficient." (Bilas, 2000). Similar problems were reported by the developers of 'Thief: The Dark Project' (budget approximately $3 million, development time 2.5 years) who also moved to adopt the data-driven approach (Leonard, 1999). The developers of 'Jurassic Park: Operation Genesis' (development time 22 months) said that the data-driven approach they used required initial investment but the time spent was saved many times over and it opened up the possibility of creating expansion packs (Chan et al., 2003). They also reported that "the data-driven approach worked so well that through much of our development, Thief and System Shock 2 (two very different games) used the same executable and simply chose a different object hierarchy and data set at run time".

From the portability point of view the separation encouraged by a data-driven approach allows for a clearer specification of the boundaries between the data and the system, thus making it more modular. A game that is represented by data is much easier to manipulate and understand than one which is intertwined in the application code. Therefore, any technique that moves the game away from the engine is beneficial to the G-factor portability cause. Moreover, it also allows for the creation of intuitive tools (Shumaker, 2002) for manipulating the data, thus increasing modifiability.

2.3.2 Object/Class Model

The object model describes the classes for the objects in a game. These objects can be divided into two types: game objects and decorative objects. Game objects represent all non-terrain and interactive logic content (Bilas, 2003) and they are the ones that are of interest to the G-factor. The decorative objects, such as terrain, sky, etc, are merely used to enhance the look of the environment. The object model used can either be static or dynamic.

A static object model has hard-coded representation and cannot be modified at run-time. For instance, a new object type (or class) cannot be added without having to modify the hard-coded representation and then recompiling and loading the application (e.g. Java is an example of static object model). The problem with this is highlighted by the development of 'Ultima Underworld 1' (Duran,

2003). Initially the development started under the impression that the non-player characters (NPCs) and doors do not share many components. Later on, the designer wanted to allow a player to have a conversation with a door just as he can have a conversation with NPCs. But, since the initial design only allowed NPCs to have the conversation component, they found that pushing the component up the hierarchy was very difficult and had to use a hack around the problem. Similar lessons were learnt by the developers of 'Dark Engine' (Leonard, 1999). The success of that was demonstrated by the ability to have no code-based game object hierarchy of any kind in 'Thief'. This was handled through a general database where an object can possess properties and hold relations with other objects.

A dynamic object model allows the creation and modification of classes along with their properties and hierarchies dynamically. The advantages and disadvantages of using a dynamic object model pattern are clearly described by Riehle et al. (Riehle et al., 2005). The primary advantages that aid portability are: end-user configuration, language independent, run-time object type creation, and explicit model. The end-user configuration ability means that the game developer or designer is able to define concepts from his domain (c.f. ontologies (Chandrasekaran et al., 1999)) and does not have to hard-code them. This means the object model can exist outside the game engine and more importantly is modifiable independently of the engine. This promotes flexibility and extensibility. The second advantage is being specified in a language that is independent from the implementation language since the object model can be stored outside the application in a file or a database, which makes it easier to port between engines of different implementation languages. It also simplifies sharing the object model between games. Run-time object type creation is important for games with persistent worlds like the massively multiplayer online games (MMOG) (e.g. 'Toontown' (Goslin, 2004; Goslin & Mine, 2004)). The final advantage is the explicit model provided by the dynamic object model, which enables querying of the object model to find the classes and their properties, property type, inheritance, etc.

The potential disadvantages of using the dynamic object model pattern are the performance and memory usage penalties associated with it. However, the use of it in industry by games such as 'Thief' shows that it is not undermining the game to the point of making it unplayable. Another disadvantage is that it requires extra work initially to create the framework that is going to hold the dynamic object model. For systems that do not provide a dynamic object model there is a workaround which involves constructing classes dynamically by using on-the-fly scripting languages (described in the next section). These languages can be grouped into two categories: class-based (e.g. Python) and prototype-based or instance-based (e.g. JavaScript). The difference is that in the prototype-based approach there are no distinct entities for classes and instances. The prototype-based approach makes sharing the classes more cumbersome and counterintuitive to developers familiar with object-oriented programming since the class description is embedded in the instance which blurs the separation object-oriented developers are accustomed to.

2.3.3 Game Logic Scripting Constraint

The third variable to govern the relationship between the G-factor and the game engine is the language processing constraint. As game development moves away from code-driven approaches to a data-driven approach it makes the data more complex to represent and manipulate. What is needed is a simpler approach than the code-driven approach but one that still retains some, if not all, of its flexibility and power. Scripting is an answer to this. Scripting is a programming language that is similar to coding but is generally simpler and also requires a shorter edit-compile-link-run process[5]. Examples of scripting languages are Python, Ruby, and Lua. They share a number of characteristics (Garces, 2006) such as: they are high-level languages, they provide flexible flow control, and they are interpreted languages (not compiled into machine code). Although scripting uses code as the basis for its representation it is considered to fall into the data-driven category (Schertenleib, 2006). Many game development teams found in scripting an ideal solution to the programmer bottleneck problem[6] as was stated by the developers of 'Treyarch's Draconus' (Fristrom, 2000). Despite the known performance issue with scripting, the developers of 'Centipede 3D' (Rouse, 1999) and 'Shiny's Wild 9' (Malenfant, 2000) found that the trade-off for scripting flexibility and ease of use over performance was a positive move. LaMothe (LaMothe, 2002) estimated that about 99% of all commercial games use scripting. The survey in section 3.4 puts this figure at 74.4%.

Scripting languages can either be precompiled or compiled and interpreted on-the-fly. Precompiled means the code is compiled before the game starts whereas on-the-fly means compiling happens at run-time. This makes the on-the-fly feature very useful for programs that cannot afford to make the application offline such as Massively Multiplayer Online Games (MMOG). However these languages run slower than the precompiled ones. Despite this many developers think the trade-off is worthwhile. The developers of 'Pirates of the Caribbean – Battle for the Buccaneer Gold' (Schell & Shochet, 2001) found on-the-fly scripting very valuable to conduct guest testing. They used the Scheme scripting language to be able to reprogram the game while the guests were live in the game. The MissionEngine (Vilhjalmsson & Samtani, 2005) architecture found in on-the-fly scripting an ideal solution to avoid making the architecture too rigid and too slow to respond to design changes. The dynamic nature of the language (Python) used by the architecture meant that the class definitions in the architecture did not have to be changed every time the data format changed when new features were requested. However that was not the case with the second scripting language they used because they chose Unreal engine. Unreal provides UnrealScript which requires precompiling. They found it to be less flexible than Python as for every change to the page type in the skill builder a new class in UnrealScript had to be created.

For portability, on-the-fly scripting plays a vital role. The first role is to facilitate the dynamic object model workaround described in the previous section. The second role of scripting is to enable

---

5 http://en.wikipedia.org/wiki/Scripting_language (accessed 5/5/2007).
6 When designers and artists are over dependent on programmers for changes they require.

translation through the use of a script mapping technique, as described in BinSubaih and Maddock (BinSubaih & Maddock, 2006). The third role is to avoid undermining the current flexibility associated with programming the game engine directly without any restrictive layers. For instance, Gamebots uses predefined text-based protocol messages to interact with the game engine to receive sensory information (synchronous and asynchronous) and send actions (e.g. CHANGEWEAPON, RUNTO, JUMP, STOP, etc). A project for teaching Bayesian behaviors to game characters (Le Hy et al., 2004) made use of Gamebots and found it to be restricting the interaction they could have with the game engine. TIELT requires adding the actions and sensors that have to be exchanged between the game engine and the decision system to the knowledge bases residing inside TIELT. In a project (Ponsen et al., 2005) that used TIELT for integration with Stratagus, which provides an on-the-fly language (Lua), it was found that every time a new action was needed the knowledge base had to be updated to allow that. This shields the on-the-fly language from the decision system undermining the power of the language. Another problem with TIELT, also shared by the protocol messages of Gamebots, is that they introduced their own scripting languages which is not ideal, as shall now be explained.

Developers wanting to add scripting support to their architecture are faced with two options: either to build their own scripting language or use one from the off-the-shelf languages available. Tong (Tong, 2003) noted that as people stop wanting to spend resources on developing their own specific scripting languages a more common option is to leverage the use of existing languages. The advantages to be gained from doing so are: having a rich feature set with plenty of documentation, utilizing a wealth of existing tools, simplifying the interface with the engine code, and utilizing fast and efficient code. The disadvantages are: performance, interface between C/C++ and the scripting language can be constraining, lacks good debugging and development tools, and lack of easily available libraries and extensions. Examples from the industry also echo Tong's call. The developers of 'Gabriel Knight 3' recommend using an existing language to avoid spending time creating documentation of the syntax and training scripters (Bilas, 2000). A more forceful example was cited by the 'Toontown' developers who had to change the scripting language after more than six months into the project (Goslin, 2004; Goslin & Mine, 2004). The issue with their own proprietary language was to do with performance and code management which forced them to switch to an existing language (Python).

2.3.4 Categorization

Table 2.1 describes the categorization created for the game engines using the three governing variables (location, object model, and scripting constraint) described in the previous subsections. For simplicity and clarity purposes no category is created for game engines that might support two properties of the three governing variables. For example, if a game engine locates the game inside it (hard-coded) and can read it from outside (data-driven) it is categorized with the most superior property – outside is superior to inside, dynamic object model is superior to static object model, and on-the-fly language is superior to precompiled. The superiority-deciding factor is based on how it promotes G-factor portability. Based on

Table 2.1: Categories of game engines.

Category	Location	Object Model	Scripting Constraint	Portability
Serviced-dynamic	Data-driven	Dynamic	On-the-fly	↑
Serviced-static	Data-driven	Static	On-the-fly	
Loaded-dynamic	Data-driven	Dynamic	Precompiled	
Loaded-static	Data-driven	Static	Precompiled	
Hard-coded-dynamic	Hard-coded	Dynamic	No scripting support	
Hard-coded-static	Hard-coded	Static	No scripting support	

this, six categories for game engines were created: serviced-dynamic, serviced-static, loaded-dynamic, loaded-static, hard-coded-dynamic, and hard-coded-static. The portability column in Table 2.1 indicates the direction of increased portability support. Table 2.2 shows the engines surveyed and the category they belong to. The table also includes two columns for the tools provided by the engine (world builders and scripting languages used) and a column for the game engine cost. These were added to the survey to help explain the popularity of a particular engine.

Table 2.2: A survey of game engines[7].

Seq	Game Engine	Category	World Editor	Scripting Language	Cost
1	Panda3D 1.2.3§[8]	Serviced-dynamic*[9]	√	**Python**	Free
2	Torque Game Engine 1.4‡[10]	Serviced-dynamic*	√	**TorqueScript**	$150 - $340
3	Nebula Device 2§	Serviced-dynamic*		**Lua, Python, Ruby, TCL, etc**	Free
4	Delta3D 1.3.0	Serviced-dynamic*	√	**Python**	Free
5	Luxinia	Serviced-dynamic*	√	**Lua**	Free - €100
6	C4 Engine‡	Serviced-dynamic*	√	**Graph-based**	$100
7	CryENGINE 2	Serviced-dynamic*	√	**Lua**	
8	Crystal Space 3D 1.0 §	Serviced-dynamic*		**Python, Java, Perl**	Free
9	Unigine v0.4	Serviced-dynamic*	√	**UnigineScript**	$1495 - $19985
10	Deep Creator‡	Serviced-dynamic*	√	**Lisp**	$1,995
11	Beyond Virtual‡	Serviced-dynamic*	√	**AngelScript**	$99-$155
12	Jet3D	Serviced-dynamic*	√	**Lua**	Free
13	Sylphis 3D	Serviced-dynamic*	√	**Python**	$122
14	Lawmaker Game Engine	Serviced-dynamic*	√	**Lua**	$149.99 - $7999.99
15	Soya 3D 0.11.2	Serviced-dynamic*		**Python**	Free
16	Shark 3D	Serviced-dynamic*	√	**Perch**	
17	Qube	Serviced-dynamic*	√	**QScript**	Free
18	Stratagus 2.1	Serviced-dynamic*	√	**Lua**	Free
19	Blender 2.43	Serviced-dynamic*	√	**Python**	Free
20	Operation Flashpoint	Serviced-static	√	√	$Game
21	3D GameStudio (A6 Game Engine 6.4)‡	Serviced-static	√	**C-Script**	$49-$899
22	Virtools 4	Loaded-dynamic	√	**VSL**	$9,500
23	Unity1.5‡	Loaded-dynamic*	√	**C#,JavaScript, Boo**	$250 - $1,499
24	DOOM 3	Loaded-static	√	**SCRIPT**	$Game
	DOOM	Hard-coded-static	√		Free
25	Quake III	Loaded-static	√	**QVM files**	$Game
	Quake II	Hard-coded-static	√		Free
	Quake	Loaded-static	√	**QuakeC**	Free
26	Unreal Engine 2.5	Loaded-static	√	**UnrealScript**	$Game -$350,000
27	Power Render 6	Loaded-static	√	**AngelScript**	$150 - $8500
28	Reality Factory	Loaded-static	√	**Simkin**	Free - $149.99
29	Serious Engine 2	Loaded-static	√	**Macro**	$20,000 - $100,000

[7] Sources: SDK, game engines published materials (books websites, etc), forums, emails, and devmaster.net as of 23/Feb/2007.

[8] § One of the top 10 open source engines cited by http://www.devmaster.net/engines/ as of 23/Feb/2007.

[9] * Uses the work around suggested in section 3.3 or an alternative technique to create the dynamic object model.

[10] ‡ One of the top 10 commercial engines cited by http://www.devmaster.net/engines/ as of 23/Feb/2007.

Seq	Game Engine	Category	World Editor	Scripting Language	Cost
30	Quest3D 3.5.2	Loaded-static	√	**Graph-based**	$999-$9,999
31	Aurora Neverwinter Nights 1	Loaded-static	√	**NWScript**	$Game
32	TV3D SDK 6‡	Hard-coded-static			Free - $500
33	Cipher‡	Hard-coded-static	√		$100
34	3Impact‡	Hard-coded-static			$99
35	DarkBASIC Pro‡	Hard-coded-static			$89.99
36	Irrlicht	Hard-coded-static	√		Free
37	OGRE	Hard-coded-static			Free
38	Half-Life 2 (Valve Source)	Hard-coded-static	√		$Game
39	Jupiter EX	Hard-coded-static	√		$10,000 - $50,000
40	Blitz3D	Hard-coded-static	√		$100

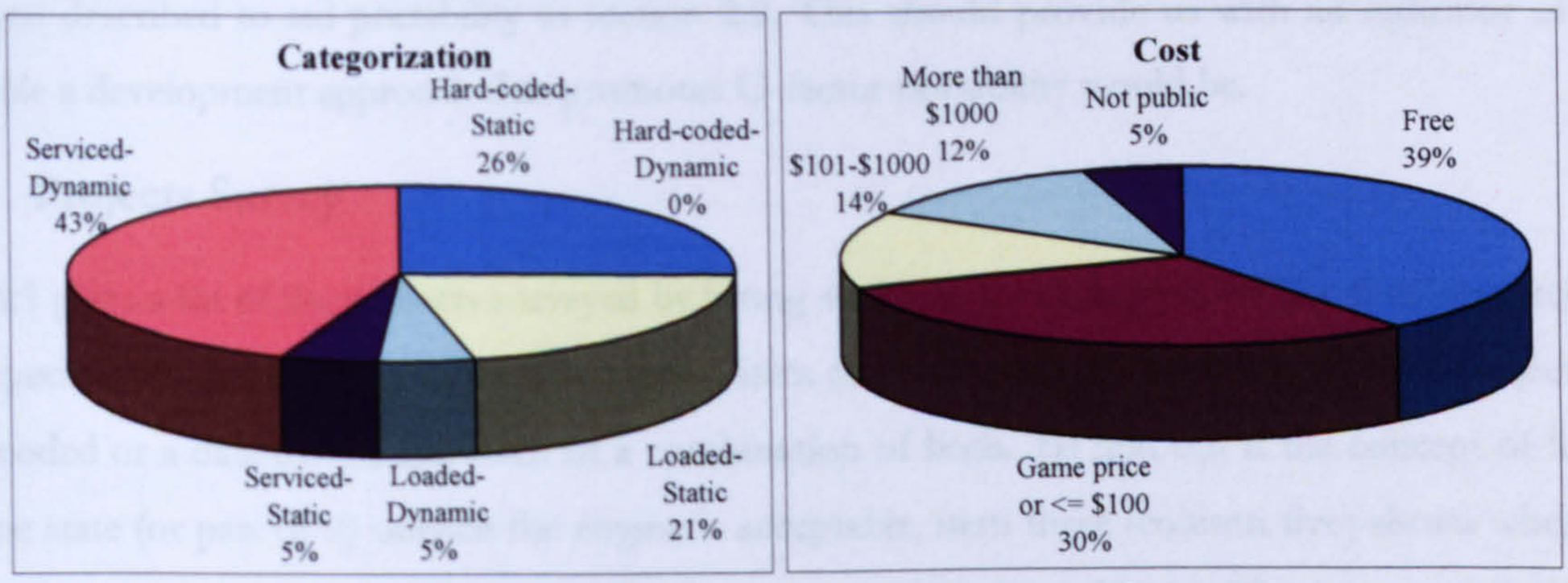


Figure 2.3: Game engines' survey showing the (a) G-factor portability, (b) cost, (c) scripting support, and (d) scripting constraint type.

The findings of the survey are summarised by the four pie charts in Figure 2.3. The categorization chart (Figure 2.3a) shows that 43% of the game engines fall into the serviced-dynamic category. However none of the engines implement the dynamic object model directly and the ones that do have done so either through the workaround using on-the-fly scripting (section 2.3.2) or through different techniques (e.g. 'Deep Creator' allows extending any scene object with metadata which can be created and modified dynamically.). The findings also show that scripting is very popular with 74.4% of the engines supporting it (Figure 2.3c). Figure 2.3d show that "on-the-fly" scripting (48.8%) is more popular than precompiled scripting (25.6%). Finally, Figure 2.3b shows that most (69%) of the game engines surveyed cost $100 or less.

2.4 Survey of Projects Using Game Engines

There are three aims in conducting this survey. First, it aims to examine how portable the G-factor is for projects that use game engines, by checking how they choose location, object model, and language. The second aim is to find out the reasons cited by the projects for using a specific game engine. This should help identify the attributes that increase the game engine's popularity and examine how they affect portability. These attributes should help form the base list of the attributes that should be addressed by any game development approach. Finally, the survey gauges the acceptance of using the approaches that have been described to aid portability in section 2.2. This should provide us with an indicator of how acceptable a development approach that promotes G-factor portability would be.

2.4.1 Projects Survey

Table A.1 gives a list of the projects surveyed by listing six items for each project. The first item (column three) specifies the game engine used. The second item (column four) specifies whether the project uses a hard-coded or a data-driven approach or a combination of both. To find out if the concept of having the game state (or part of it) outside the engine is acceptable, item three (column five) shows where the game state is at run-time (i.e. inside or outside or uses a combination of both). The game state holds the game objects. If these objects are living inside the engine only then are they labelled inside. If they are living outside the engine and have corresponding objects inside the engine then they are labelled outside. Finally if part of them is inside and the other part is outside then they are a combination of both.

The fourth item (column six) describes whether the object model is specific or independent or uses a combination of both. If the object model uses the engine specific model or extends it then it is considered specific. If however it uses its own model independently from the engine's model then it is considered independent. If it mixes both then it is considered to be a mixture of both. The fifth item (column seven) specifies the language used to set the game logic. This can either be specific/custom made (e.g. UnrealScript) or independent/general (e.g. Python) or a combination of both. The last column details the approach used to aid portability.

Figure 2.4 shows five pie charts representing the G-factor location, object model, game language, where the game state is held at run-time, and engine usage. To examine if the results were swayed by Unreal as it was used in the majority of projects surveyed (51%) a balanced table was created which contained one project per engine. This reduced Table A.1 to 10 rows of unique game engines (see Table 2.3). As the listing of the projects in the table was not organized in any way, the first occurrence of the engine was selected and the rest of the projects that use the same engine were disregarded. The results of the balanced table are shown in Figure 2.5. These results confirmed the trend that was exhibited by the previous results (i.e. unbalanced table) which indicated that the majority of the projects surveyed share the same characteristics of: a high tendency to use data-driven approaches, a high tendency to use the

engine's specific object model, a high tendency to use the engine's proprietary language, and a high tendency to specify the game state inside the engine.

Table 2.3: The balanced table of the survey of projects that have used game engines.

Seq	Project	Engine	Location		Game State (run-time)		Object Model		Game Logic Language		Approach
			Hard-coded	Data-driven	Inside	Outside	Specific	Independent	Specific	Independent	
1	Ambush! (Diller et al., 2005)	Operation Flashpoint		√	√		√		√		
2	Tactical Iraqi (TLTS) (Vilhjalmsson & Samtani, 2005)	Unreal Tournament 2003		√	√	√	√	√	UnrealScript	√ (C++, Python, database, and xml files)	Gamebots, MissionEngine
3	Sonocard[11]	Virtools		√	√		√		√ Graphical tools		
4	3D Driving Academy (Traffic AI & Physics engine) (Blackman, 2005)	3D GameStudio (A6 engine)		√	√		√		C-Script		
5	Information and Decision-Making (Creel et al., 2006)	Neverwinter Nights Aurora Engine		√	√		√		NWScript		
6	PSDoom (Chao, 2001)	Doom	√		√	√	√	√	√	√	
7	Visualisation Tools (software Visualization tool and a biomedical visualisation tool) (Wuensche et al., 2005)	Quake 3	√	√	√		√		Shader script		
8	VU-Life 2 (Eliens & Bhikharie, 2006)	Half-Life	√		√		√			√	
9	Stratagus: An Open-Source Game Engine for Research in Real-Time Strategy Games (Ponsen et al., 2005)	Stratagus		√		√		√		√	TIELT
10	NERO project (Stanley et al., 2005)	Torque		√	√		√		TorqueScript		

[11] http://www.virtools.com/applications/simulation-enteccs.asp (accessed 1/3/2007)

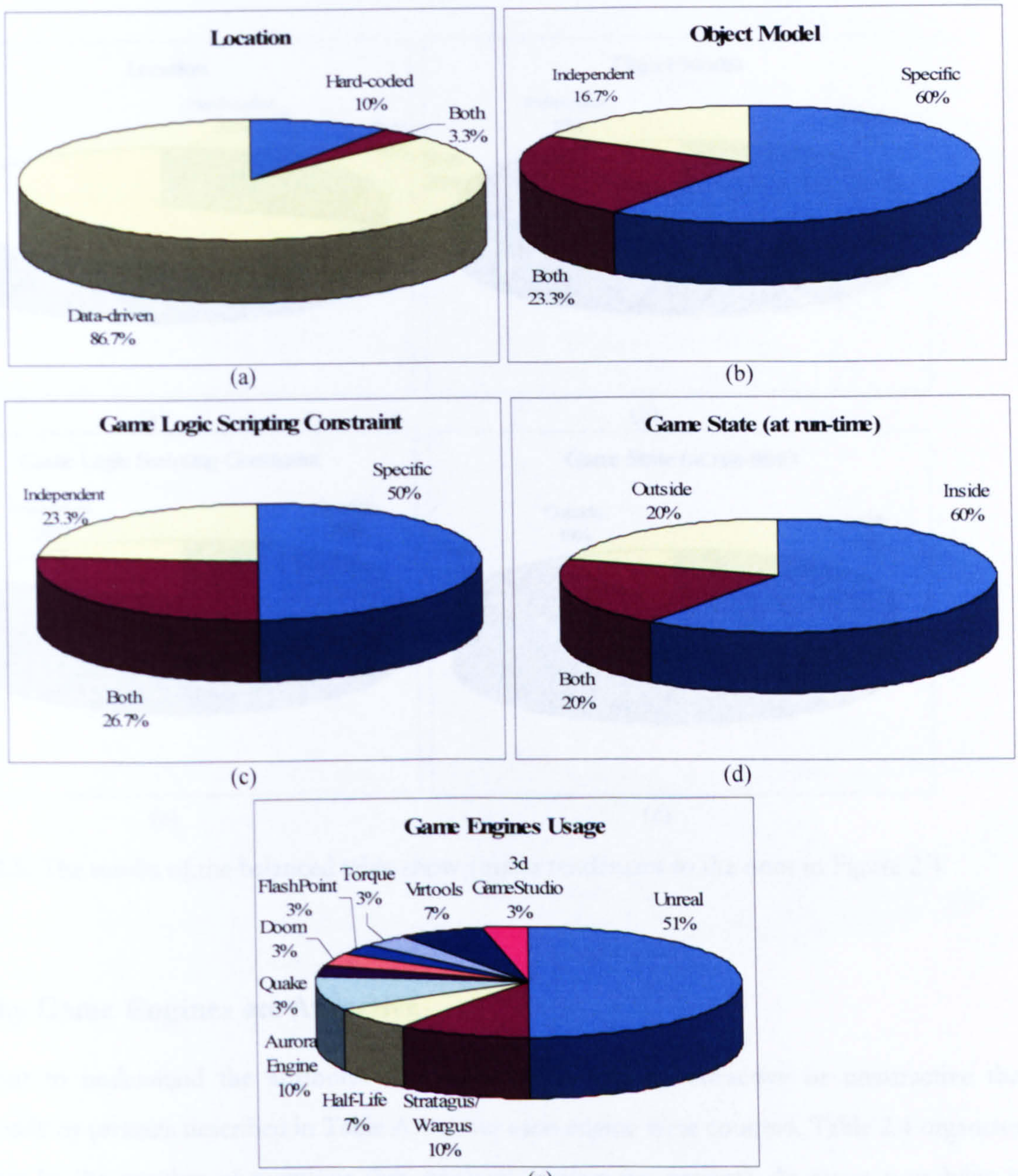


Figure 2.4: Projects survey showing the (a) location, (b) object model, (c) scripting constraint, (d) game state, and (e) game engines used.

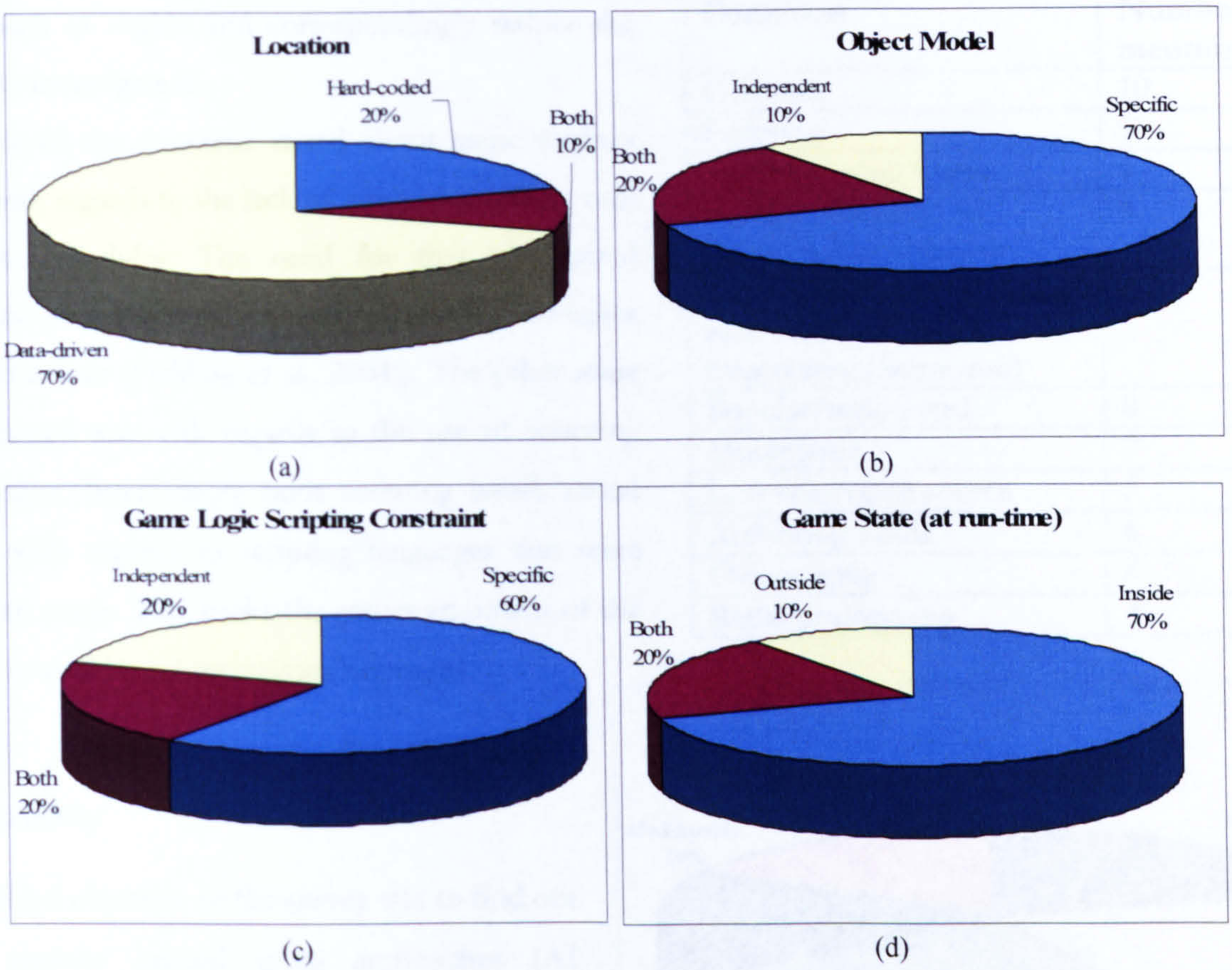


Figure 2.5: The results of the balanced table show similar tendencies to the ones in Figure 2.4.

2.4.2 Why Game Engines are Attractive

In an attempt to understand the attributes that make game engines attractive or unattractive the comments made by projects described in Table A.1 about each engine were counted. Table 2.4 organizes the comments by the number of mentions they received (unique per project). As far as portability is concerned, Figure 2.6 shows the six comments that are of importance to any game development approach that aims to promote G-factor portability. These are the elements that should be guarded as much as possible by any new approach. The pie chart shows the level of importance each holds which should help when trading off one over another when a decision may affect more than one element. For instance, scripting received 22% while performance was not highly mentioned. This makes scripting a high priority attribute. It is also reflected by the examples cited earlier from industry where trading scripting over performance was found to be a positive move (see section 2.3.3). The chart also shows that a small learning curve is also highly regarded. This backs the earlier argument that introducing something completely new (e.g. new scripting language or new standards) might not be the best option and instead any new approach should aim to make use of well-known practices wherever possible. This should also reduce the time it takes to make a decision about a particular approach or engine since

knowing that the basic building blocks have been tried and tested would increase the confidence in that approach or engine and correspondingly reduce the time to investigate it.

One of the concerns raised about game engines was with regards to the lack of integration ability with external modules. The need for that was raised because of the lack of required features (i.e. complex AI behaviour (Fielding et al., 2004)). The other issue mentioned was with regards to the use of scripting languages. Interestingly both scripting issues raised were with regards to scripting languages that were custom made. This backs the earlier argument of the need to avoid creating custom languages.

Table 2.4: Comments ordered by the number of mentions received.

Comment	Number of mentions
Graphics	10
Scripting	9
Small Learning Curve	9
Features (Physics, AI, Statistics, Recordable)	9
Modifiability (configurable, extensible, flexible, integration, abstraction)	8
Popular/well-tested	8
Multiplayer	7
Low cost/open source	7
Authoring Tools	6
Outsourcing	4
Rapid prototyping	3

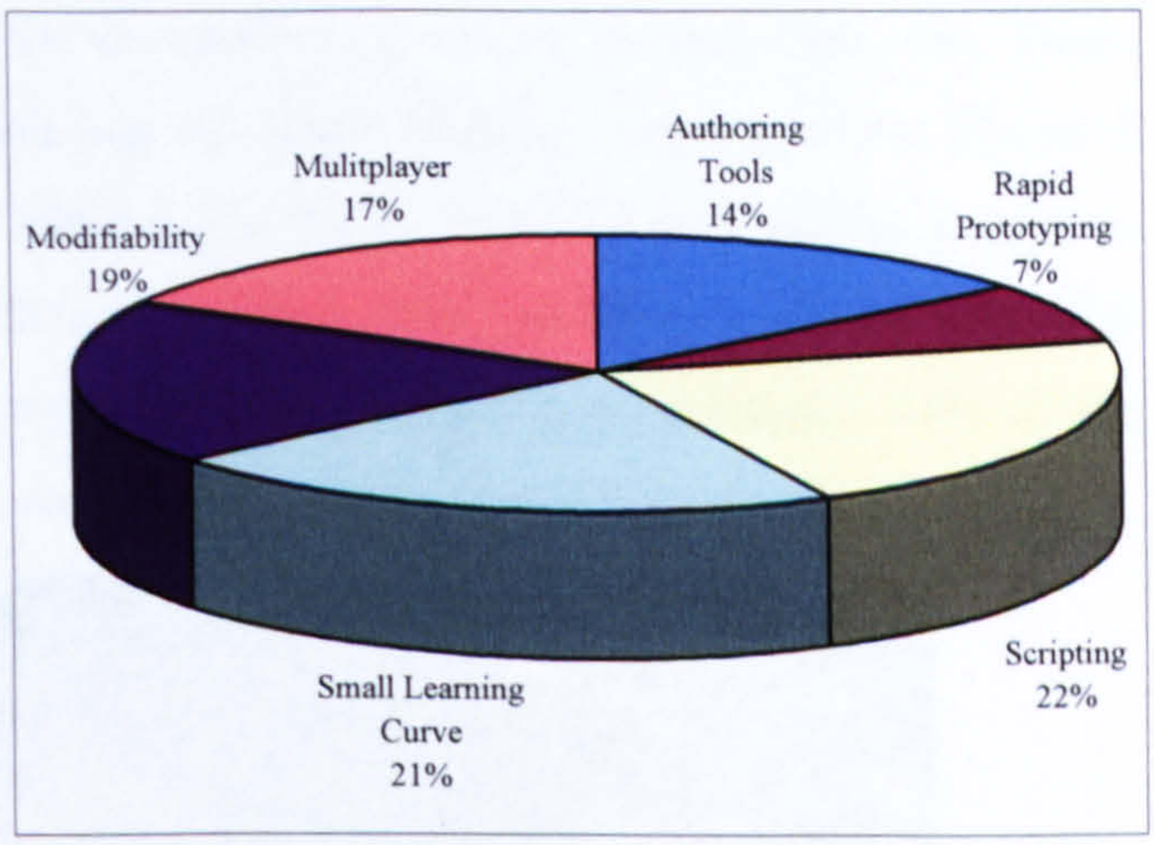


Figure 2.6: Comments made about the features that are important to projects using game engines which any game development approach should aim to preserve.

2.4.3 Usage of Approaches that Aid Portability

The third objective of the survey was to find out the reasons behind using approaches (AI architectures, interfaces, standards and file formats, and frameworks or protocols) that aid portability. The findings show that 30% of the projects described in Table A.1 made use of approaches that fall into the AI and interfaces areas. The primary reason mentioned for adopting these was to integrate the game with external modules. The issue raised of their suitability was with regards to the access restriction they introduced which limits functionalities exposed by the game engine through scripting.

2.5 A Practical Demonstration of the Portability Issues

To demonstrate the dependency of the G-factor on the game engine and provide a diagnosis for it, section 2.5.1 presents an example of game development using a typical game development approach. Section 2.5.2 then highlights the steps that result in the tight-coupling which hinders the G-factor portability. Section 2.5.3 presents recommendations of how to address these issues to simplify G-factor portability.

2.5.1 An Example of a Typical FPS Game Development Approach Using a Game Engine

The practice of using a game engine in game development will be illustrated by a simple game called 'Moody NPCs'. In this game a number of non-player characters (NPCs) react to a player based on their mood. The player can carry out actions such as greeting or swearing. Each NPC reacts to the action based on his mood which is governed by two variables: cowardness/courage and forgiveness/punishment. The game allows the user to navigate the level and click on an NPC which reveals its current mood and the actions available. The player can adjust the mood variables and try out different actions. The Torque game engine is used to demonstrate how the game is developed. The typical game development approach can be grouped into four main steps:

1. Create the game level data: to create the game level data Torque engine provides a level editor called World Editor (shown in Figure 2.7). The level can also be created using other approaches such as scripting, an application programming interface (API), and configuration files. The game level data contains the terrain of the environment and the decorative objects (e.g. houses, trees, etc). These objects can be exported from 3D modeling tools (e.g. 3D Studio Max) to Torque's format. The level also contains location markers for the game objects (e.g. NPCs and player). Scripting is used to create the other game objects (e.g. Reaction, Action, and Interaction) as shown in Figure 2.9c. This approach for creating the game level data is very common amongst game engines – 84% of the engines surveyed provide editors to create the game level.

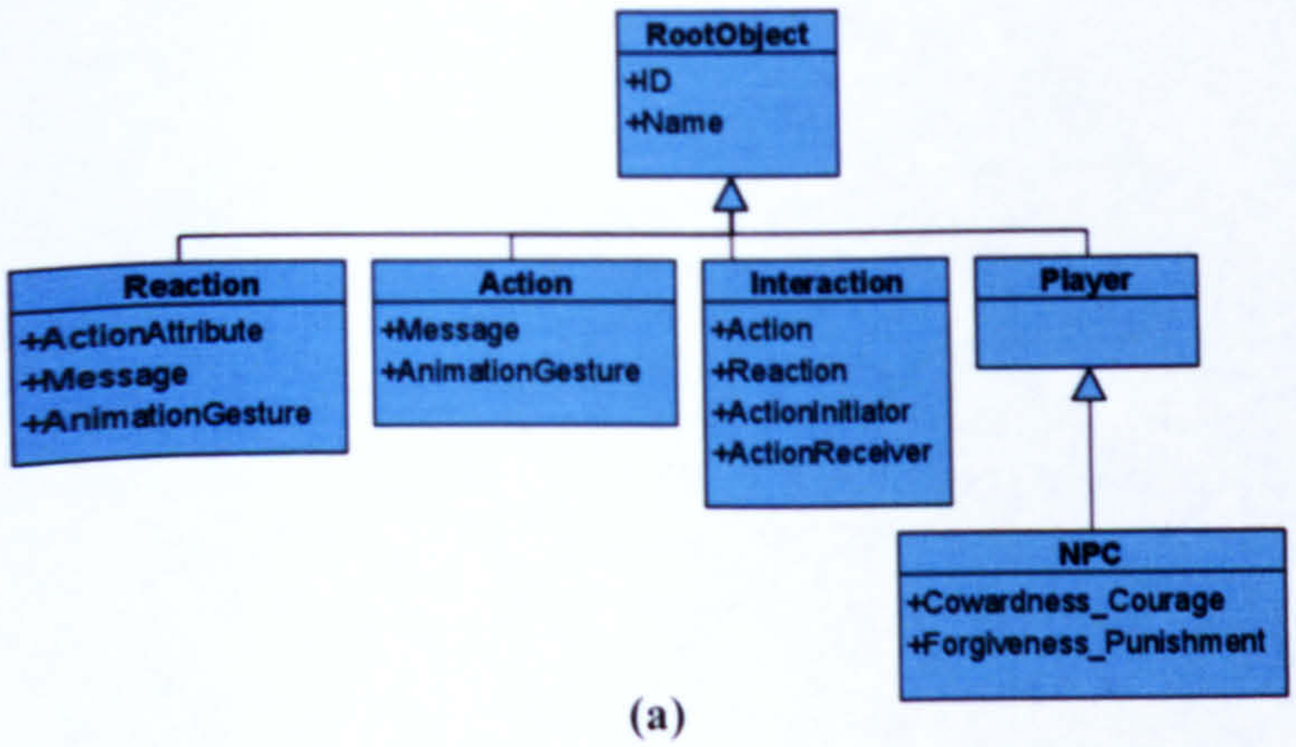
Figure 2.7: Using Torque's World Editor to create the game level.

2. Create the graphical user interface (GUI): Figure 2.8 shows the graphical user interface (created using Torque GUI editor) which has mood variables sliders on the top left corner of the screen and an actions controller on the bottom left corner of the screen. The player can use the keyboard to navigate around and the mouse to select an NPC which reveals its mood variables and actions. As with the game level data the interface can be created by other means such as scripting and configuration files.

Figure 2.8: The interface created using GUI editor.

3. Create the object model: the object model holds the structure for the game objects. The object model consists of five classes (see Figure 2.9a): Player, NPC, Action, Reaction, and Interaction. Torque has a default object model for the player and the AI player. These can be extended to add the properties that are specific to the game (i.e. mood variables for an NPC). The extension and the creation of the other classes can be created using a static object model using either C++ or TorqueScript. The other game object models are created using scripting (see Figure 2.9b for an example).

4. Create the game logic: the game logic controls how the NPC reacts to the player actions and is created using TorqueScript (see Figure 2.9d for an example).



(a)

```
//Object Model: Create the Action object
function
CreateAction(%ID,%Name,%Rate,%Message,%Gesture)
{
  %action = new SimObject()
  {
    ID=%ID;
    Name=%Name;
    Rate=%Rate;
    Message=%Message;
    Gesture=%Gesture;
  };
  $actionsArray.add(%action);
}
```

(b)

```
//Level Data: Add action and reaction game objects
createAction("40","action1","bad","You are an idiot!","looknw");
createAction("41","action2","good","Hi there!","celwave");
createReaction("80","reaction1","40","bad","No you are the
idiot!","looknw");
```

(c)

```
//Game Logic
function
calculateEmotionOutput(%actionType,%forgiveness,%punishment,
%cowardness,%courageness)
 {
   %actionWeight=0;
  if(%actionType$="verybad") {
    %actionWeight=-1;}
  else if(%actionType$="bad") {
    %actionWeight=-0.5;}
  …
```

(d)

Figure 2.9: Developing Moody NPCs using a typical game development approach.

2.5.2 Addressing the Over-dependency on a Game Engine

The typical development approach outlined in the previous section makes the game over-dependent on the game engine. There are three main causes for this dependency. The first dependency occurs because the first step (create level data) creates the game objects using the game engine's proprietary format. This can be addressed by moving the game objects outside the game engine in a separate game process which can host the G-factor elements – in the architecture presented in the next chapter this is called a game space (BinSubaih & Maddock, 2007). However that should take into consideration the different types of game objects. The first type are the game objects that have to have representations inside the game engine such as the player and the NPCs who have a 3D object that has to be rendered to provide visual representation. These objects would have to be created in the game engine as well as outside the game engine. The second type of game objects are the ones that do not have to have representations inside the game engine such as the Action, Interaction, and Reaction objects. These objects can be created in the game space only. The creation of the decorative objects can remain the same as they are not considered as being part of the G-factor. As far as the creation of the GUI is concerned it can remain the same for the creation of graphical elements. However, there is a need to pass the interactions that occur to the

game space. For instance when the player modifies the slider this information has to be sent to the game space which is now controlling the behaviour.

The second dependency relates to step 3 which is used for creating the object model. The use of the engine's object model to set the classes creates a dependency between the two. What is needed is a way to set the game's object model independently from the engine's object model. The other problem with step 3 is the use of the static object model which very often leads to the object model extending the engine's proprietary object model and sometimes even using its proprietary language. In step 3 choosing a general language like C++ instead of TorqueScript avoids the language dependency but does not solve the issues with the static object model which complicates modifiability and introduces the programmer bottleneck problem. The third dependency occurs when the game logic is formatted in the game engine's proprietary language format.

2.5.3 Characteristics to Increase the New Game Development Approach's Appeal

Chapter 3 will present a new game development approach which aims to make the G-factor portable. To increase the appeal of such an approach, lessons from this chapter must be learnt. The lessons concern the need for the approach to exhibit three characteristics: (i) to be modular in nature rather than monolithic (i.e. recommends patterns rather than a single complete solution), (ii) to avoid introducing new mechanisms (or new languages or new formalization) and instead to follow something that has been proven and tested, preferably in the game development field, and (iii) to avoid undermining the attractive qualities of the typical development approach which is responsible for the widespread use of game engines in the first place. These characteristics are now described in greater detail.

2.5.3.1 Characteristic 1: Modularize the approach

The approaches described in section 2.2 to aid portability lack modularity. In the AI architecture area the developer is expected to use the architecture produced, which, as was mentioned earlier, means moving from one proprietary format to another. The interfaces area consisted of two types of interfaces: specific and common. Interfaces that propose a single interface per game engine lack the complexity needed to cope with the dynamic nature of game engines, which was an issue with Gamebots (Le Hy et al., 2004). The other interfaces, which propose a unified interface for all engines, are still at their early stages, and it is not clear how they will cope with the issues hampering single interfaces. The third area proposes standards as a way of unifying the representation of the game. This again expects the developers to use the final artefact produced and, as was discussed earlier, it is lacking the complexity and the freedom needed for games. Moreover proposing a new mechanism or language is not the way forward as shall be discussed in the next characteristic. The frameworks or protocols area also suggests a complete artefact to allow interoperability between different engines.

There is a need to reverse the monolithic trend by providing a modularized approach that identifies the decisions that aid portability, describes how they do so, and shows how they trade against each other.

By doing so the developer is relieved from having to commit to a single artefact. Committing to a single artefact is as risky decision as is the decision for choosing a game engine. Modularizing the approach should make it possible for developers to select the decisions that are going to contribute towards portability knowing the overheads involved and the effect they have on the ultimate goal. A modularized approach also means that developers can select to progressively move towards portability rather than make one single leap if they feel it is too risky. For instance they can start with addressing the issue with the object model by implementing a dynamic object model or limit their engine choice to those that support this feature. This should lead to easier migration.

2.5.3.2 Characteristic 2: Avoid introducing new mechanisms or languages

The second characteristic of a successful approach is to avoid radically introducing new mechanisms or languages that are unknown. Game development is very dynamic, and flexibility and freedom are crucial to its success. Therefore any approach would have to accommodate this requirement.

2.5.3.3 Characteristic 3: Avoid undermining the attractive attributes of the typical game development approach

Table 2.4 showed the current motivation factors that attracted developers to use game engines. Any approach should avoid undermining these factors as much as possible. These attributes are a direct consequence of following the data-driven development route which resulted in the following important attributes: use of scripting languages, small learning curves, and use of authoring toolkits. These are not always portability friendly. For example using a proprietary or precompiled scripting language (e.g. UnrealScript) would undermine the portability. Authoring toolkits could reduce the learning curve but they should avoid specifying the game in a proprietary format.

Unfortunately some decisions (e.g. avoid game engines that only provide a proprietary scripting language) might mean excluding some of the popular engines (e.g. 51% of the projects surveyed used Unreal engine). However, in cases where the exclusion is completely unacceptable the approach (or parts of it) is still applicable because of its modular nature. For instance you can add dynamic object model support to Unreal which means that the object model is independent from Unreal's object model and easily portable. This flexibility of the modular approach means that even if the trade-off made is favouring using Unreal engine, you still can reduce the cost of porting the game by employing the other parts of the approach.

3. A New Architecture for Serious Games

3.1 Introduction

This chapter presents an architecture called game space architecture (GSA) which has been implemented to enable G-factor portability. The objective however is not to promote a monolithic artefact but to illustrate the architectural decisions[1] made during the implementation of the architecture and how they simplify the migration of the G-factor between game engines. The chapter also demonstrates how a sample game has been developed using GSA. The success of the architecture in achieving its requirements is evaluated in chapter 4. Chapter 6 will show that GSA is scalable by developing a serious game for traffic accident investigators (SGTAI).

Section 3.2 provides an overview of GSA. Section 3.3 describes the decisions used to develop GSA and how they trade-off against one another. Section 3.4 details the architecture design. Section 3.5 uses GSA to redevelop the Moody NPCs example game demonstrated in section 2.5.1 to show the changes made to the typical game development approach in order to make G-factor portable.

3.2 An Overview of GSA

The objective of GSA is to reduce the dependencies the G-factor elements have on the game engine by adopting a client-server architecture which enables the G-factor to exist independently of the game engine. The client-server approach supports distributed computing over the network and is widely used in game development.

The novel design approach employed in GSA combines a variant of the model-view-controller (MVC) pattern to separate the G-factor (i.e. model) from the game engine (i.e. view), with on-the-fly scripting to enable communication through an adapter (i.e. controller The use of a variant of the MVC pattern, rather than the normal MVC pattern, avoids the known liability where the view is tight-coupled to the model (Buschmann et al., 1996). This variant, with all communication passing through the middleware tier (controller), is similar to both a three-tier architecture, which is linear rather than triangular, and also to MVC++ (Vuorenmaa, 2000). GSA also follows a distributed paradigm where the server holds the model and the controller, and the client becomes the view (see Figure 3.1). This distribution of MVC is similar to a variant called message-based or method-based MVC (Qiu, 2005). Qiu also proposed a variant which uses a publish-subscribe communication mechanism which relies on messaging middleware where a broker orchestrates the services. This communication is similar to what service-oriented architectures uses (e.g. Web Services) to loosen the coupling between the service provider and the service consumer but it adds performance overhead. It would also undermine modifiability which is an aim of GSA.

---

[1] An architectural decision (AD) is made from an overall system perspective.

The use of on-the-fly scripting aims to maintain the ease of modifiability associated with a typical game development approach (see section 2.5.1). Most notably modifiability is upheld in the typical game development approach using scripting, which the two surveys described in chapter 2 found to be very popular with game engines and projects that use game engines. To maintain this level of modifiability (i.e. scripting level access) to the game engine and the game space (a separate game process that hosts the G-factor elements), GSA uses on-the-fly scripting to communicate with both via the adapter, as shown in Figure 3.1. For example, communication may begin with the game engine sending the updates to the adapter (step 1). The adapter converts them into scripts or direct API calls (step 2) which are then used to update the game space (step 3). When the game space needs to communicate with the game engine it notifies the adapter of the changes that need to be communicated (step 4). The adapter formats these into the engine's scripting language (step 5) and sends them to the engine to be executed (step 6).

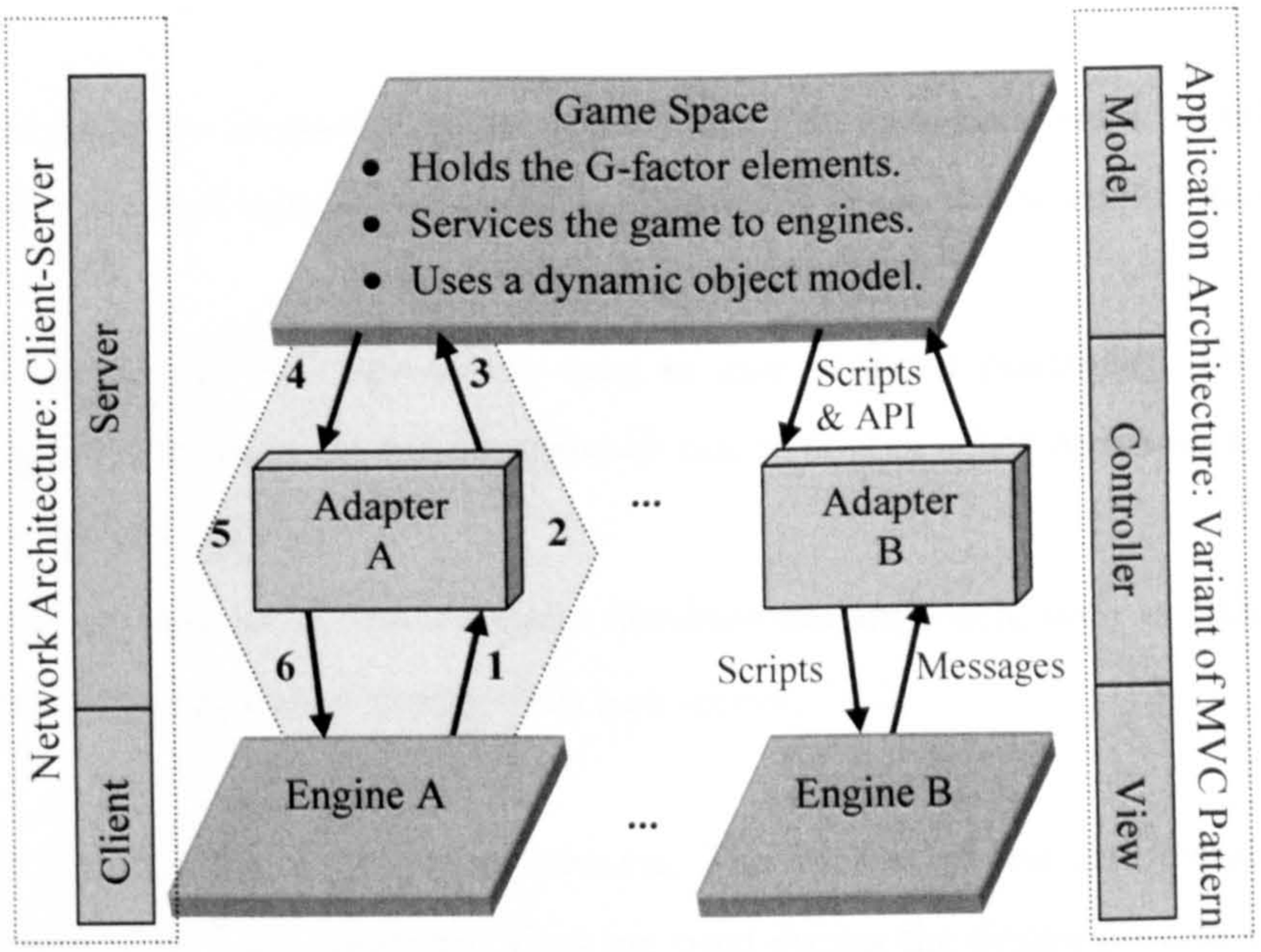


Figure 3.1: An overview of GSA design (steps 1 to 6 are explained in the main text).

The separation and the communication mechanism allow the G-factor to exist independently of the game engine. The effect this has on portability means that when migrating to a new engine the elements in the game space – the game state (which contains the working memory which includes game objects, time, and interactions), the object model (which describes the classes for the objects), and the game logic (which controls the game behaviour)) – can stay intact. Contrasting this to migrating a game developed using a typical game development approach, which often requires all three to be created again, shows the extent of the effort saved. Section 3.4 details the full architecture design. Before that the next section presents the decisions used to develop GSA and how they trade-off against one another.

3.3 Building Blocks: From Concept to Architecture

GSA's requirements can be grouped into two main categories: characteristics and quality attributes. The characteristics represent the general guidelines or principles that need to be considered in an approach that aims to make G-factor portable (see section 2.5.3). The quality attributes represent the specific

requirements of the architecture, which are the ones that are going to be used during the evaluation of the architecture (see chapter 4).

The characteristics were elicited from the issues facing the approaches that have been tried to aid portability (see section 2.2) and from the two surveys presented in chapter 2. These characteristics aim to:

- Produce a modularized approach that can be progressively used to ease G-factor portability. This should be useful when it is not feasible to adopt the whole approach due to cost or other constraints.
- Avoid introducing new mechanisms or languages.
- Avoid undermining the attractive attributes of the typical game development approach, such as rapid prototyping, small learning curve, scripting, and authoring tools (see section 2.4.2).

The quality attributes are used to guide the development process. The success of the architecture becomes dependent on the achievement of these attributes. They are used during the design process to determine the trade-offs that need to be made. For GSA, the three quality attributes selected are (prioritized in order of importance): portability, modifiability, and performance. The portability attribute refers to the ability to run the same G-factor elements (e.g. game logic for first-person shooter game) on different game engines. Modifiability refers to the ability to run different G-factor elements. Performance refers to the ability to run the game without major overheads imposed on the display rate, game's responsiveness, and network throughput.

The forces at work behind the quality attributes are the architectural decisions. The difference between architectural decisions and other decisions that concern the development is that they are made from an overall system perspective. There is no principle of how to spot an architectural decision and it is context dependent (Clements et al., 2001). The architect judges what is architectural depending on how it affects critical requirements (i.e. quality attributes). For instance, in GSA, the decisions on which database to choose or the language of implementation do not represent architectural decisions. However the choice of a scripting language is an architectural decision and is constrained by allowing on-the-fly processing since it affects the three quality attribute (see Table 3.1). The actual effect of all eight architectural decisions will be evaluated in the next chapter using the architecture trade-off analysis method (ATAM). The remainder of this section presents the architectural decisions and describes how they satisfy GSA's requirements.

3.3.1 AD1: Model-View-Controller (MVC)

The first architectural decision made was to adopt the MVC pattern (see Figure 3.2) which separates the system core from the view. In the MVC pattern the model contains the core functionality and system data. In GSA these are the G-factor elements and are placed in the game space. The view is used to display the information to the user, and in GSA the game engine becomes the view. Lastly the controller

handles the change requests by linking the view to the model. In GSA the adapter becomes the controller.

The main benefit of using this approach is that multiple views can exist for the same model and any modification carried out by one of the views is visible to other views. The separation of the model from

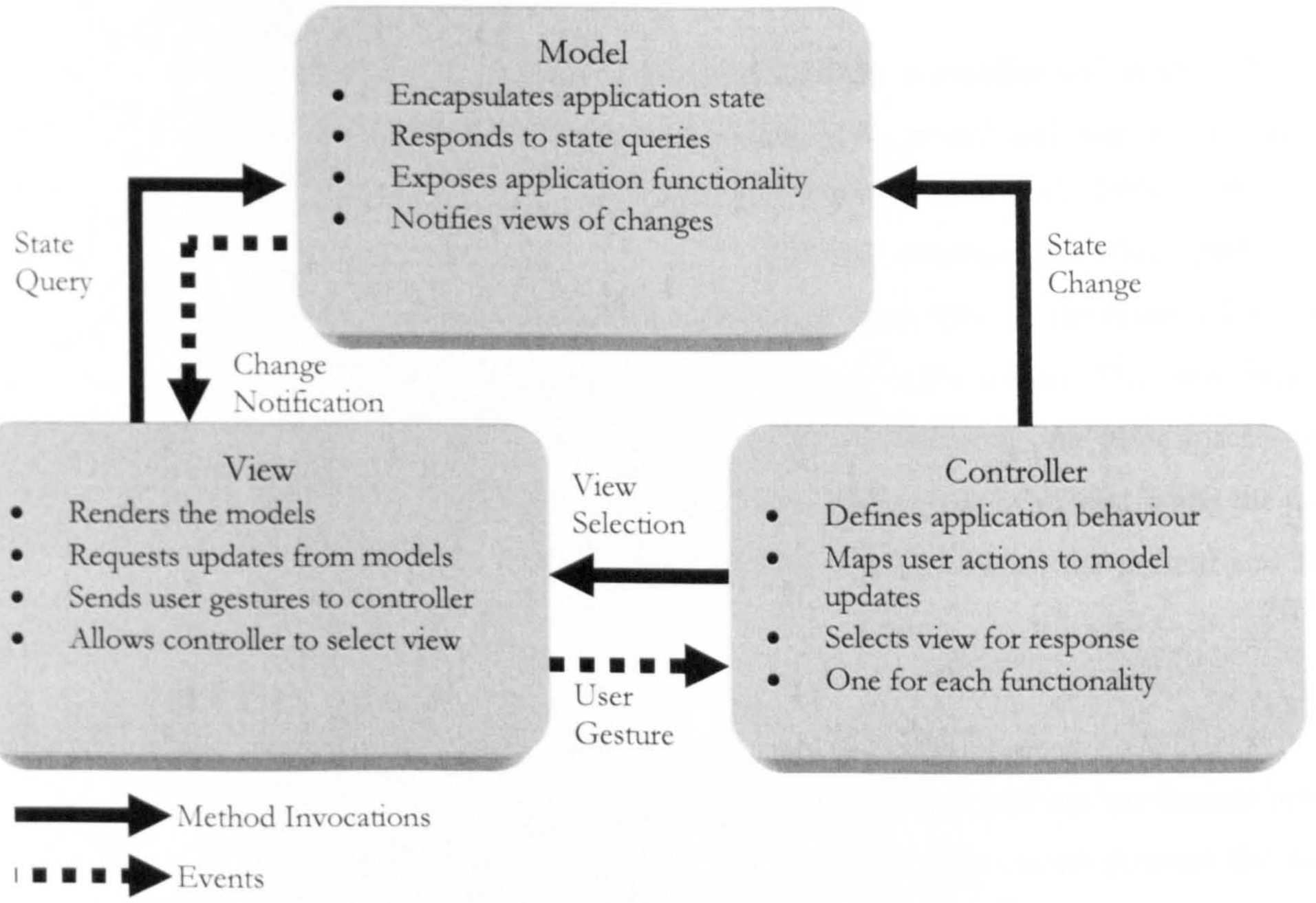


Figure 3.2: The Model-View-Controller pattern (after (Singh et al., 2002)).

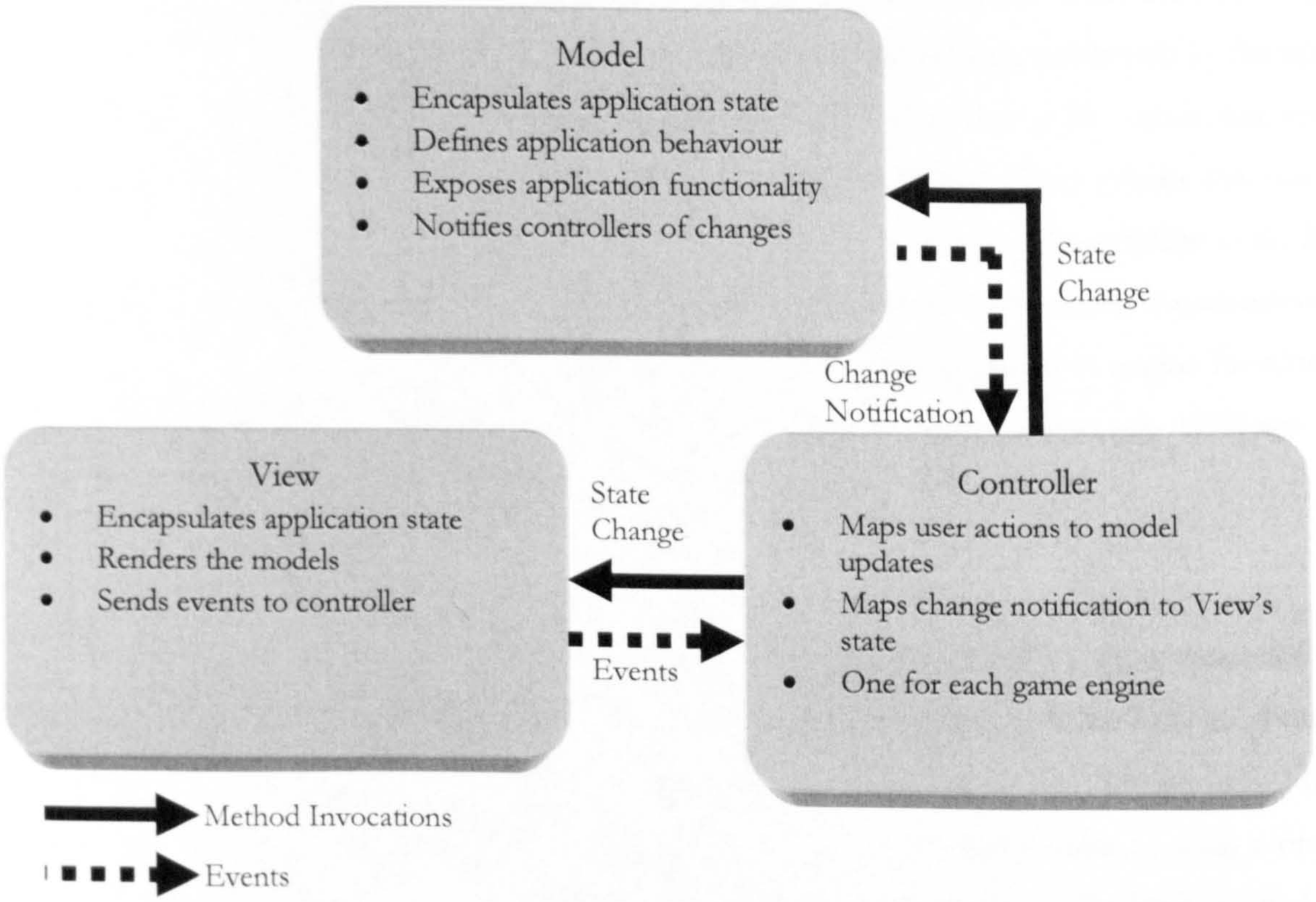


Figure 3.3: A variant of the Model-View-Controller pattern.

the views is exactly what the portability attribute pursues (the G-factor being the model and the game engines being the multiple views). However, GSA breaks the direct link between the view and the model and only allows communication through the controller, as shown in Figure 3.3. Therefore the approach can be considered as a variant of the MVC pattern. Removing the link between the view and the model means that the MVC pattern also promotes better modifiability since it allows the game object model and logic to be modified independently from the view.

The separation promoted by using the MVC pattern to promote portability and modifiability adds processing overhead to the architecture as information needs to be passed and sometimes translated between the three parts. The controller employs the adapter pattern (Gamma et al., 1995) to accomplish the translation by using a scripts mapping table (see AD7). The other overhead that affects performance is the network overhead because of the distributed environment used. One of the reasons for using a distributed environment is because game engines are known to be CPU-intensive. The other benefit is the interoperability it facilitates as different game engines can be linked through the game space – this is an interoperability quality attribute for the future but this shows the benefits of at least listing the quality attributes as it forces the architecture to weight the impact of any decision on the present and future attributes.

3.3.2 AD2: Asynchronous Messaging

The second architectural decision made was to use messaging as a communication mechanism between the game space and the game engines. More specifically the communication occurs between the adapter (which is in the game space) and the game engine. The messaging role is to allow synchronization of the two game states in the game engine and the game space. Since the two game states are not using the same object model or language, synchronization requires a translation which is performed by the adapter in the MVC pattern. To compensate for the overhead introduced by messaging, the architecture uses an asynchronous type to reduce the impact on the display rate, since the calling system can continue processing after making the request and does not have to wait for a response (Trowbridge et al., 2004). Asynchronous messaging supports the performance and the portability attributes but it undermines the modifiability attribute since the developer does not have direct access to the game engine functionality. To reduce the impact of messaging on modifiability an on-the-fly scripting decision (see AD3) was taken to allow the developer access to the engine's scripting mechanism.

3.3.3 AD3: On-the-fly Scripting

The third decision was to use scripting to manipulate the game on both sides (i.e. game space and game engine). On-the-fly scripting is used instead of pre-compiled scripting since using the latter would require some of the game logic to be put into the game engine's specific format, as was the case with Mimesis which used pre-compiled UnrealScript classes (Young et al., 2004). The applicability of using scripting is dependent on the level of game engine functionality exposed through scripting. The level required varies

from one game to another and the minimum requirement is for a level of access which caters for the interaction required by that game. Furthermore, as was shown by the two surveys in chapter 2, scripting is a very popular mechanism for modifying the game. However on-the-fly scripting has a negative impact on the performance as it runs much slower than precompiled scripting. Despite this it is used in many commercial games – survey 1 (game engines) in chapter 2 (see section 2.3.4) showed 48% of game engines provide this feature.

3.3.4 AD4: Dynamic Object Model

The aim of using a dynamic object model is to preserve the flexibility and extensibility of the object model which were found to be very useful in game development (see section 2.3.2). The dynamic object model eases the modifiability of the G-factor and remedies the issues associated with the static object model (see section 2.3.2). The other added value of using a dynamic object model is its portability since it is independent of the implementation language. The model can be stored externally, and commonly in a data format which makes it more accessible compared to its static counterpart, which is language dependent and stored in the application code. Although the dynamic object model supports portability and modifiability, it undermines performance. However since it has been tried and tested in game development it is adopted here. The other benefit of a dynamic object model is with regards to the future requirement of allowing domain experts to share domain knowledge easily by manipulating this model without the need for a developer, just like ontologies[2]. This was one of the reasons for developing a graphical tool for creating the object model (BinSubaih et al., 2005c).

3.3.5 AD5: Application Programming Interface

Providing application programming interface (API) access serves the projects that might find scripting too slow for setting the G-factor or for embedding external tools in the game space.

3.3.6 AD6: Interoperate with External Tools

Interoperability of the game engine with external tools (e.g. rules engine) was found to be valuable by 30% of the projects in survey 2 (projects using game engines) in chapter 2 (see section 2.4.1). GSA allows interoperability in two ways: through an adapter or by embedding the tool in the game space. A similar mechanism to the one used for interacting with game engines (i.e. through adapters) can be used to interoperate with external tools. The second way is to embed the external tool in the game space application and interface with it using an API or using scripting. Having an external tool linked to the game space instead of the game engine means that it can be easily ported to another game engine which is linked to the game space. Moreover, since there is no direct link between the game engine and the external tool the ripple effect is confined to the game space. As far as the performance is concerned the

---

[2] Ontologies are "content theories about the sorts of objects, properties of objects, and relations between objects"(Chandrasekaran et al., 1999).

first option would be affected by the additional translation required through the adapter and by the network overhead. The second option would remove the network overhead. An example of embedding an external tool is a rules engine called JESS which was embedded in an earlier version of GSA (BinSubaih et al., 2005b). The role of this rules engine was to dictate the game behaviour.

3.3.7 AD7: Scripts Mapping Table

Using a scripts mapping table is an architectural decision taken to translate between the different scripting languages. The table holds the same sentence in two languages: the one understood by the game space (Jython) and the one understood by the game engine (TorqueScript or Python) or the external tool (JessScript). Each sentence holds placeholders for the information to be replaced at run-time. Although this decision supports portability, it undermines modifiability since a record has to be added for the information that has to be exchanged. The impact on performance is dependent on the table size.

3.3.8 AD8: Objects Mapping Table

The final architectural decision made is to use an objects mapping table. This is necessary to link the corresponding objects on both sides (i.e. game space and game engine) if similar unique identifiers cannot be set on both sides for the same objects. Similar to the scripts table this aids portability but undermines modifiability as a record has to be added for each object to be mapped. Also, similar to the scripts table, performance is dependent on the table size.

3.3.9 Impact of the Architectural Decisions

Table 3.1 summarizes the impact the architectural decisions have on the three quality attributes and how they trade-off against each other according to the architect. The effects of these decisions are further scrutinized in chapter 4 using the architecture trade-off analysis method (ATAM).

Table 3.1: The impact of the architectural decisions on the quality attributes (↑ = supports, ↓ = undermines, space = neutral).

Architecture Decision (AD)	Portability	Modifiability	Performance
AD 1: MVC	↑	↑	↓
AD 2: Asynchronous messaging	↑	↓	↑
AD 3: On-the-fly scripting	↑	↑	↓
AD 4: Dynamic object model	↑	↑	↓
AD 5: API		↑	↑
AD 6: Interoperate with external tools	↑	↑	
AD 7: Scripting mapping table	↑	↓	
AD 8: Object mapping table	↑	↓	

3.4 Architecture Design

This section describes the architecture design based on the decisions listed in the previous section and it uses the example game (described in section 2.5.1) to illustrate aspects of the design. Figure 3.4 shows

the architecture design. This follows a client-server approach with the game space residing in the server machine and the clients running on separate nodes and communicating over the network. Following the MVC pattern makes the game engines the views and the adapters the controllers. The rest of the game space acts as the model and comprises of a dynamic object model package, game state package, scripting engine, and networking engine. The server side also holds two artefacts: the behaviour scripts and a persistent database to hold the dynamic object model and the game state. The game logic can also be set by embedding a behaviour engine inside the server just like the networking engine, or it can be connected to it with an adapter just like a game engine.

Figure 3.5 shows the activity diagram demonstrating the communication process between the different parts of the architecture. When an event, such as a player action, occurs in the game engine, the event is checked to see if it affects the G-factor and if it does the process of communicating it with the game space starts. If the event is of no interest to the G-factor no communication is necessary. For instance, in the Moody NPCs game described in section 2.5.1, the G-factor is not interested in dealing with any collision between the player and other objects in the game. Therefore this type of action is not communicated.

The send activity is responsible for communicating the event to the game space. After formatting the event in a predefined format it is sent to the adapter in the game space. Each engine is assigned its own adapter which understands its communication protocol. Upon receiving the message the adapter queries the scripts mapping table and translates the message into a script formatted in the game space's scripting format (i.e. Jython). The adapter also has the role of mapping between the two object models if they are different. The translation converts the predefined text message into a script which is then executed by the scripting engine to update the game state. The game state can also be updated by the behaviour script or by an external tool such as a rules engine (e.g. JESS (BinSubaih et al., 2005b)).

Each update activity carries with it an exemption condition. The exemption serves two roles. First it is necessary to stop the round trip that might result from the adapter being notified of its own updates which causes unnecessary processing and networking overheads. The second role is for exempting other adapters from receiving updates from an adapter that they might not be interested in. For example if the game space is linked to two external tools, e.g. rules engine and dialogue engine, and if the dialogue engine is not interested in receiving notifications for the changes that occur in the rules engine, the exemption flag can then be used to avoid the extra communication overheads.

The update activity also notifies the monitoring activity whenever an update occurs passing it the object type that has been updated (it can also be extended to pass the property that has been changed). The monitoring activity examines its list of the object types that need to be communicated with the game engine. If the object type is found, the monitoring activity notifies the adapter which queries the script mapping table and translates the notification into a script formatted into the game engine scripting language (e.g. TorqueScript for the Moody NPCs example). The script is then sent to and executed by the scripting engine in the game engine. This asynchronous form of communication does not require the game engine to wait thus allowing the game loop to run continuously to achieve the best frames per second (fps).

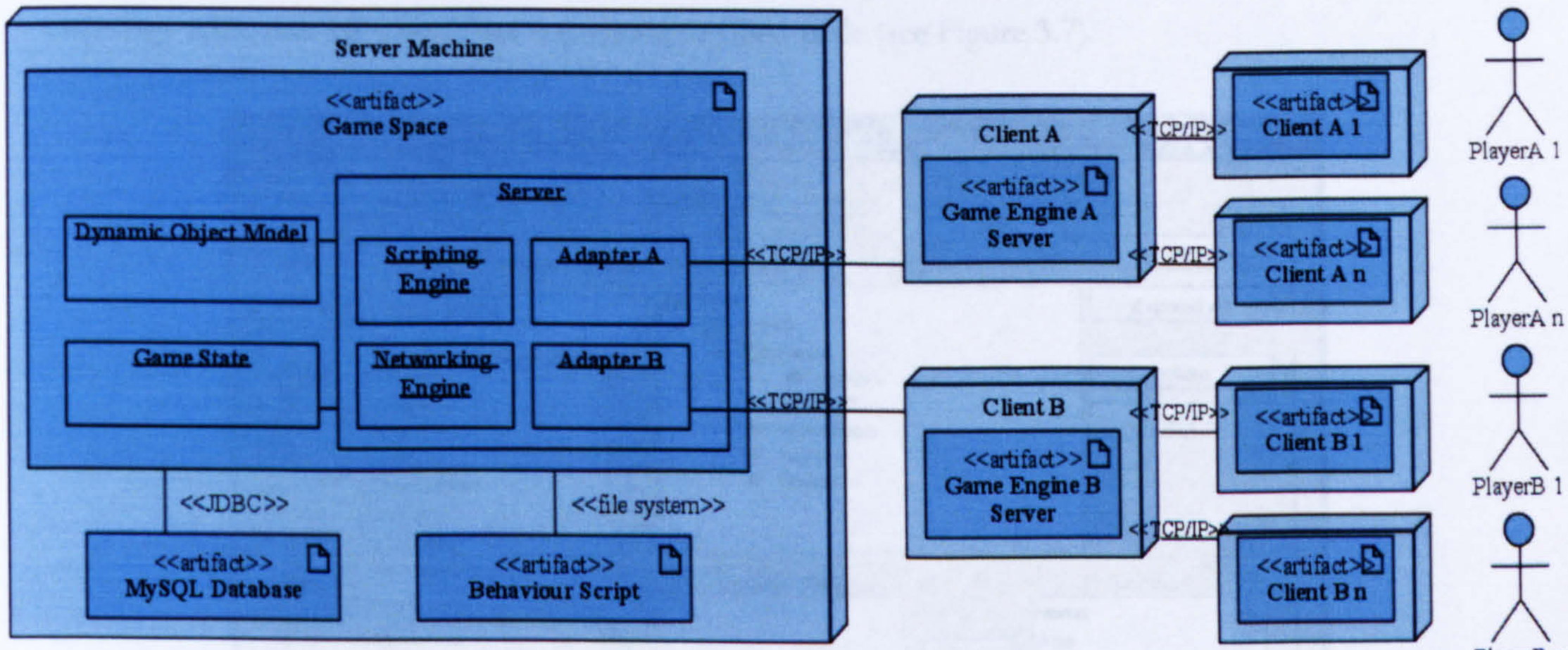


Figure 3.4: Game space architecture.

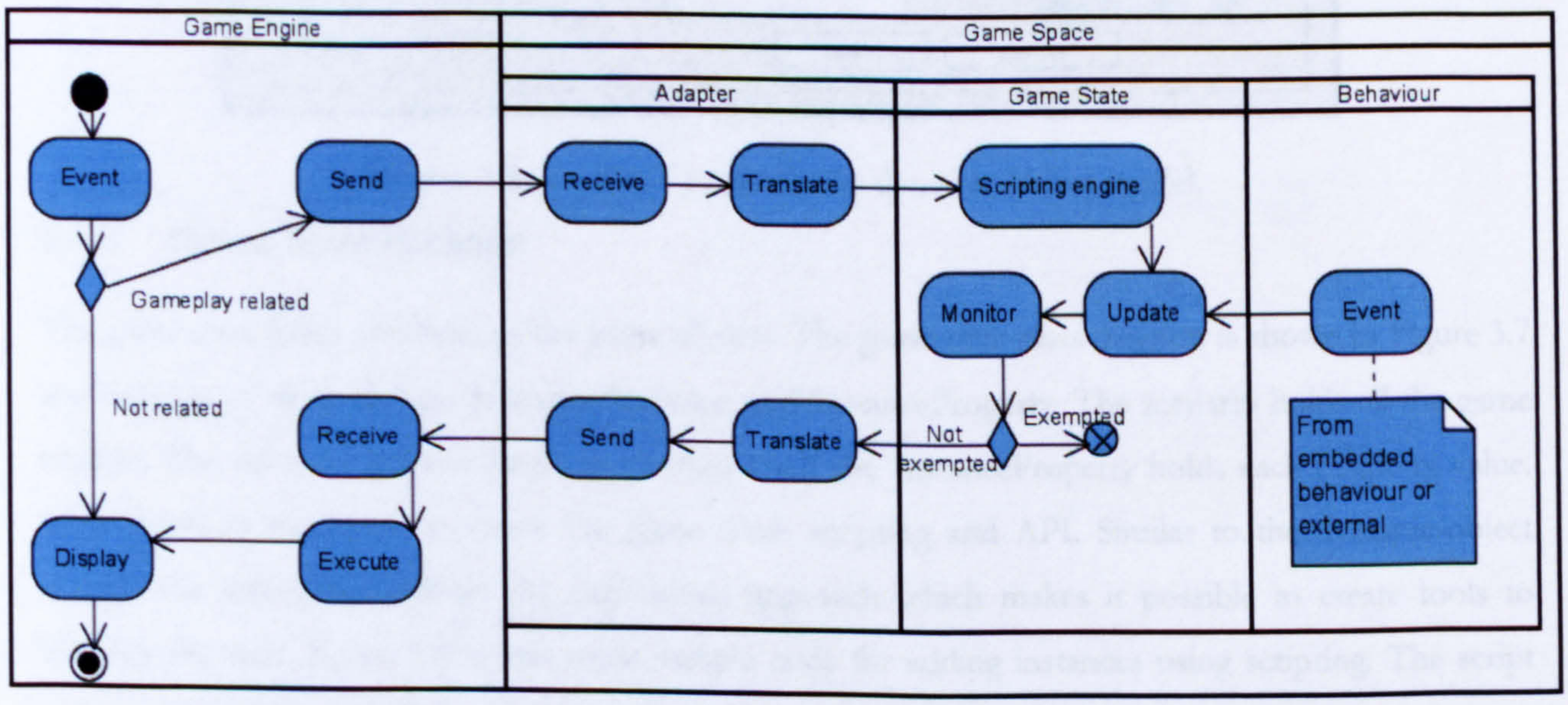


Figure 3.5: Activity diagram

3.4.1 Dynamic Object Model Package

The dynamic object model replaces the need to create a static object model. The only static object model used is the base classes for holding the dynamic object model. The class diagram for the dynamic object model consists of the following classes: Ontology, Class, Property, and Type.

Since each game (i.e. G-factor) can have a different object model, the Ontology class holds all the classes for a particular game. The Class holds all its properties and the Property holds what type of values it can store. Figure 3.6 shows the tool developed to simplify the creation of the dynamic object model (BinSubaih et al., 2005c). The object model created is then stored in a database by mapping the classes to the database tables as shown in Figure 3.7. For instance the Ontology class is mapped to the Ontology table and the Class class is mapped to Class table (see Figure 3.7).

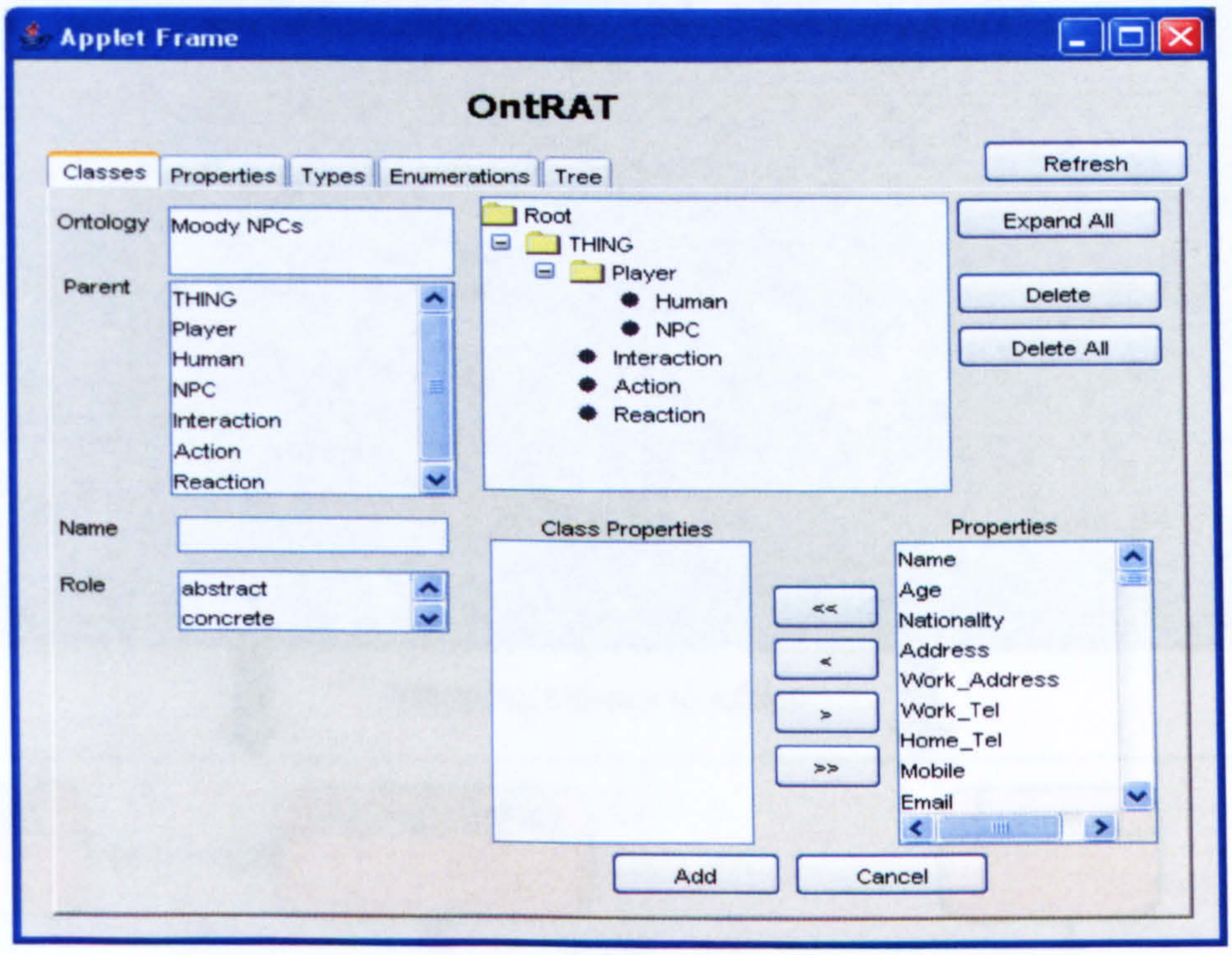


Figure 3.6: OntRAT to create the dynamic object model.

3.4.2 Game State Package

The game state holds instances of the game objects. The game state class diagram is shown in Figure 3.7 and consists of three classes: Scenario, Instance, and InstanceProperty. The scenario holds all the game objects. The Instance holds a single game object and the InstanceProperty holds each property value. GSA provides two ways to create the game state: scripting and API. Similar to the dynamic-object model, this separation follows the data-driven approach which makes it possible to create tools to simplify the task. Figure 3.8 shows some sample code for adding instances using scripting. The script creates two NPCs. It achieves that by calling the 'addInstance' method which takes a class name (e.g. NPC) as the first argument followed by the properties that need to be set (e.g. Name) followed by the values (e.g. Kork1).

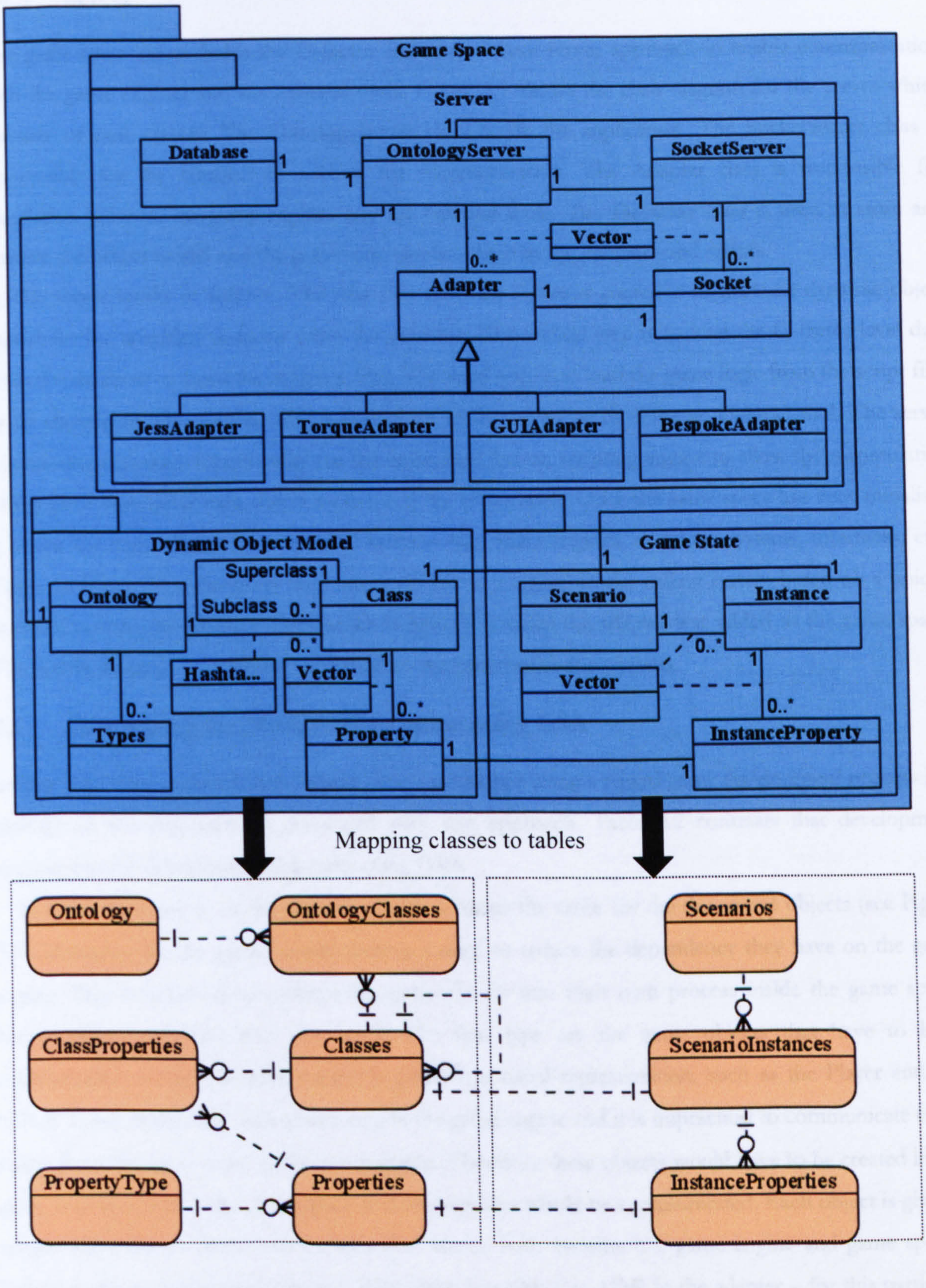


Figure 3.7: Game space class and table diagrams.

```
Scenario.addInstance("NPC", "ID,Name,Forgiveness,Punishment,Voice", "1,Kork1,0,0,1");
Scenario.addInstance("NPC", "ID,Name,Forgiveness,Punishment,Voice", "2,Kork2,0,0,2");
```

Figure 3.8: Adding instances using scripting.

3.4.3 Server

The game space server holds the G-factor and uses a client-server approach to enable communication with the game engines and the external tools. Figure 3.7 shows the class diagram for the server which consists of nine classes. The OntologyServer class holds the application. The SocketServer class is responsible for the creation of sockets for communication. The Adapter class is responsible for translation between the game engines and the external tools. The Database class is used to store and retrieve the object model and the game state can be stored in the database and scripts.

The server works as follows. The first step after the server is started is to load the dynamic object model for the specified G-factor from the database. The second step is to load the G-factor level data from the database or from the scripting files. The third step is to load the game logic from the script files or by starting another module that is embedded in the server or by using an external tool. The server exposes the necessary functionality for the embedded Jython scripting engine to allow the manipulation of the game state, level data, object model, and the server itself. Once the game space has been initialised it listens for connections from external systems (e.g. game engines, decisions systems, interfaces, etc). Upon receiving the connection the correct adapter is assigned to the calling system based on a unique keyword sent by the calling system which is agreed on when the adapter was added to the game space. The activity diagram (see Figure 3.5) describes the communication process.

3.5 Developing the Moody NPCs game using GSA

Section 2.5.1 demonstrated how a game can be developed using a typical game development process and highlighted the dependencies associated with that approach. Table 3.2 contrasts that development approach to the development approach using GSA.

In GSA the creation of the game level data remains the same for the decorative objects (see Figure 3.9). However, for the game objects there is a need to reduce the dependency they have on the game engine. This is achieved by moving the game objects into their own process inside the game space. Game objects are split into two types. The first type are the game objects that have to have representation inside the game engine to provide a visual representation, such as the Player and the NPCs. These require real-time processing in the game engine and it is impractical to communicate every frame from the game space to the game engine. Therefore these objects would have to be created in the game engine as well as the game space and only updates would be communicated. Each object is given a unique identifier so that it can be identified across both systems (i.e. game engine and game space). Failure to do so requires creating the object mapping table (i.e. AD8) in the adapter – for this particular game this was not necessary (and it was not necessary for the development of SGTAI in chapter 6).

Table 3.2: Comparing a typical game development approach to the GSA approach.

Step	Typical Approach	GSA's Approach
1. Create the level data.	• Create the decorative objects in the game engine.	
	• Create the game objects using the world builder and load them using TorqueScript.	• Create the game objects using the world builder in the game engine and give them a unique ID which identifies these objects in the game space as well. Load these object using TorqueScript. • Create the game objects in the game space with the same unique ID using Jython.
2. Create the GUI.	• Use the game engine interface builder or TorqueScript to create the interface. The behaviour is set as part of the game logic (step 4).	
3. Create the object model.	• Use TorqueScript to extend the objects or create new ones.	• Create the object models for the game objects models that require representation in the game engine and the game space. • Create the other game objects models in game space.
4. Create the game logic.	• Use TorqueScript to set the behaviour in the game engine.	• Use Jython or Java to create the logic in the game space.
5. Create the adapter.		• Send the updates from the game engine to the game space. • Create the adapter which translates between the game engine and the game space.

The second type of game objects are the ones that do not have a representation inside the game engine, such as the Action, Interaction, and Reaction objects. These objects can be created in the game space only. The process of creating the game objects in the game space can be achieved by using Jython scripting.

Interface creation remains the same. The behaviour of the game interface is considered part of the game logic and is specified in step 4 (in Table 3.2). If it is important to make the interface portable then it can be created externally and linked via the game space. (Section C.5 describes how this approach was used in the development of SGTAI to deal with lack of support for Unicode in an earlier version of Torque.) In the typical approach the object models rely on the game engine's object models. In GSA the object models are split into two types. The first type (i.e. Player and NPC) is created in the game engine as well as the game space, as shown in Figure 3.10. The second type (i.e. Action, Reaction, and Interaction) is only created in the game space.

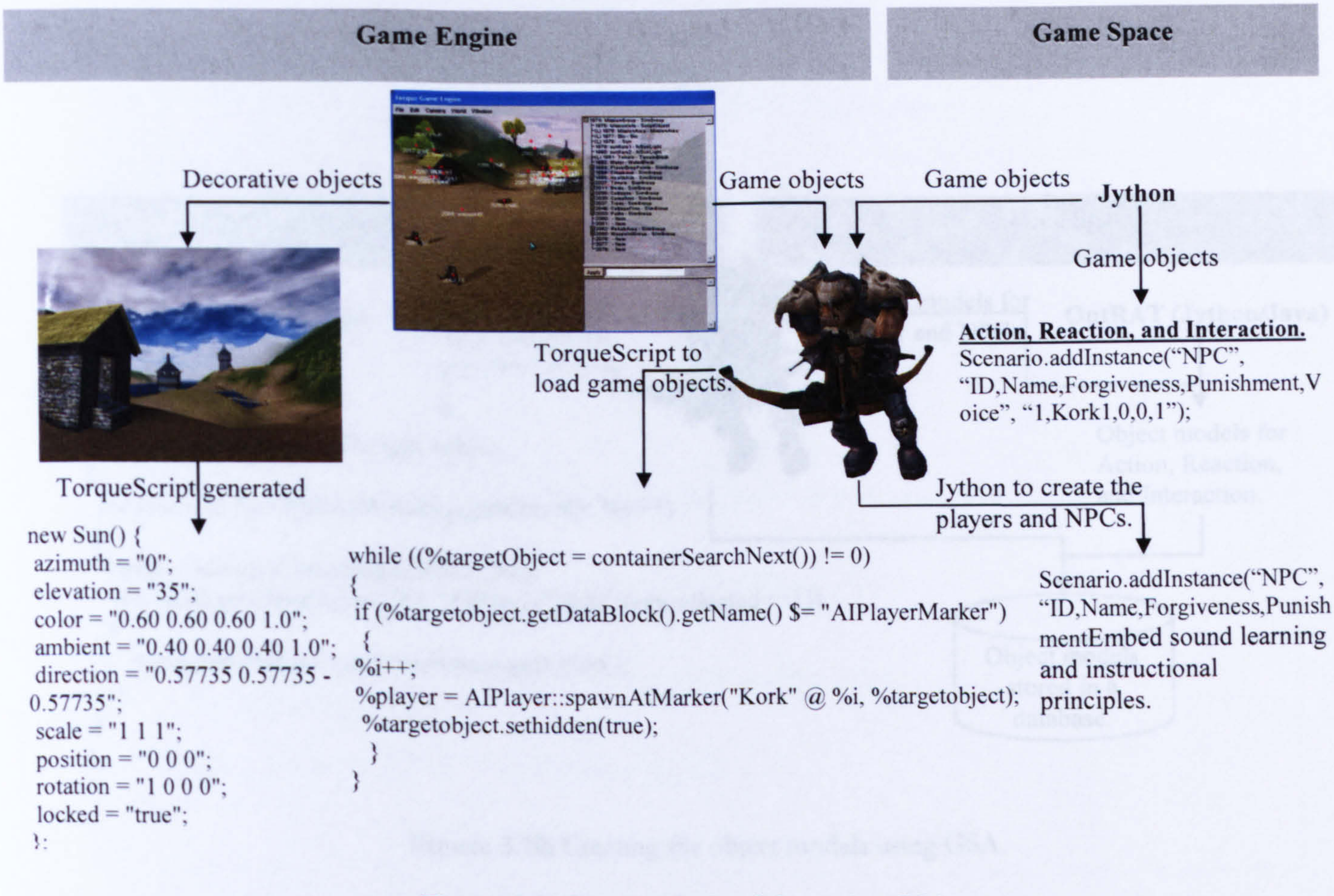


Figure 3.9: Creating the level data using GSA.

The creation of the game logic is moved to the game space which provides Jython and Java to set the logic, as shown in Figure 3.11. This ensures that the game logic does not have to be altered at all when migrating to another engine since it runs independently from the game engine.

The fifth step is creating the adapter which allows for communication between the two systems. Two on-the-fly scripting languages (Jython and TorqueScript) were used on both sides of the adapter to avoid introducing restrictions. This ensures that the same level of access for manipulating the G-factor by on-the-fly scripting is also available to the adapter. To achieve the translation between the two systems the adapter holds a scripts mapping table. It also holds an objects mapping table for game objects that could not be given the same unique identifier across the two systems. A sample of the adapter code is shown in Figure 3.12. The first part of the sample code shows that when an update is received from the game engine for a game object the corresponding object on the game space is retrieved and updated. In this game this is achieved by using Java, however it can equally be done using Jython. The second part of the sample code shows how updates are sent the other way (i.e. from the game space to the game engine). The code shows that when a player selects an object of type NPC the game space makes the NPC turn and face the player by sending a script (objectName.setAimObject(user);) written in TorqueScript. A similar process is followed for the other interactions. Figure 3.13 shows when a player greets an NPC who is in a good mood he is greeted back by a wave and a text message that appears on top of the NPC player.

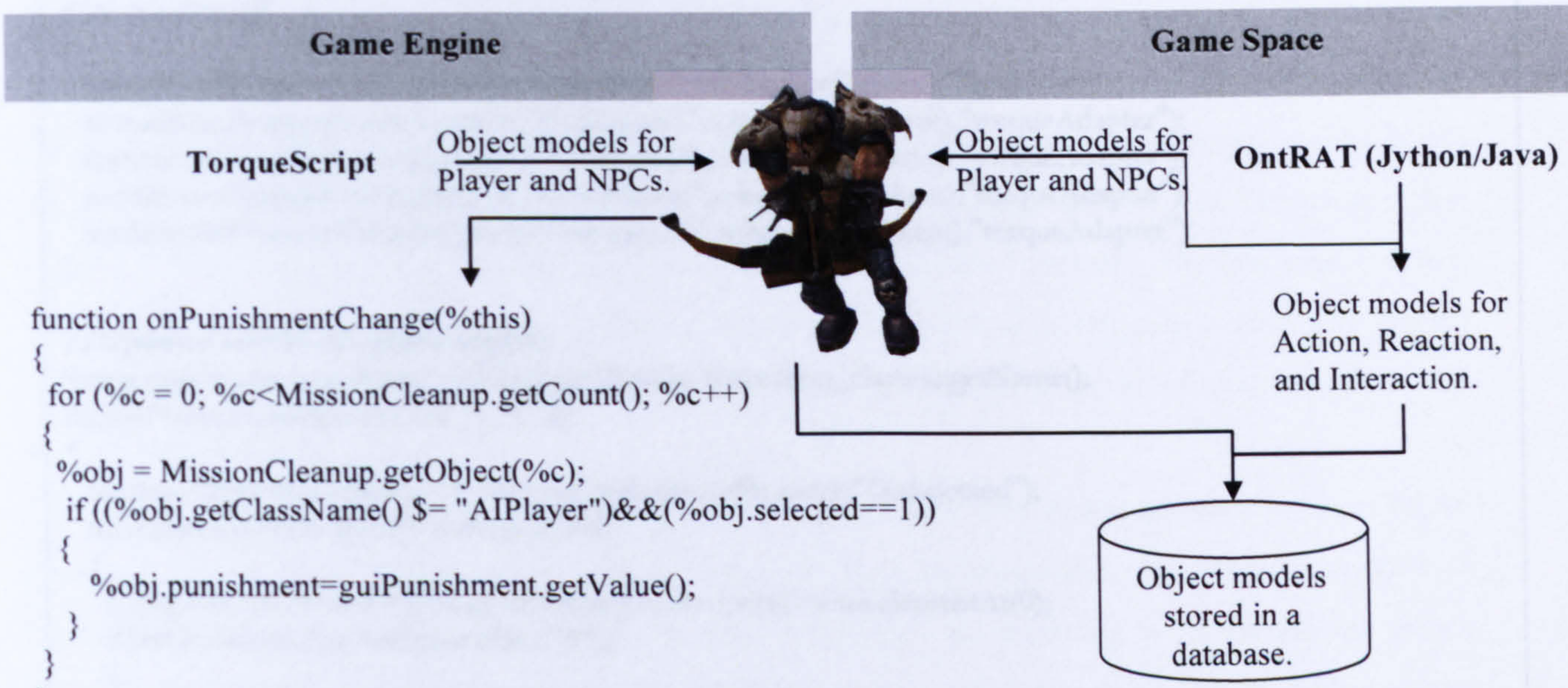


Figure 3.10: Creating the object models using GSA.

```
public String calculateEmotionOutput(String actionType, double forgiveness, double punishment, double
cowardness,double courageness)
 {
  /*
   * Action: verybad=-1; bad=-0.50; normal=0;good=0.50,pleasant=1
   */
   double actionWeight=0;
   if(actionType.equalsIgnoreCase("verybad"))  {
    actionWeight=-1;
   }
   else if(actionType.equalsIgnoreCase("bad"))  {
    actionWeight=-0.5;
   }
   else if(actionType.equalsIgnoreCase("normal")){
    actionWeight=0;
   }
   else if(actionType.equalsIgnoreCase("good")) {
    actionWeight=0.5;
   }
  else if(actionType.equalsIgnoreCase("pleasant")) {
   actionWeight=1;
  }
  ...
```

Figure 3.11: Game logic created in the game space using Java.

```
//Updates received from the game engine are used to update the game space
Instance instance=(Instance)htInstances.get(torqueName);
Scenario scenario = ontServer.getScenario();
if(instance!=null)
{
  scenario.setPropertyValue(instance,"onSelected",tokens.nextToken(),"");
  scenario.setPropertyValue(instance,"Forgiveness",tokens.nextToken(),"torqueAdapter");
  scenario.setPropertyValue(instance,"Punishment",tokens.nextToken(),"torqueAdapter");
  scenario.setPropertyValue(instance,"Cowardness",tokens.nextToken(),"torqueAdapter");
  scenario.setPropertyValue(instance,"Courageness",tokens.nextToken(),"torqueAdapter");
}

//Updates sent to the game engine
String className = ontServer.getOnt().getClass(instance.idon_classes).getName();
if(className.equalsIgnoreCase("NPC"))
{
  InstanceProperty onSelected = instance.getInstanceProperty("OnSelected");
  if(onSelected.vecPropertyValues.size()>0)
  {
   String intOntSelected =(String)onSelected.vecPropertyValues.elementAt(0);
   if(intOntSelected.equalsIgnoreCase("1"))
   {
    sendToTorque(instance.name+".setAimObject(user);");
   }
  }
}
```

Figure 3.12: A sample of the adapter code.

Figure 3.13: A player being greeted back by an NPC. (The text is shown enlarged for the purposes of illustration only.)

3.6 Summary

This chapter has presented a new development approach which tackles the dependency that undermines the G-factor portability. When contrasting the GSA approach with a typical game development approach it is obvious that the GSA approach requires extra work. Therefore, the choice between the two can be dependent on the project size. If the project is a prototype, and if rewriting the game is not an issue, then the typical development approach is more suitable for rapid prototyping. However, if that is not the case, then GSA development approach presents a number of benefits. The first benefit is that developers can keep the visual aspects of their game up to date with the latest game engine. The second benefit is not having to face 'the RenderWare Problem', i.e. the discontinuation of an engine that time and effort were invested in.

Dounis (Dounis, 2006) predicts that gameplay is going to be the distinguishing factor between future games. This will generally mean an increase in the game size. Combined with the increased number of commercial licensees of game engines and the interest engines are receiving from outside the games industry (e.g. serious games community), this will increase the need for portable games. Although the sample game presented in this chapter is too small to be considered a representative test of the ability of GSA to cope with the growth in the game size, it nevertheless is indicative of GSA's capability in addressing the over-dependency issue. Further tests are required to examine how gameplay growth will affect the three quality attributes (i.e. portability, modifiability, and performance). Chapter 4 and 6 will provide further evaluation of GSA. Chapter 4 will reveal how the architectural decisions have affected these quality attributes and chapter 6 will use GSA to develop a serious game which has been used to train traffic accident investigators in the Dubai police force.

4. Evaluating the Game Space Architecture

4.1 Introduction

The software architecture represents the foundation of any system by describing the components that make up the system, the externally visible properties of these components, and the relationship between them (Bass et al., 1998). The main objective of evaluating a software architecture is to find errors at an early stage in the design process (Clements et al., 2001). These errors are easier and cheaper to fix than ones that have already been implemented. The evaluation can also be carried out at different stages of the software development lifecycle even after the software has been implemented. Late evaluation is useful to gain a better understanding of the architecture as a whole by revealing the critical decisions made and the risks associated with these decisions. This information is invaluable when evolving the architecture since knowing this helps guide the development process to ensure that the strong points of the architecture are guarded and the risks are reduced or fixed.

A common way to evaluate a software architecture, which is still used today, is to follow an unstructured or ad hoc approach (Bahsoon & Emmerich, 2006). In the unstructured evaluation there is no obvious pattern to follow other than randomly throwing challenges at the architecture and hoping that either the architecture can address them, or otherwise that they will reveal the architecture's limitations. The unstructured approach has been criticised for its failure to ensure that the whole architecture has been exercised, which can lead to serious problems remaining undiscovered (Parnas & Weiss, 1987; Clements et al., 2001). The field of software architecture evaluation emerged to address this need. It uses systematic methods to assess and validate the architecture's success in meeting its requirements (Bahsoon & Emmerich, 2006). In structured evaluation the probing of the architecture uses methods such as ATAM (Clements et al., 2001), SAAM (Kazman et al., 1998), ARID (Clements, 2000), and ABAS, PASA and CBAM (Bahsoon & Emmerich, 2003). These methods help guide the probing process and thus have a better chance at exercising the whole architecture.

This chapter presents the findings of evaluating the game space architecture (GSA) presented in chapter 3 using both unstructured (BinSubaih et al., 2005b) and structured (BinSubaih & Maddock, 2006) approaches. Section 4.2 describes how a number of challenges were thrown at GSA in an unstructured evaluation process. This section also describes the shortcomings of unstructured evaluation. Section 4.3 describes how a structured evaluation compensated for the shortcomings. Section 4.4 presents the lessons learned from carrying out the evaluation.

4.2 Unstructured Evaluation

The unsystematic approach of throwing challenges at an architecture is appealing as it does not require the extra effort often associated with structured evaluation of having to consider the whole architecture

and having to interact with stakeholders[1]. This is time consuming and often seen as unnecessary when you just want to see how well the architecture can address a certain attribute, such as to see the frames-per-second (fps) achieved when using GSA compared to using the typical game development approach. Examples of architectures that have been evaluated using this way are TIELT (Aha & Molineaux, 2004), MIMESIS (Young et al., 2004), and Gamebots (Adobbati et al., 2001).

For GSA, unstructured evaluation is used to reveal how the architecture handles four challenges:

- Portability challenge (see section 4.2.1): the success of this is judged by the ability to service the same G-factor to two different game engines without modifying it to suit that engine. The only modification allowed is to the adapter, which includes the game engine's code used to create the predetermined messages that are used to notify the adapter of the game engine's state changes.
- Modifiability challenge (see section 4.2.2): the success of this is judged by the ability to create different G-factors on the same architecture using, for example, different object model and game logic.
- Performance challenge (see section 4.2.3): this aims to find the average reduction in fps due to the use of GSA.
- Scalability challenge (see section 4.2.4): this aims to identify how much overhead is added as the game size grows.

The remainder of this section describes each challenge in greater detail by describing the scenario used, what the test revealed, and the implementation overhead endured. The implementation overhead for each challenge will be used in section 4.2.5 to forecast the growth of the adapter compared to the game logic size.

4.2.1 Portability Challenge

The first challenge queries the ability to run the same G-factor on two different game engines. The game used for the challenge consists of what is known as a smart terrain. The environment consists of NPCs, objects, and zones. The player takes the role of a traffic accident investigator and can navigate the scene and interact with the NPCs or objects to get information and receive clues of what he should do next. The zones are proximity sensors which are used to warn the player when he enters or leaves areas of particular importance. The same G-factor was run on two different engines (BinSubaih et al., 2005b): a bespoke engine developed on top of DirectX 9.0 and the Torque game engine. Figure 4.1 illustrates both. This was done without modifying the G-factor and was constrained to modifying the adapter. The two adapters used are shown in Appendix B. A sample of the object model used for the G-factor is shown in Figure 4.2 and a sample of the game logic is shown in Figure 4.3. The game logic size is small (60 lines) compared to the implementation overheads of the bespoke and Torque adapters which are 346 and 354 lines of code, respectively. The test is indicative of the ability of GSA to make the G-factor

[1] Stakeholders are people with interest in the system that will be built on the architecture (Clements et al., 2001).

portable. The size increase however will affect GSA's performance and the adapter's implementation overhead which will be examined in sections 4.2.4 and 4.2.5 respectively.

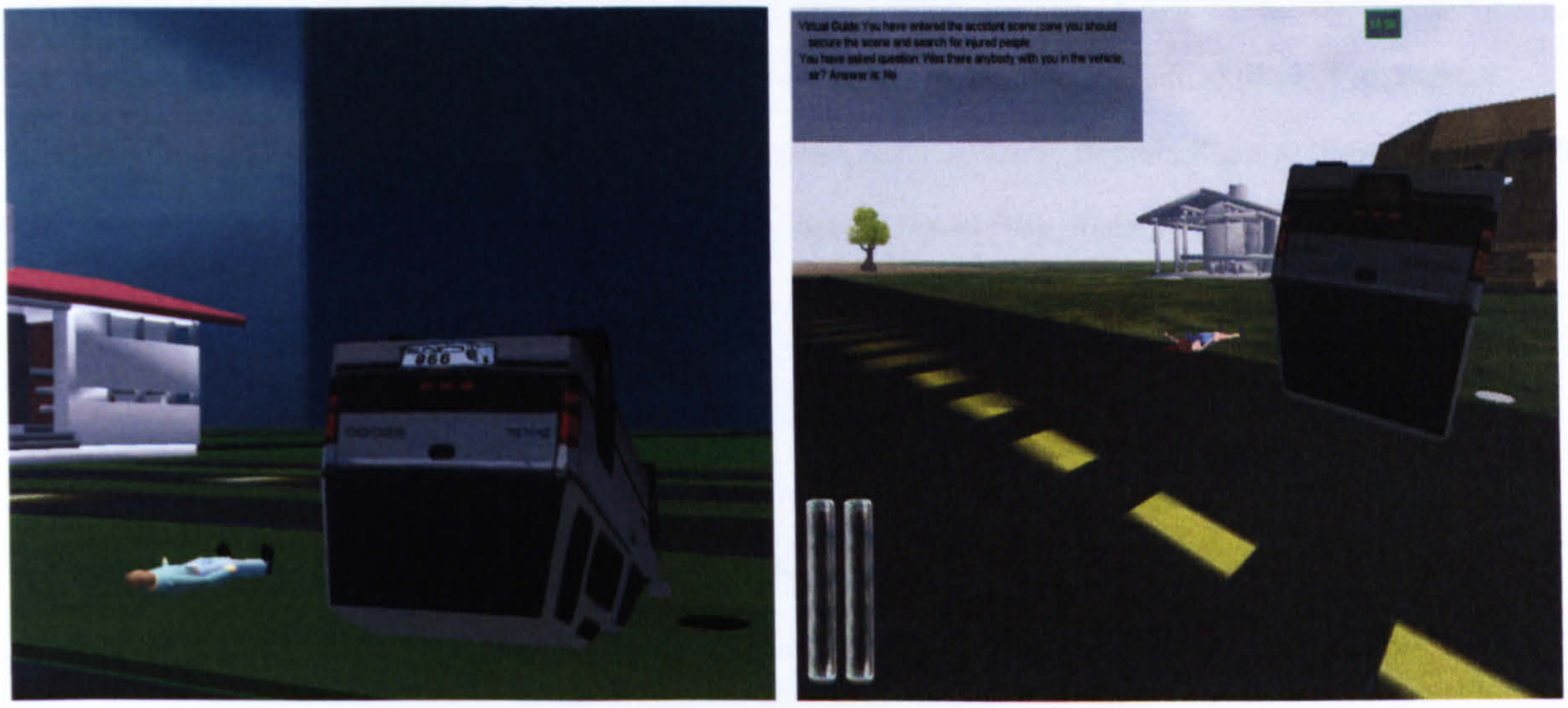


Figure 4.1: (left) Smart terrain running on bespoke engine; (right) the same G-factor running on Torque engine.

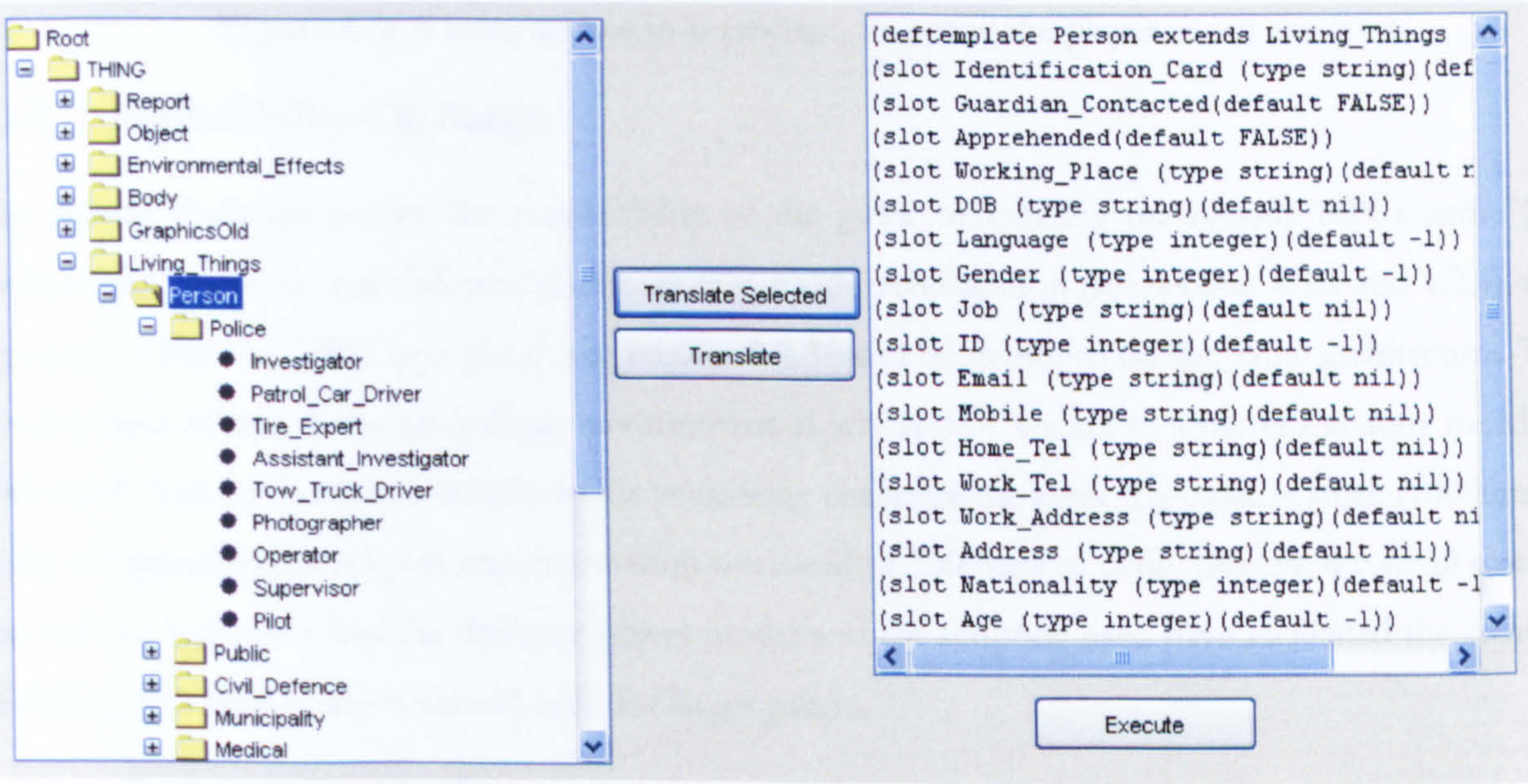


Figure 4.2: Object model used. Translated from tabular format in a database into Jess engine's format which was used as the behaviour engine.

```
(defrule Accident_Scene_Zone_Rule_Enter "Inside accident scene zone"
  (declare (no-loop TRUE))
  ?invPosition <- (Graphics (x ?x)(y ?y)(z ?z))
  ?investigator <- (Investigator (Graphics_Attribute ?invPosition)(Virtual_Guide_Attribute ?virtualGuide)(Zone_Attribute nil))
  ?accidentScene <- (Zone (Name "Accident_Scene_Zone")(Back_Bottom_Right_z ?Back_Bottom_Right_z) (Front_Top_Right_z
  ?Front_Top_Right_z)
  (Back_Bottom_Left_y ?Back_Bottom_Left_y) (Back_Bottom_Left_z ?Back_Bottom_Left_z) (Back_Bottom_Right_x
  ?Back_Bottom_Right_x)
  (Back_Bottom_Right_y ?Back_Bottom_Right_y) (Back_Bottom_Left_x ?Back_Bottom_Left_x) (Back_Top_Right_z
  ?Back_Top_Right_z)
  (Front_Bottom_Right_y ?Front_Bottom_Right_y) (Front_Bottom_Right_x ?Front_Bottom_Right_x) (Front_Bottom_Left_y
  ?Front_Bottom_Left_y)
  (Front_Bottom_Left_x ?Front_Bottom_Left_x) (Front_Top_Right_y ?Front_Top_Right_y) (Front_Top_Right_x
  ?Front_Top_Right_x)
  (Front_Top_Left_y ?Front_Top_Left_y) (Front_Top_Left_x ?Front_Top_Left_x) (Back_Top_Right_y ?Back_Top_Right_y)
  (Back_Top_Right_x  ?Back_Top_Right_x) (Back_Top_Left_z ?Back_Top_Left_z) (Back_Top_Left_y ?Back_Top_Left_y)
  (Back_Top_Left_x  ?Back_Top_Left_x)(Front_Bottom_Left_z ?Front_Bottom_Left_z) (Front_Bottom_Right_z
  ?Front_Bottom_Right_z) (Front_Top_Left_z ?Front_Top_Left_z))
    =>
     (bind ?maxX (max ?Back_Bottom_Right_x ?Back_Bottom_Left_x ?Front_Bottom_Right_x ?Back_Bottom_Left_x))
     (bind ?minX (min ?Back_Bottom_Right_x ?Back_Bottom_Left_x ?Front_Bottom_Right_x ?Back_Bottom_Left_x))
     (bind ?maxY (max ?Back_Bottom_Right_y ?Back_Bottom_Left_y ?Back_Top_Right_y ?Back_Top_Right_y))
     (bind ?minY (min ?Back_Bottom_Right_y ?Back_Bottom_Left_y ?Back_Top_Right_y ?Back_Top_Left_y))
     (bind ?maxZ (max ?Back_Bottom_Right_z ?Back_Bottom_Left_z ?Front_Bottom_Right_z ?Back_Bottom_Left_z))
     (bind ?minZ (min ?Back_Bottom_Right_z ?Back_Bottom_Left_z ?Front_Bottom_Right_z ?Back_Bottom_Left_z))
     (if (and (>= ?maxX ?x)(<= ?minX ?x)(>= ?maxY ?y)(<= ?minY ?y)(>= ?maxZ ?z)(<= ?minZ ?z)) then
      (modify ?investigator (Zone_Attribute ?accidentScene))
      (modify ?virtualGuide (Guide_Messages "You have entered the accident scene zone you should secure the scene and search
       for injured people"))
     else
      (printout t "Enter Rule" crlf))
)
```

Figure 4.3: A rule, written in JessScript, to notify the player about the zone.

4.2.2 Modifiability Challenge

The second challenge probes the modifiability of the game by creating the Moody NPCs game (see section 2.5.1) to show that different G-factors (e.g. traffic investigation (see section 4.2.1 and 4.2.4), and first-person shooter (FPS) type game (see section 4.2.3)) can be deployed on the same architecture. The development of this challenge built an environment in which NPCs react to a player's actions based on their mode (see section 3.5). Similarly to the portability challenge the game logic size is small (100 lines of code) compared to the adapter implementation overhead of 350 lines of code. Despite the small size the test remains indicative that the dynamic object model and the scripting used have facilitated the creation of different G-factors which should hold for larger games.

4.2.3 Performance Challenge

The third challenge aims at getting a performance indicator by running a game where NPCs run for cover every time they spot a human player. The object model created consists of Player, AIPlayer, Waypoint, and MoveTo. To avoid a human player the NPC chooses a waypoint which is in its sight but outside the human's range and moves to that waypoint. The information that needs to be sent from the game engine to the game space is shown in Figure 4.4. The "in sight" variables are considered real-time

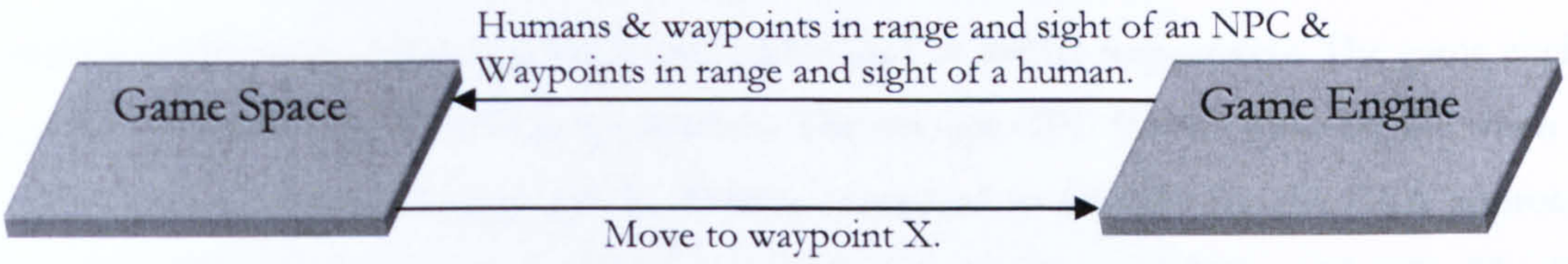


Figure 4.4: An overview of GSA design.

variables (described in section 3.5) because it is not easy to calculate them in the game space as they depend on many non-game objects such as the terrain, houses, trees, etc.

To get a performance indicator a player was simulated to be running continuously around the path shown in Figure 4.5 for 30 minutes. Using this simulation two performance tests were run to contrast the overheads of a game developed with the typical development approach to one developed using GSA. The performance overheads measured are: fps, CPU, memory, and network (for the test using the game space). The two tests were run on a laptop with a 3.20 GHz Pentium 4 CPU and 2 GB RAM. Figure 4.6 contrasts the fps recorded for the two runs. Table 4.1 shows the average reduction in fps reported is 11.69% when following the GSA approach. This average fps reduction is relatively large for a small game and therefore further tests need to be performed to get a better indication of how this reduction will scale with the game size. Figure 4.7a and Table 4.2 shows the CPU and memory usage for

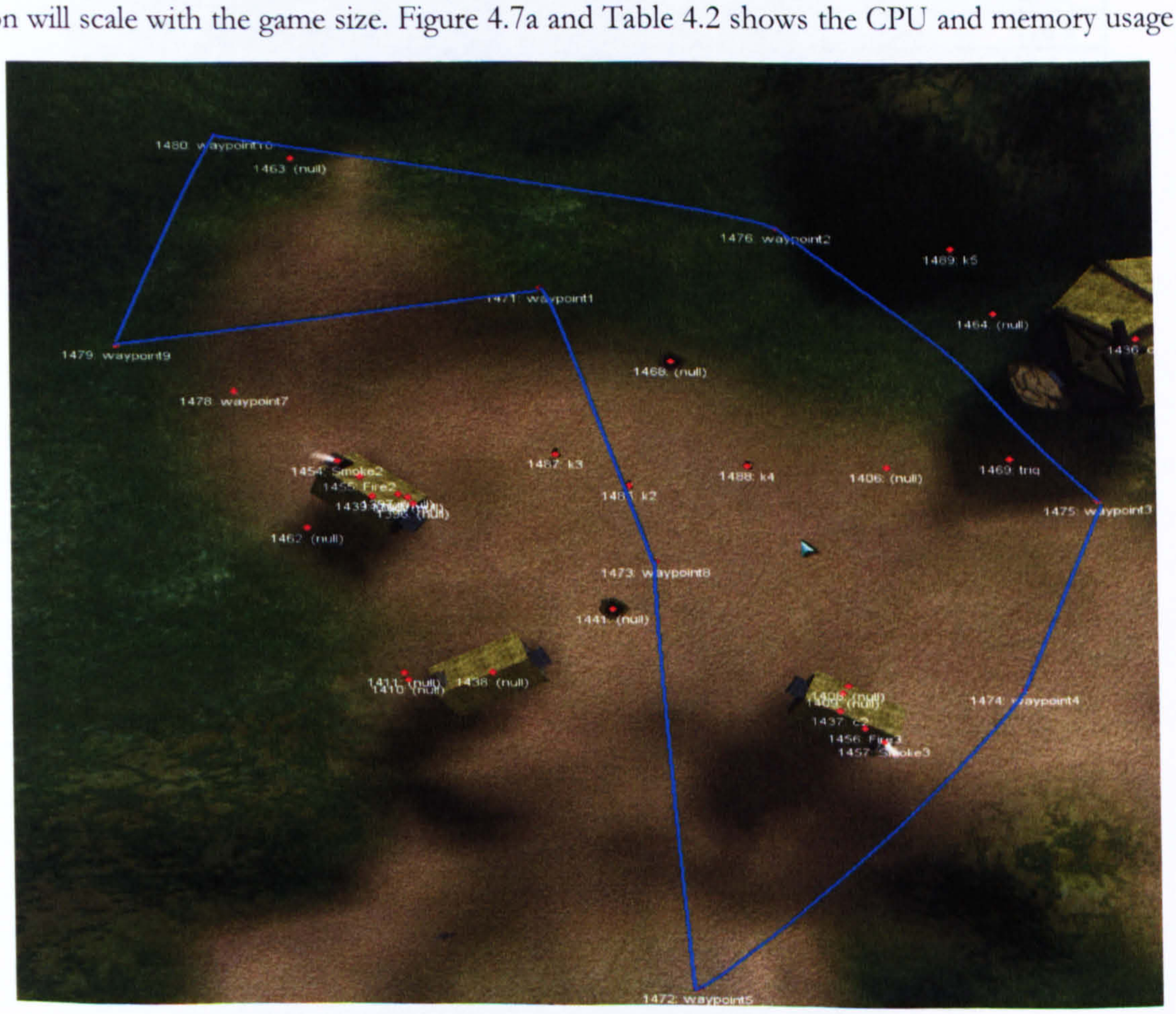


Figure 4.5: The path followed by the player for the performance test.

a test built using the typical game development approach while Figure 4.7b and Table 4.3 shows the same results for a test built using the GSA approach. The average game space's CPU and memory overheads reported for the GSA approach are: 1.51% and 16.49MB respectively. The game engine runs at almost the same CPU speed in both approaches. The average CPU for the game engine when running using the typical development approach is 98.80% compared to 98.43% for the GSA approach. The machine's total CPU also reported a small increase in the average from 53.74% to 55.60%. The CPU and memory usage show that the game space and the game engine can run on the same machine due to the game space's small CPU and memory usage. This finding needs to be verified against larger games as game engines are CPU-intensive. Finally the implementation overhead for the adapter was 300 lines of code for a game logic of 70 lines. To get an early indication of the network load GSA can handle, Figure 4.8a and Figure 4.8b show the network messages sent and received by the game engine and the game space respectively. Figure 4.8c shows the average throughput per second used during the test is 55.49 messages sent from the game engine and received by the game space. This test needs to be expanded further to locate GSA's breaking point.

Table 4.1: Contrasting the fps for the typical game development approach and the GSA approach.

	Mean	Median	Std	Min	Max
fps (Typical Approach)	106.83	111.5	12.34	43.4	127.9
fps (GSA's Approach)	94.35	97.9	14.45	45.8	116.8
Difference	12.49	13.6	-2.11	-2.4	11.1
Difference (%)	11.69	12.20	-17.06	-5.53	8.67

Table 4.2: The CPU and memory results for the typical game development approach.

	Mean	Median	Std	Min	Max
Game Engine CPU(%)	98.80	100	1.67	90.63	101.56
Game Engine Memory (MB)	80.03	80.13	0.77	68.24	80.13
Total CPU(%)	53.74	53.91	2.39	50	60.94

Table 4.3: The CPU and memory results for the GSA approach.

	Mean	Median	Std	Min	Max
Game Engine CPU(%)	98.43	98.44	1.92	89.06	101.56
Game Engine Memory (MB)	80.03	80.13	0.77	68.24	80.13
Game Space CPU(%)	1.51	0	2.36	0	23.44
Game Space Memory (MB)	16.49	16.56	0.22	15.47	16.57
Total CPU(%)	55.60	55.47	3.18	50	71.09

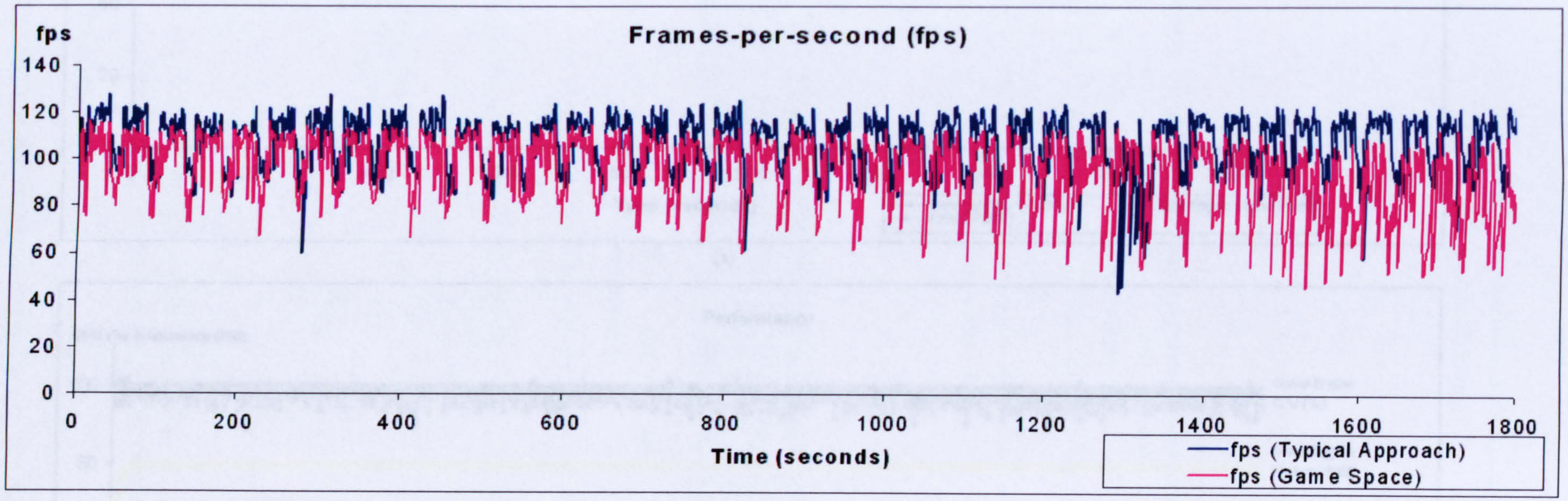
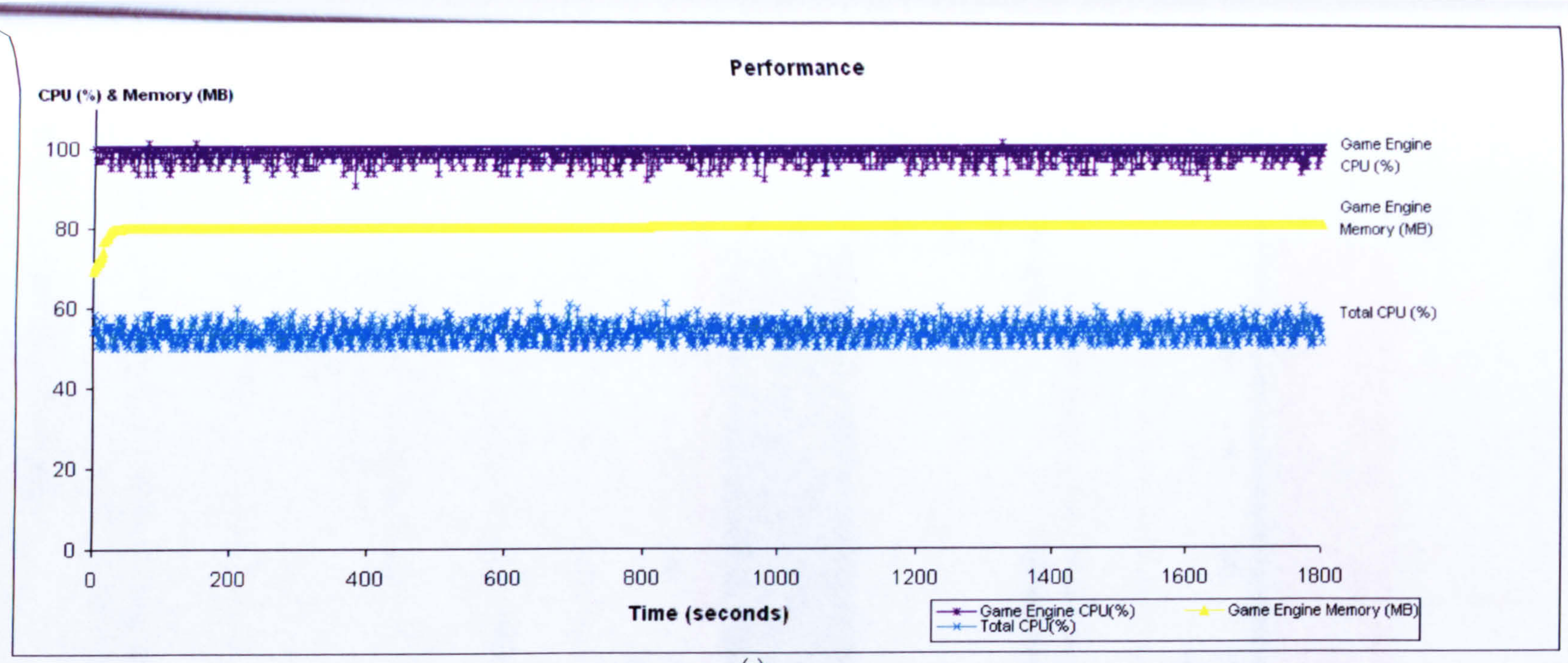
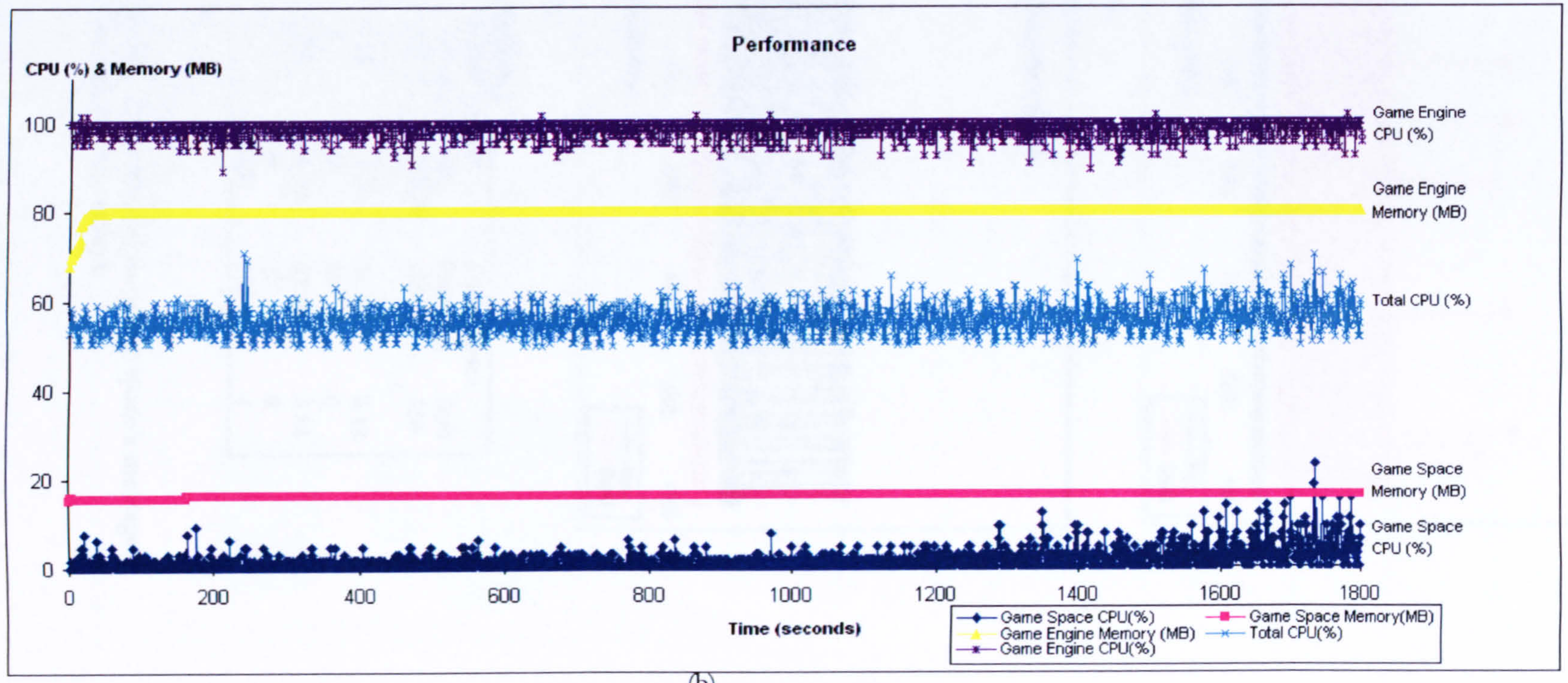


Figure 4.6: Contrasting the fps for the typical game development approach and the GSA approach.

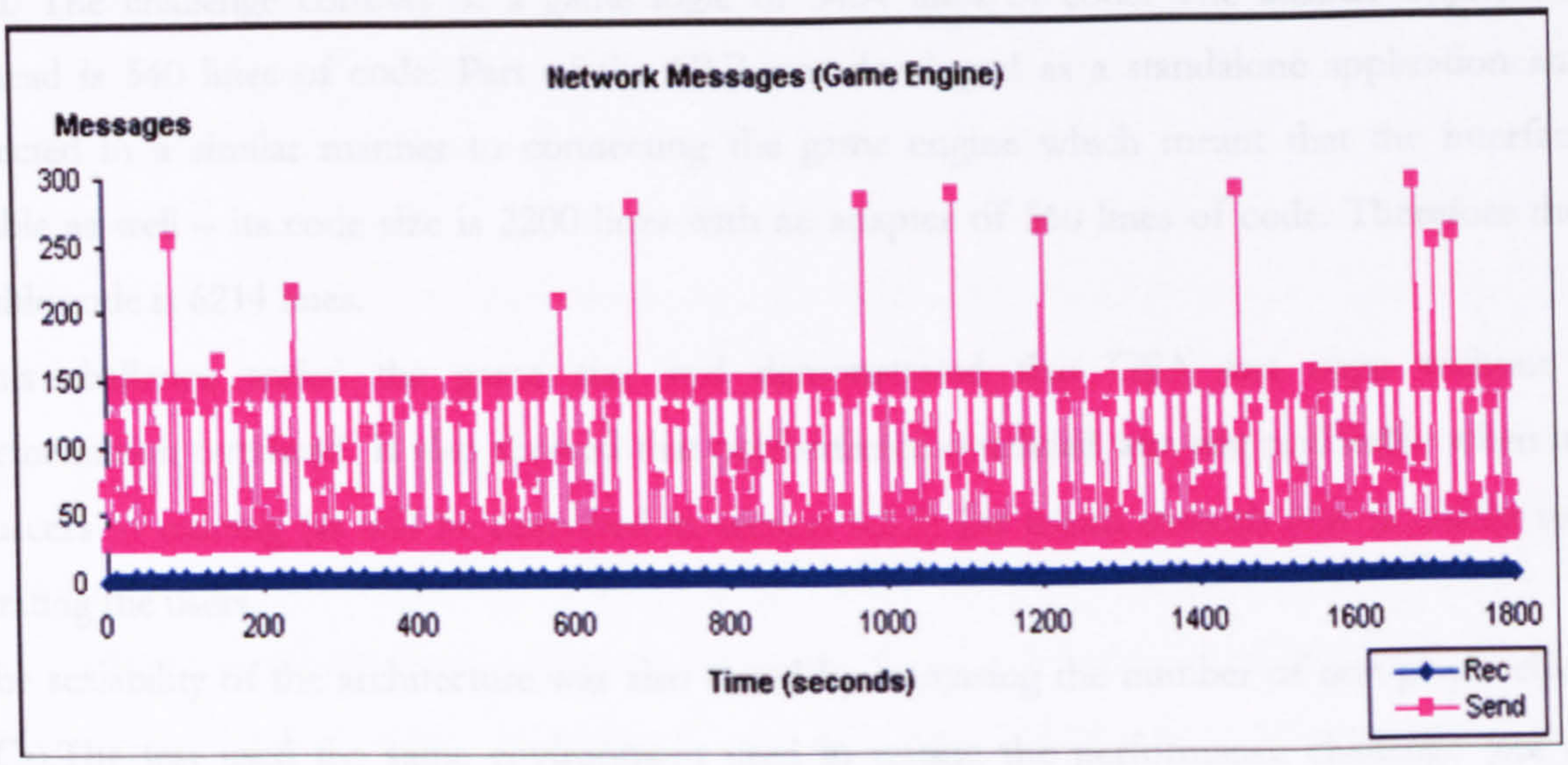


(a)

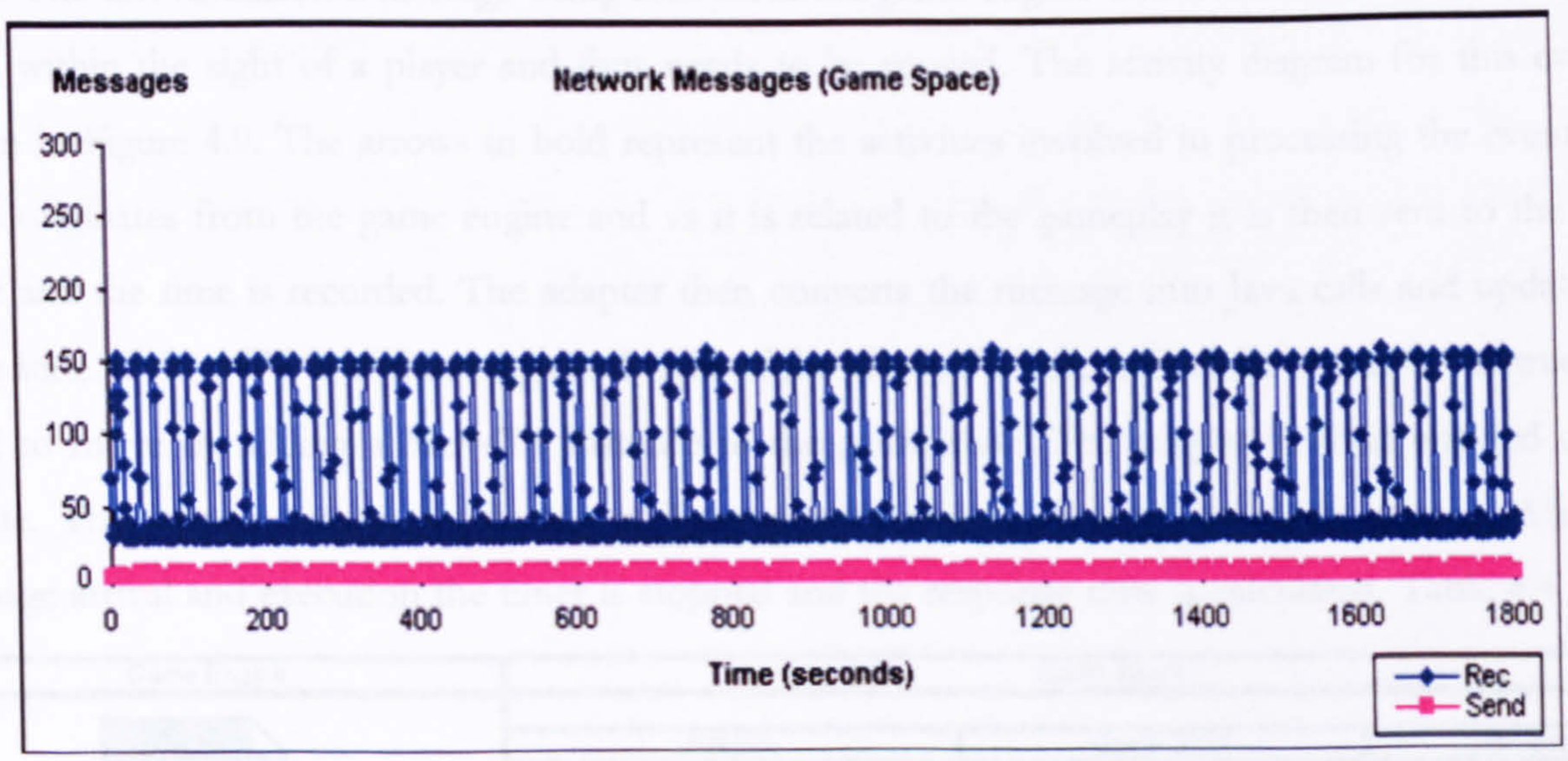


(b)

Figure 4.7: Contrasting the CPU and memory results for (a) a typical game development approach and (b) the GSA approach.



(a)



(b)

GSA's Approach

	Game Engine		Game Space	
	Received	Sent	Received	Sent
Total Messages	336	104876	104876	336
Messages Throughput/Second				
Mean	0.18	55.49	55.49	0.18
Median	0	29	30	0
Std	0.81	46.76	45.69	0.81
Min	0	26	27	0
Max	4	289	155	4

(c)

Figure 4.8: Network overhead: (a) the game engine's messages, (b) the game space's messages, and (c) the messages throughput per second.

4.2.4 Scalability Challenge

The fourth challenge examines the architecture's scalability by developing a serious game for traffic accident investigators (described in chapter 6). The previous challenges had a game logic containing a

maximum of 100 lines of code. This challenge wanted to scale that and examine how the architecture coped. The challenge consists of a game logic of 3454 lines of code. The adapter implementation overhead is 540 lines of code. Part of the GUI was developed as a standalone application and was connected in a similar manner to connecting the game engine which meant that the interface was portable as well – its code size is 2200 lines with an adapter of 560 lines of code. Therefore the total portable code is 6214 lines.

This challenge scaled the game size and demonstrated that GSA can cope without major implementation overhead. It also showed that performance overhead was not noticeable when judging its success in training (as will be described in section 6.5.3) for which smooth play is crucial to avoid frustrating the users.

The scalability of the architecture was also tested by increasing the number of non-player characters (NPCs).The test used the same environment used in testing the performance challenge (see section 4.2.3). The test simulated a message being sent from the game engine which indicates that an NPC has come within the sight of a player and thus needs to be moved. The activity diagram for this event is shown in Figure 4.9. The arrows in bold represent the activities involved in processing the event. The event originates from the game engine and as it is related to the gameplay it is then sent to the game space and the time is recorded. The adapter then converts the message into Java calls and updates the game state. If the NPC is within range and sight of the player then the behaviour controller instructs the NPC to move by adding a MoveTo instance in the game state. The adapter is then notified of this update. The adapter creates a message in TorqueScript and sends it to the game engine. After the message arrival and execution the timer is stopped and the response time is calculated. Table 4.4 shows

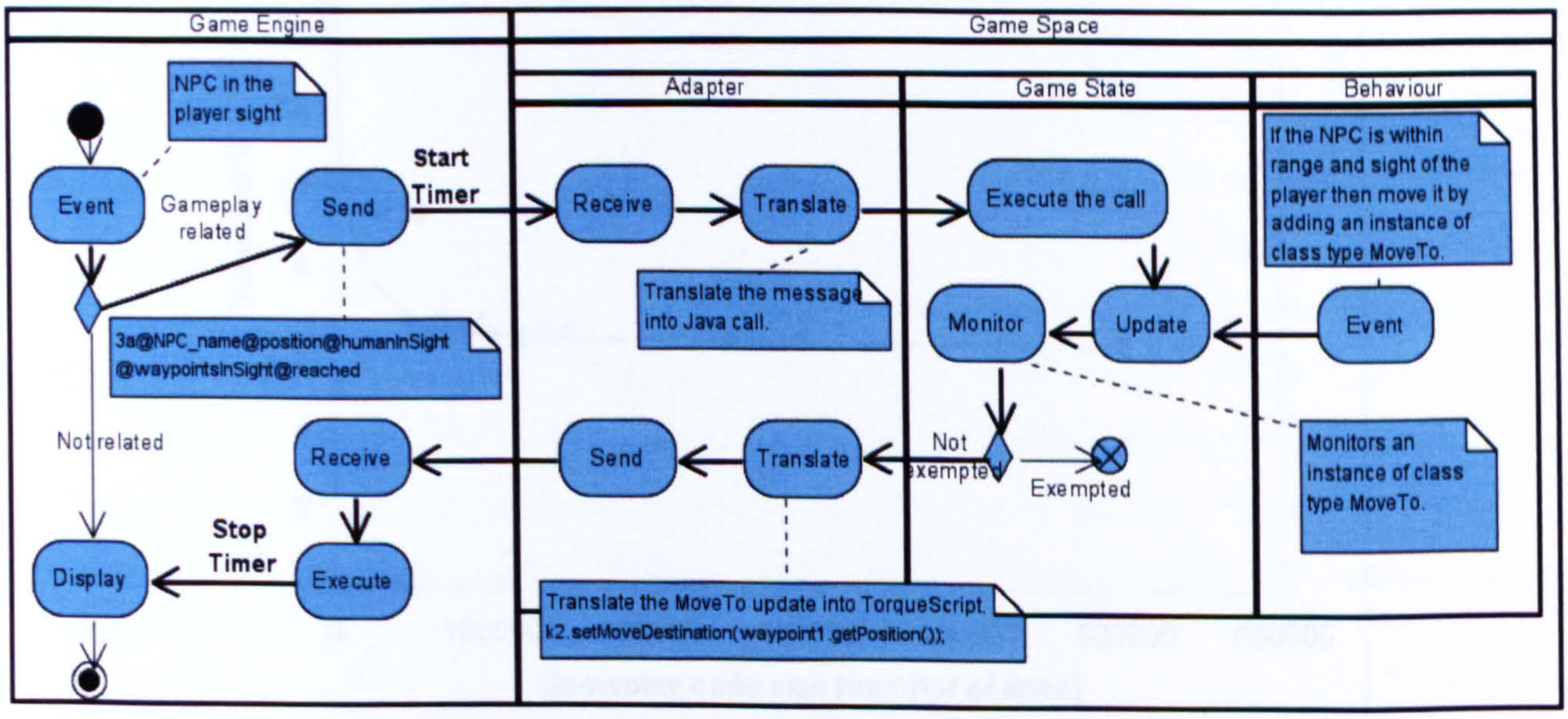


Figure 4.9: Activity diagram for scaling the number of NPCs test.

Table 4.4: Average response time in milliseconds as the number of NPCs is scaled.

Number of Entities	1000	10,000	100,000	500,000
Average Response (milliseconds)	13.2	26.3	155.7	770.9

the average response time in milliseconds as the number of NPCs is increased. This shows that for 500,000 entities GSA remains able to respond to events in under 1 second. However, beyond that the response time becomes a concern. A possible contributing factor to the increase in the response time is due to the way the instances are stored and retrieved in the game space.

Currently game space does not sort the instances in any particular order. They are added to a vector data structure in the same order they are created in. Sorting the instances by class type (e.g. NPC and Player) and by the instances names could improve the response time.

4.2.5 Implementation Overhead

The implementation overhead of the adapters compared to the game logic sizes was described for each of the challenges presented in the previous subsections. To find out how the adapter code scales against the game logic code a forecast[2] was conducted on the implementation overhead using the game logic code for the four challenges and the implementation overheads of their adapters. The result of the forecast trend line produced is shown in Figure 4.10. Initially the overhead starts high with a small game logic code size and starts to drop as the code size increases until it stabilises at around 6% for code of size between 100,000 and 500,000 lines. So for a game like 'Gabriel Knight 3' with 40,000 lines of code

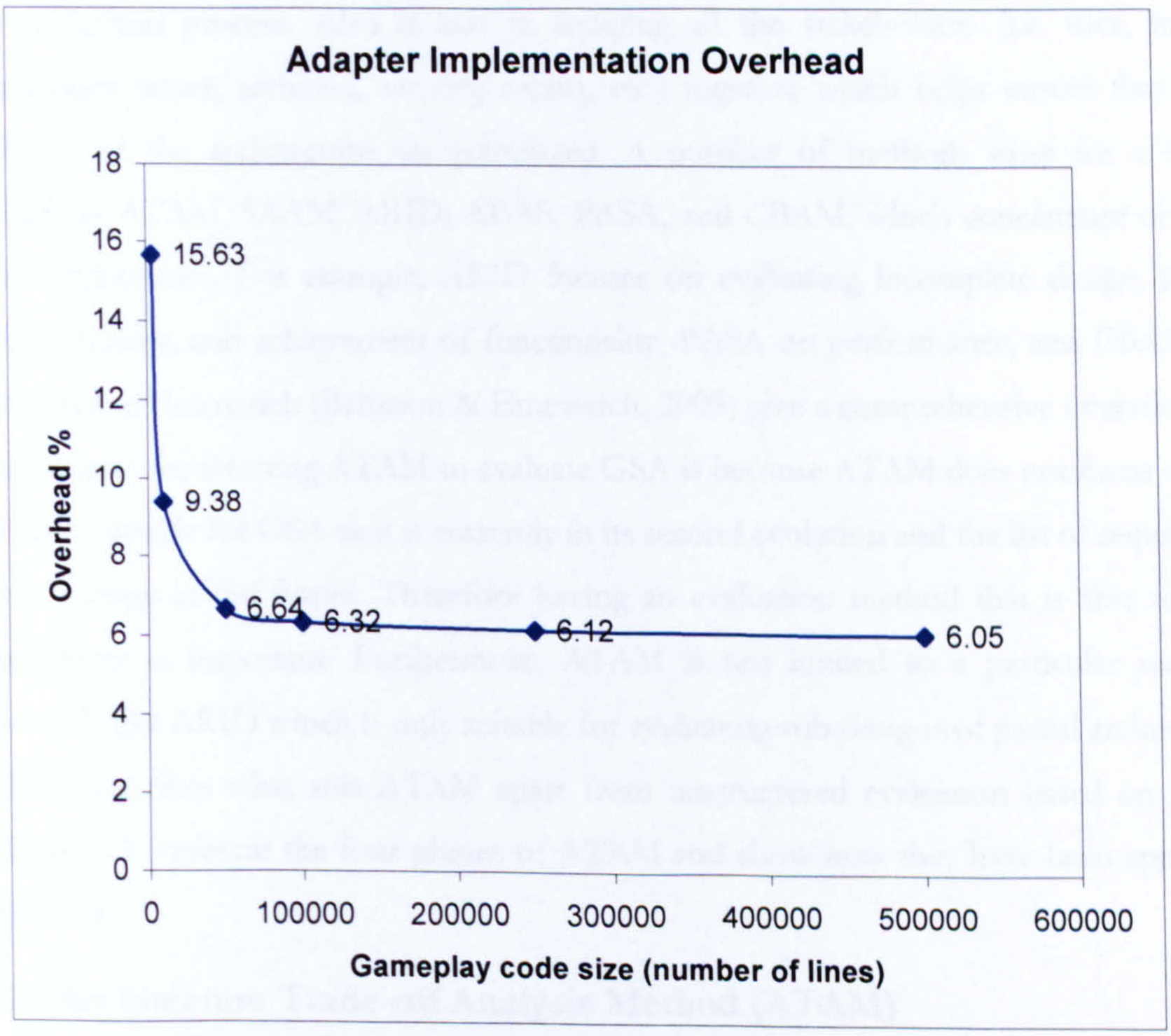


Figure 4.10: The adapter code forecast compared to the game logic code size.

[2] http://support.microsoft.com/kb/828236 (accessed 24/8/2007).

(for scripts and logic) the forecasted adapter overhead cost is 2,700 lines of code which is 6.5% of the game logic size.

4.2.6 Summary

The unstructured evaluation of throwing challenges at an architecture manages to reveal how the GSA approach addresses the quality attributes. However, it suffers from the following problems:

- Challenges do not explicitly establish links between the quality attributes desired and the decisions made and the trade-offs made between these decisions. For example it is not clear what architectural decisions have contributed to the portability attribute and what have undermined it.
- There is no guarantee that all the architecture's components are going to be exercised during the evaluation.
- There is no guarantee that the challenges thrown are not redundant since they could be evaluating the same architectural decisions.

4.3 Structured Evaluation

The appeal of a structured evaluation is that it has a better chance at exercising the whole architecture by guiding the evaluation process. Also it aids in bringing all the stakeholders (i.e. user, maintainer, developer, manager, tester, architect, security expert, etc.) together which helps ensure that the main quality attributes of the architecture are prioritized. A number of methods exist for a structured evaluation, such as ATAM, SAAM, ARID, ABAS, PASA, and CBAM, which concentrate on different aspects of an architecture. For example, ARID focuses on evaluating incomplete design, SAAM on modifiability, variability, and achievement of functionality, PASA on performance, and CBAM on cost benefit. Bahsoon and Emmerich (Bahsoon & Emmerich, 2003) give a comprehensive overview of these methods. The reason for selecting ATAM to evaluate GSA is because ATAM does not focus on specific attributes. This is suitable for GSA as it is currently in its second evolution and the list of required quality attributes may change in the future. Therefore having an evaluation method that is able to scale for additional attributes is important. Furthermore, ATAM is not limited to a particular stage of the development cycle like ARID which is only suitable for evaluating sub-designs of partial architectures.

Section 4.3.1 describes what sets ATAM apart from unstructured evaluation based on its output. Sections 4.3.2 to 4.3.5 present the four phases of ATAM and show how they have been applied in the evaluation of GSA[3].

4.3.1 The Architecture Trade-off Analysis Method (ATAM)

ATAM reveals how well the architecture satisfies its requirements and describes how the quality attributes interact and trade-off against one another (Clements et al., 2001). It comprises of 9 steps

[3] The evaluation process has been carried out by the author and help of the supervisor.

grouped into four phases: presentation (3 steps), investigation and analysis (3 steps), testing (2 steps), and reporting (1 step). Contrasting ATAM's output to the unstructured evaluation results, which quite often answer the challenge with yes or no, or with some metrics such as network load or fps, highlights the following reasons for using ATAM:

- ATAM provides clear articulation of the correlation between the architectural decisions and the quality attributes.
- ATAM identifies risks associated with each architectural decision.
- ATAM reveals sensitivities for which a slight change has significant impact on the architecture.
- ATAM identifies trade-offs made which are decisions affecting more than one quality attribute (Kazman et al., 1998).
- ATAM identifies nonrisks which are assumptions that must be held for these to remain as nonrisks. If changed these have to be rechecked.
- ATAM provides more direct probing of the architecture in the form of a utility tree (shown later in Figure 4.11) which transfers ambiguous requirement statements to more concrete measurable scenarios.

4.3.2 Phase 1: Presentation

The first phase aims to exchange information between the evaluation team through a series of presentation sessions. The first step describes the ATAM phases to the team. The second step involves describing the architecture's quality attributes (i.e. portability, modifiability, and performance, as described in section 3.3). The third step presents the architecture and the focus is to show how it addresses its attributes. The information required for the second and third steps was described chapter 3.

4.3.3 Phase 2: Investigation and Analysis

This phase comprises of three steps: identifying the architectural decisions made (step 4), generating the quality attribute utility tree (step 5), and analysing the architectural decisions (step 6). Although in perfect documentation all the architectural decisions should be listed that is not always the case, as was found when evaluating the BCS project where the documentation was too vague and did not include discussion of the architectural approaches used (Clements et al., 2001). Therefore, the evaluation team has the responsibility to elicit architectural decisions from the architecture documentation, and the architect's presentation. Table 4.5 summarizes the eight architectural decisions described in chapter 3.

The utility tree elicits the quality attributes down to the scenario level to provide a mechanism for translating architectural requirements into concrete practical scenarios. The utility tree also aids in prioritising the quality attributes. According to Clements et al. (Clements et al., 2001), this step is considered a crucial step which guides the rest of the analysis and without which the evaluators could

Table 4.5: Summary of the architectural decisions.

Architecture Decision (AD)	Purpose
AD 1: MVC	Used to separate the G-factor from the game engine by using an adapter.
AD 2: Asynchronous messaging	Used to exchange messages between the game space and the game engine.
AD 3: On-the-fly scripting	Used in the adapter to manipulate the game engine and the game space.
AD 4: Dynamic object model	Used to create the object model.
AD 5: API	Used to allow alternatives to controlling the game for games that find scripting too slow.
AD 6: Interoperate with external tools	Allows control of the game through external tools (e.g. rules engine).
AD 7: Scripting mapping table	Used in the adapter to translate between the game engine and the game space.
AD 8: Object mapping table	Used in case a unique identifier could not be set for the game objects across the game engine and the game space.

spend valuable time analysing the architecture without addressing the important quality attributes as far as the stakeholders are concerned.

Figure 4.11 shows the utility tree for GSA. There are three levels in the tree: the quality attributes level, the refinements level, and the scenarios level. The aim of the refinement is to decompose the quality attribute further, if possible. The last level holds the scenarios in a concrete form. This level also holds the scenarios' rankings which decide their priorities. In ATAM, ranking format can differ based on the participants' preferences. It could be a scale based on 0 to 10, or relative ranking such as High (H), Medium (M), and Low (L). The ranking is done using two variables: importance and difficulty. Importance states how important the scenario is for the success of the architecture and difficulty describes the degree of difficulty in achieving that scenario. For the work described in this thesis relative ranking was used to prioritise the scenarios.

ATAM relies heavily on scenarios to turn ambiguous quality statements such as "the system should be modifiable" into something more concrete, as shown in Figure 4.11 by the scenarios: M1.1, M2.1, M2.2, M2.3, and M3.1. The benefits of using scenarios are threefold (Clements et al., 2001). First, they are simple to create and understand. Second, they are inexpensive. Third, they are effective. The mechanism followed by ATAM to describe these scenarios uses a three-part format: stimulus, environment, and response. The stimulus describes what initiated an interaction with the architecture. The environment describes the state of the architecture when the interaction takes place. The response explains how the architecture reacts to the interaction. ATAM also encourages eliciting scenarios by thinking about different scenario types. The three types used are: use case scenarios, growth scenarios, and exploratory scenarios. The first describes typical uses of the architecture, the second lists anticipated changes, and the third lists extreme changes aimed to stress test the architecture. These will be covered in section 4.3.4. Some of the scenarios are related to the scenarios used in the unstructured evaluation, but differ in that

they have been created in a more structured way which aims to reduce the number of tests that need to be implemented. This is made possible by the creation of an analysis table for each scenario which shows the architectural decisions affected, as will be described in the next step.

Nine scenarios have been elicited for the architecture as shown in the utility tree (Figure 4.11). As an example, consider the scenario elicited to prove that the architecture supports portability (P1.1 in Figure 4.11). This is similar to the first challenge described in the unstructured evaluation (see section 4.2.1). This scenario is ranked with high for importance because it is the main quality attribute for the success of the architecture. The difficulty rate is set to medium, as the change requires creating a new adapter between the new engine and the game space.

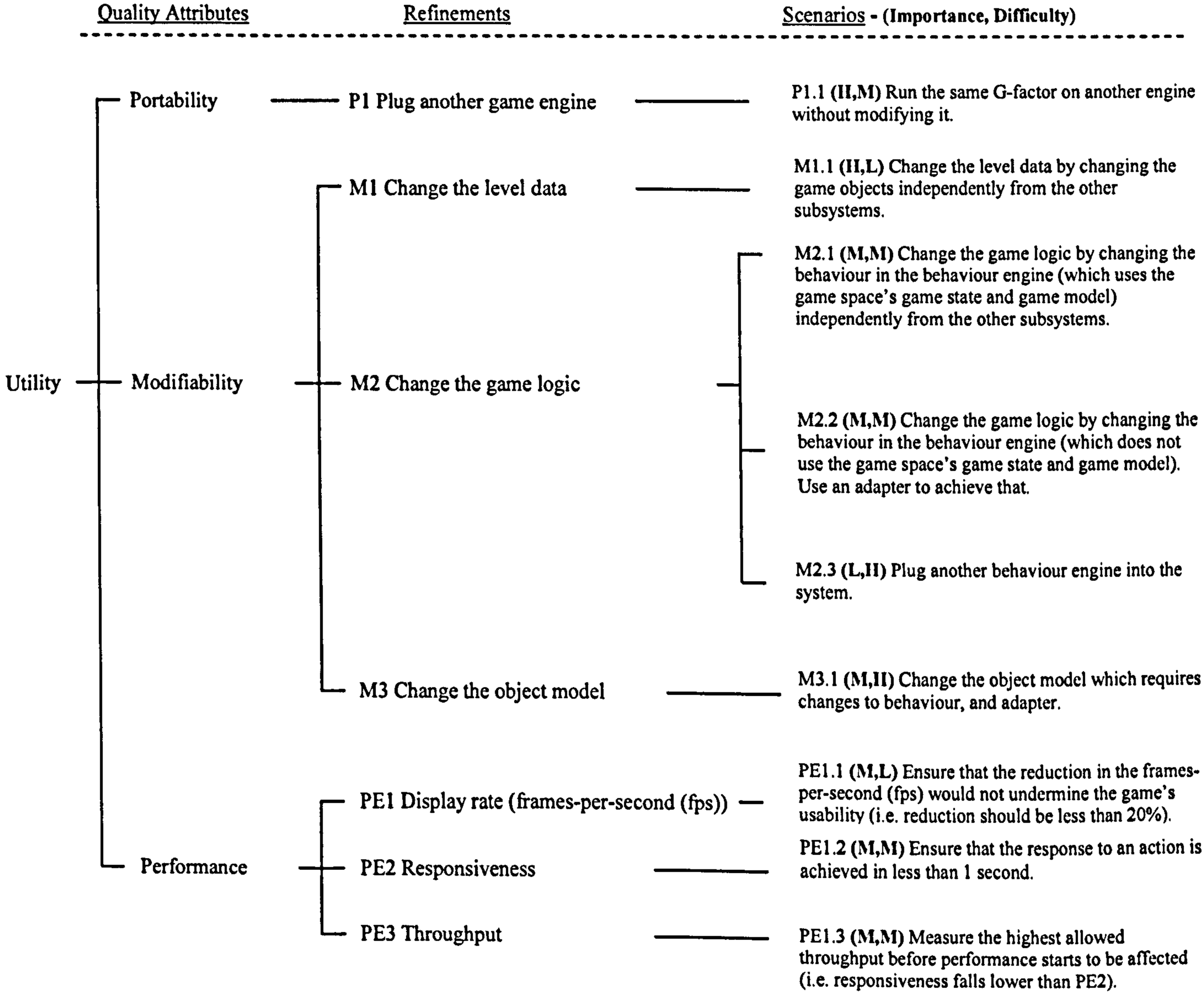


Figure 4.11: The utility tree used to guide the evaluation.

The last step in this phase is analysing the architectural decisions made to examine how well they correlate to the quality attributes. This step is where the architect's decisions come under close scrutiny. The output of this step is a detailed description of the decisions that are aiding the pursued quality

attributes and the ones that undermine them. The process followed in this step looks for well-known weaknesses and sensitivity points with the architectural decisions that affect this scenario. The process also looks for the interactions and trade-offs made with other decisions and how that affects the quality attribute in question. Table 4.6 shows the outcome of analysing the portability scenario, for which a number of key architectural decisions have been identified. Similar tables are created for each scenario listed in the utility tree. As illustrated in the table, for each architectural decision this step reveals sensitivities, trade-offs, risks, and nonrisks, which are described in Figure 4.12.

As shown in the table, the first decision affecting this scenario is the MVC decision (AD1). MVC has one trade-off, two risks, and one nonrisk associated with it. The trade-off made (T1) favours portability over performance, as shown in Figure 4.12. The first risk (R1) is caused by the tight-coupling between the controller and the model which is a known liability of using this pattern (Buschmann et al., 1996). The second risk (R2) is caused by the difficulty in maintaining the data integrity between the two states. The nonrisk (N1) exists because of the decision taken to remove the other liability of using MVC – the removal of the direct link between the view and the controller, as described in section 3.3.1.

Table 4.6: Portability scenario analysis (see Figure 4.12 for description of S1, S2, etc).

Analysing Scenario	P1.1			
Scenario	Plug another game engine to the architecture			
Attributes	Portability			
Stimulus	Running the same game on another game engine			
Environment				
Response	Should be achieved by adding an adapter to connect the game space and the newly added game engine			
Architecture Decision	**Sensitivity**	**Trade-off**	**Risk**	**Nonrisk**
AD1: MVC		T1	R1,R2	N1
AD2: Messaging	S1,S2	T2	R3	N2
AD3: On-the-fly scripting		T3	R4	N3
AD4: Dynamic object model		T4		
AD6: COTS behaviour engine	S1,S2		R2	N4

The analysis of the messaging approach (AD2) has revealed two sensitivities, one trade-off, one risk, and one nonrisk. The two sensitivities (S1, S2) are introduced because of concerns over network latency and messages load. Increasing either of these would have negative impact on the architecture's performance. The risk (R3) is introduced by the tight-coupling as a consequence of using pre-determined messages for the messages arriving from the game engine, which means changes to them would require changing the adapter. Using messaging middleware (such as publish-subscribe (Buschmann et al., 1996)) could improve the portability and loosen the coupling, however it would have an adverse affect on performance and the trade-off made here favours the performance. The nonrisk (N2) exists because of using asynchronous messaging as it avoids a negative impact on the frame rate that could be caused by the synchronous mechanism.

Sensitivities:

- S1: There is a concern over network latency.
- S2: There is a concern over messages load.
- S3: In the event that a single unique identifier cannot be set this architecture decision becomes very sensitive to any modification as they have to be added manually to the table.
- S4: Using dynamic object model allows for better game model scalability but it makes the architecture very sensitive to change as the change propagates to behaviour and adapters.

Trade-offs:

- T1: Portability (+) and Modifiability (+) vs. Performance (-) – separating the architecture using MVC adds an overhead for exchanging information between the different parts which affects the performance.
- T2: Portability (+) vs. Performance (-).
- T3: Portability (+) and Modifiability (+) vs. Performance (-) – on-the-fly scripting allows better portability and modifiability but runs slower than pre-compiled code.
- T4: Portability (+) and Modifiability (+) vs. Performance (-) – using dynamic object model allows for having a game model that is independent from the game engine's game model however it has adverse effect on performance.
- T5: Modifiability (-) vs. Performance (+).

Risks:

- R1: The risk is caused by the tight-coupling between the controller and the model which is a known liability of using this pattern.
- R2: Data integrity.
- R3: The risk is introduced by the tight-coupling as a consequence of using pre-determined messages for the messages arriving form the game engine which means changes to them would require changing the adapter.
- R4: The risk is raised as a consequence of the game engine or the game space not having fully exposed their functionality through the scripting.
- R5: If no unique id can be set this means the mapping table should be done manually.

Nonrisks:

- N1: The nonrisk exists because of the decision taken to remove the other liability of using MVC – the removal of the direct link between the view and the controller described earlier.
- N2: The nonrisk risk exists because of using asynchronous mechanism as it avoids negative impact on the frame rate that could be caused by the synchronous mechanism
- N3: The nonrisk is with the use of the functionality exposed for scripting which should stay compatible.
- N4: The nonrisks assumes that the behaviour engine requires the game state to be replicated to its own working memory.
- N5: A one-to-one mapping for objects across both game states is established by having a unique identifier and if that is not possible the objects mapping table handles that.
- N6: The nonrisk is with the use of API which should stay compatible.

Figure 4.12: The analysis of sensitivities, trade-offs, risks and nonrisks for the Utility Tree in Figure 4.11.

On-the-fly scripting (AD3) has one trade-off (T3), one risk (R4), and one nonrisk (N3). The trade-off favours portability over performance since using scripting is slower than using pre-compiled code. The risk raised is a consequence of the game engine not allowing full exposure to its functionality through scripting. The nonrisk is the use of an API which should stay compatible as the API evolves.

The use of a dynamic object model (AD4) has one trade-off (T4) associated with it. The trade-off is between portability and performance as dynamic models improve portability but affect performance because they are not hard-coded.

The final decision that could affect portability is the use of the commercial of the shelf (COTS) behaviour engine (AD6) which has two sensitivities (S1, S2), one risk (R2), and one nonrisk (N4). The sensitivities are similar to those caused by the messaging decision. The risk is similar to the second risk caused by MVC which relates to maintaining data integrity. The nonrisk assumes that the behaviour requires the game state to be replicated to its working memory.

4.3.4 Phase 3: Testing

This phase consists of two steps: brainstorming and prioritising scenarios (step 7), and analysing the architectural decisions (step 8). The first looks similar to generating scenarios for the utility tree in phase 2, however the aim here is different. There the stakeholders were asked to generate scenarios based on given quality attributes, whereas here they are asked to ignore that and give general scenarios. The goal of this step is to widen the spectrum from which scenarios can be elicited. Once the scenarios are generated the ones that address the same concern are merged together. Then the scenarios are prioritised. The prioritisation process here differs from the prioritisation approach adopted earlier. Here each stakeholder is given a number of votes (usually ATAM suggests 30 percent of the number of scenarios rounded up) and they use these votes to prioritise the scenarios. Table 4.7 specifies the scenarios generated in this step for GSA.

Table 4.7: The scenarios generated in step 7.

Type	Scenario
Use case	1 - Run two games from the same domain. 2 - Run two games from two different domains.
Growth	3 - Interoperate between two game engines. 4 - Run two behaviour engines (internal and external).
Exploratory	5 - Increase the load by increasing the number of NPCs the user can interact with. 6 - Increase the load by simulating multiple number of simultaneous players. 7 - Run the game space on the same machine as the game engine. 8 - Run the behaviour engine and the game space on the same machine as the game engine.

The scenarios generated are compared to the ones generated in the utility tree creation step. The goal is to plug these scenarios into the utility tree. While doing so one of three things could happen. First, the scenario might match an already-existing scenario and no further action is required (e.g. scenarios 1 and 2 are already addressed by the scenarios generated for M1, M2, and M3 in the utility tree (see Figure 4.11)). Second, the scenario will fall under a new leaf node of an existing branch; if so it is then placed under the branch (e.g. scenario 4 falls under M2 and scenarios 7 and 8 can fall under the performance attribute). Third, the scenario might not fit in any branch and in that case it means it is expressing a new quality attribute which has not been accounted for and thus the quality attribute is added to the tree (e.g.

interoperability is an additional attribute identified by scenario 3, and scenarios 5 and 6 fall under the scalability attribute which can be added to the utility tree). The second step in this phase is to analyse the architectural decision following similar tasks to the ones carried out in phase 2. This second step of this phase was not done for GSA. Instead, it remains as part of the future work.

4.3.5 Phase 4: Reporting

The only step in this phase is presenting the findings to the stakeholders (step 9). The findings are usually summarized in a document containing the following outputs: architectural decisions used, scenarios and their prioritisation, utility tree, risks and nonrisks, and finally sensitivity and trade-off points. One of the strengths of ATAM is that at the end of the evaluation process the results already exist since in each step the documentation is quite comprehensive. Thus, generating the final document becomes a simple task of merging the outputs of previous steps. Also suggested in this step is to generate risk themes by grouping risks discovered by some common factors. The risk themes generated for GSA are:

- Performance is affected by the separation decisions made (MVC, scripting, and messaging) to achieve portability.
- The architecture relies on the ability to set a unique identifier for corresponding objects otherwise it severely affects modifiability.
- The data integrity across the different game states is at risk.
- There is a danger if the message load increases that the game space becomes the bottleneck in the architecture.

4.4 Summary

This chapter has evaluated GSA to measure how well it manages to achieve the three main quality attributes (e.g. portability, modifiability, and performance). The evaluation process followed two types of evaluations. GSA's evaluation started by following the unstructured approach of using challenges, as described in section 4.2. These challenges served their purpose of revealing how well the architecture can cope with them. However, there was no easy way to establish the correlation between the challenge and what architectural decisions have supported or undermined it. Furthermore the unsystematic way of generating scenarios meant that some time was unnecessarily spent in implementing different tests when one could have served all the challenges (e.g. the implementation of SGTAI used in challenge 4 could have been used to test all the previous challenges). This could be attributed to the incomplete overall evaluation picture due to lack of systematic guidance. Although, there is no guarantee that a structured evaluation would not produce redundant probing since, just like the unstructured evaluation, it is also scenario-based. However, the chances are reduced due to the fact that the generation of scenarios is guided by using a utility tree (see Figure 4.11) in which all the scenarios are identified. This serves two purposes. The first purpose is once all the scenarios are present the experimentation can begin by

choosing a test where preferably all these scenarios can be addressed. The second purpose is that it describes the architectural decisions that are going to be analyzed by the scenario which means any repetitive probing can be identified.

The problem with scenario-based evaluation which both unstructured and structured evaluations use is that the evaluation is only as good as the scenarios generated, which in turn depends on the stakeholders in the evaluation team. Although there are measures put in place to ensure the selection includes all the important personnel (i.e. architects and domain experts), the fundamental problem still persists.

Nevertheless, the structured evaluation using ATAM has delivered on its two main promised outputs: sensitivities and trade-offs, and architectural risks. The most important risks found have been described in the risk themes (see section 4.3.5). ATAM has also classified the architectural decisions according to how they affect the architecture (support or undermine) by discussing their sensitivities, trade-offs, risks, and nonrisks. Furthermore, the ATAM process has not only been helpful to understand the architecture better, but it should also act as a guide when there is a need to modify or evolve GSA. The guidance it gives is based on the revealed architectural decisions and their strengths and weaknesses.

A problem faced when using ATAM is that it produces an analysis table (e.g. Table 4.6) per scenario. As the number of scenarios grows, it becomes difficult to conceptualise the whole architecture as a single artefact based on all the tables generated. For instance, if we wanted to look up the quality attributes affected by an architectural decision we would have to examine all the tables to find that out. One possible workaround is to consolidate all the disparate tables into one single entity that can reveal this faster. A view was created for the architecture called the Architecture Reactive View (ARV) shown in Figure 4.13 (BinSubaih & Maddock, 2006). From this view the architect gets the whole picture about the architecture as it describes the interaction between three elements: quality attributes, architectural decisions, and ATAM output. First, the architect can find out which architectural decisions affect which quality attributes. Second, he can also find which ATAM output affects which architectural decisions. Finally, he can see how ATAM output affects the quality attributes.

This chapter has evaluated GSA and revealed how the architectural decisions have affected the quality attributes. The next chapter will use GSA to develop a serious game to evaluate the scalability of the architecture.

Architectural Decisions

ATAM output (risks, trade-offs, sensitivities, nonrisks)	AD1	AD2	AD3	AD4	AD5	AD6	AD7	AD8	P	M	PE
R1	√								√		
R2	√					√			√	√	
R3		√	√						√	√	
R4			√				√		√	√	
R5								√		√	
T1	√								+	+	-
T2		√							+		-
T3			√						+	+	-
T4				√					+	+	-
T5					√					-	+
S1		√				√			√	√	√
S2		√				√			√	√	√
S3								√		√	
S4				√			√			√	√
N1	√								√		
N2		√							√		√
N3			√						√	√	√
N4						√			√	√	
N5	√									√	
N6					√					√	√
	√	√	√	√		√			P	M	PE
	√		√	√	√	√	√	√	M	Quality Attributes	
	√	√	√	√	√				PE		

Figure 4.13: Architecture Reactive View (ARV) consolidating the disparate tables generated for each scenario into a single artefact. P, M, and PE refer to portability, modifiability and performance respectively.

5. Serious Games and Learning

5.1 Introduction

Crawford (Crawford, 1984) states that games are the most ancient and natural vehicle for learning. McLuhan said that "[a]nyone who makes a distinction between games and learning doesn't know the first thing about either" (Becker, 2006a). If this is true it is ironic that one of the difficulties stated for the lack of use of games in education or training (see section 5.2 for the distinction between the two) is because of the difficulty in getting acceptance for their use. One possible explanation is because we have been brought up to believe in school as a vehicle for education and possibly that has lessened our belief in the natural link between playing and learning. Another possible explanation is the perception associated with having fun which is often associated with ridicule and frivolity which leads to games being perceived as antithetical to learning (Becker, 2006b). Another perception is that video games are shallow and often violent indulgences (Stokes, 2005). These perceptions have been contested by many (Prensky, 2001; Gee, 2003), and the widespread cultural acceptance of computer games indicates that these perceptions are changing. A further possible explanation is that the people at the decision making level are possibly of the generation who have not grown up with the technology of computer games (Prensky calls them the digital immigrants (Prensky, 2001)). It is like there is a barrier (school system, digital immigrants, etc) between the learner and the natural way of learning. This barrier is starting to be questioned with the growing use of serious games. The early evidence emerging from empirical studies illustrates the power of games as educational tools. This has brought the focus back to a natural way of learning (i.e. learning by doing) albeit through the computer as mediator.

Section 5.2 provides evidence from different domains that have demonstrated the effectiveness of serious games. Section 5.3 examines the traits of serious games that make them effective for educational purposes, such as being motivating, engaging, and entertaining, all which are often associated with active learning. This section also details the impact computer games have had on today's learners. It also describes the reasons why serious games have a better chance than two closely related fields which have tried to make the journey back to a natural way of learning: edutainment (education through entertainment) and Virtual Reality (VR). It has been stressed that it is necessary to start any research on the use of games for educational purposes by critiquing these legacies so as to be able to build a better educational foundation for computer games (Egenfeldt-Nielsen, 2005). Section 5.4 describes the challenges facing the development and utilization of serious games. One of these challenges, which this thesis investigates, is the implication of using game engines to develop serious games (see chapters 2, 3, and 4). Chapter 6 will describe the development of a serious game for training traffic accident investigators in the Dubai police force – some of the following sections will comment on the needs for

this game. Section 5.5 provides background on the theories that have been shown to explain learning in serious games.

5.2 Evidence of the Effectiveness of Serious Games

Although there is no single definition agreed upon for serious games the consensus is that they are games that can be used for purposes other than entertainment, such as education, training, advertising, or politics (Abt, 1970; Michael & Chen, 2005; Susi et al., 2007; Zyda, 2005; Narayanasamy et al., 2006). The term serious games is sometimes viewed as nothing more than resurrected edutainment. However, Michael and Chen (Michael & Chen, 2005) argue that the targeted audience differs. Edutainment primarily targets preschool and young children which form a subset of the wider audience (all types of education and at all ages) targeted by serious games.

What distinguishes serious games from entertaining computer games is that serious games add pedagogy to the three main elements of computer games: story, art, and software (Zyda, 2005). Zyda describes pedagogy as any activity that educates or instructs and the challenge lies in making it subordinate to the game story. The addition of pedagogy has made two changes to the main characteristics of computer games (Susi et al., 2007). The first change is that in serious games it is more important to provide task fidelity (i.e. accurate representation of the problem that needs to be solved) than to provide the rich experience which computer games prefer. The second change is that in serious games the focus is on delivering learning objectives while computer games are focused on delivering fun. The focus of this thesis is on the role of serious games in facilitating learning. According to the Compact Oxford English Dictionary, "to learn" is to "acquire knowledge of or skill in (something) through study or experience or by being taught". Learning can be acquired through education or training. The difference between education and training, according to LeGrand and Freeman (LeGrand & Freedman, 1988), is that education "refers to the processes used … to produce knowledge and highly generalizable skills needed to reason and solve problems." and training refers to the processes used "to produce skills to accomplish a specific, practical goal." They add that education answers the "why" question whereas training answers the "how" question. Serious games have been used in education and training across a wide variety of domains, for which they have illustrated their learning effectiveness (see Table 5.1). The power of serious games stems from the fact that they build on the power of computer games which in turn build on the power of games. Each of these three mediums, discussed in each of the following subsections, has been shown to be effective at transferring learning across a wide skills range (see Figure 5.1).

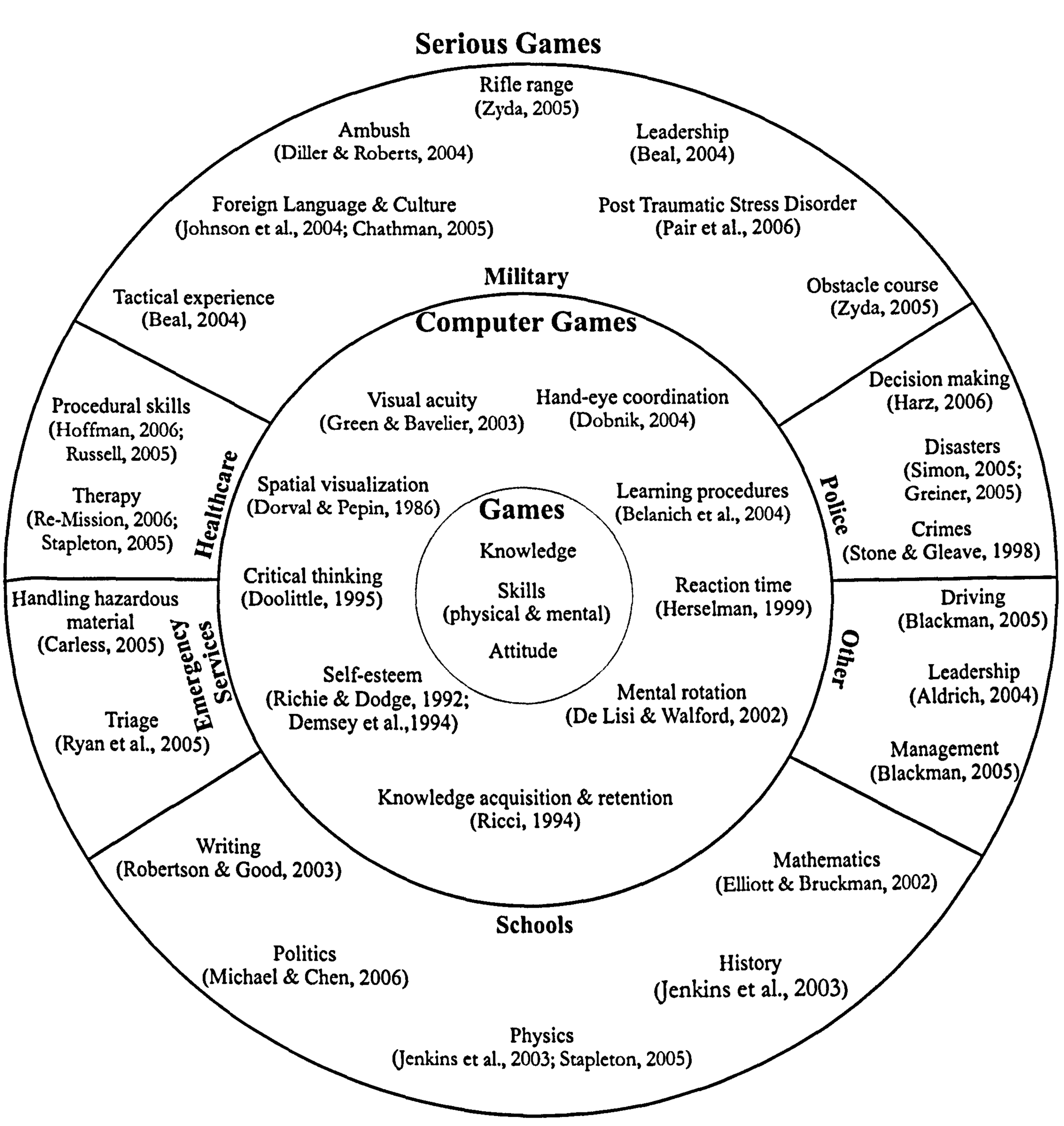


Figure 5.1: The learning effectiveness of serious games builds on the power of computer games which builds on the power of games.

5.2.1 Games

A game is defined by the Compact Oxford English Dictionary as "an activity engaged in for amusement". Despite amusement being the main focus of a game, Crawford (Crawford, 1984) argues that the fundamental reason why games are played is to learn. He adds that this is the case despite learning not being a conscious drive and in spite of it becoming a secondary objective to other objectives

such as: fantasy, challenge, and socialising. The ability of games to transfer learning is well acknowledged across Bloom's three learning domains: cognitive (ideas, opinions, and thoughts), affective (emotions, attitude, attention, and awareness), and psychomotor (motor skills and physical abilities) (Gunter et al., 2006; Abt, 1970).

5.2.2 Computer Games

Computer games have been found to be effective at enhancing performance across a wide range of skills even when computer games were not specifically built to do that. An example is hand-eye coordination. A study conducted at the Beth Israel Medical Centre showed that laparoscopic surgeons who played video games for 3 hours a week made about 37% fewer mistakes and managed to perform the task 27% faster than those who did not play video games (Dobnik, 2004). Another study at the University of Rochester looked at the effects of computer games on perceptual and motor skills (Green & Bavelier, 2003). The study found visual acuity[1] to be significantly higher among first-person shooter (FPS) players, and suggested using 10 hours of gaming as training to improve visual acuity. The Army Research Institute found FPS games to be best suited for learning procedures and recalling experiential details (Belanich et al., 2004). Their study showed that procedural information was retained at higher rates than factual information and graphic images and spoken text were recalled more accurately than printed text. Many more studies have been cited in the literature. For instance, Mitchell and Savill-Smith (Mitchell & Savill-Smith, 2004) cite studies which show the positive effects on psychomotor skills, analytical and spatial skills, strategic thinking and insight, and many more areas (Pillay et al., 1999; Kirriemuir, 2002; Ko, 2002; Green & Bavelier, 2003). Rosser et al. (Rosser et al., 2007) cite studies that have shown the positive effects of video games on eye-hand coordination tasks, neuropsychological tests and better reaction time, spatial visualization, mental rotation, and visual attention (Griffith et al., 1983; Yuji, 1996; De Lisi & Wolford, 2002; Dorval & Pepin, 1986).

5.2.3 Serious Games

The use of serious games dates back to the 1980s when Battlezone was used for military training. Another game representing a major step forward in the history of serious games, according to (Stone, 2005), is 'The Colony', a first person space survival game created in 1988. However the interest in serious games has only lately been accelerated by the increased interest shown by the U.S. Department of Defence (DOD) in video games technology (Zyda & Sheehan, 1997; Keller-McNulty et al., 2006), and also initiatives with more than a military focus such as the Serious Games Initiative (www.seriousgames.org), International Simulation & Gaming Association (www.isaga.info), North American Simulation and Gaming Association (www.nasaga.org), The Education Arcade (www.educationarcade.org), Game Research (www.game-research.com), and the UK Serious Games

[1] Visual acuity is an important skill to help focus on relevant information in chaotic environments.

Alliance (www.seriousgamesalliance.org). These initiatives have widened the learning spectrum that needs to be addressed (see Figure 5.1).

In the military domain, the use of serious games has reached a point where the domain is described as a "true believer" (Prensky, 2001). Therefore it is no surprise that most of the serious games are found in this domain and also most of the investment. The skills trained on include rifle range and obstacles courses (Zyda, 2005), and leadership and tactical experience (Beal, 2004). The healthcare domain has also experienced the benefit of serious games. Here, the rapid growth has reached a point where a 'games for health' conference is held annually. The usage of serious games in this domain varies from therapy (Re-Mission, 2006; Stapleton, 2005) to training procedural skills (Hoffman, 2006; Russell, 2005). The education domain has also reported the benefits of using serious games in teaching physics (Jenkins et al., 2003; Stapleton, 2005), mathematics (Elliott & Bruckman, 2002), and history (Jenkins et al., 2003).

The domain that is still lagging behind all of these is the police domain. Very few examples of the use of serious games appear to exist in this domain. Most of the examples found use video-based simulations and there is a lack of empirical study, as was shown by a report conducted by Bennell and Jones (Bennell & Jones, 2003). Despite an exhaustive search and two decades of video-based simulations, the report found that the documentation of their effectiveness was scarce. The report only managed to find four studies that used simulations for police training: Boyd (1992), Helsen and Starkes (1999), Scharr (2001), and Justice and Safety Centre (2002). Boyd reported the effectiveness of using simulations for training range shooting. Helsen and Starkes reported the effectiveness of simulations that used pop-up targets to improve complex decision-making skills for shooting precision. Participants also showed superiority in visual fixations which are crucial to identify suspects and assess potential weapon possession. Scharr's study demonstrated the ability of simulations to increase mental preparedness, perceived ability to resolve violent incidents, and better appreciation of effective communication skills. The Justice and Safety Centre study illustrated the training effectiveness by measuring: accuracy (number of shots fired, number of shots hitting the target, etc), tactics (identification of suspects, use of cover, etc), judgement (appropriate use of force), and safety (proper indexing of trigger, keeping weapon operational, etc). The results showed that shooting accuracy increased, and the effective use of cover also increased. Regarding the judgement to use force the results showed marked improvement. Finally, with regards to safety, the training decreased the tendency of the participants to point their guns outside the line of fire.

The above four examples from the police domain relied on video-based simulations which Aldrich criticised for being too costly (Aldrich, 2004). Bennell and Jones' report cited the work of Seymour et al. (Seymour et al., 1994) who also raised the prohibitive cost and time required. In the police domain, in general, funding is considerably less than in other domains such as the military and therefore there is a need to reduce the development cost. The other problem with video-based simulations is their inability to compete with the modding ability of computer games. Modding is one of the attributes that is very

desirable because it provides a training infrastructure where users can create modifications of the game to share experiences. Aldrich also mentioned the problems with freedom of movement, difficulty in extrapolating rules from videos, and difficulty in making small changes without re-shooting the scene.

Looking at the range of skills trained on in other domains it is noticeable that some of these skills can be used across domains. The police domain, which this thesis focuses on, can use a number of serious games form other domains to train on relevant skills such as shooting accuracy (military), dealing with hazardous material and performing first aid (military), and leadership (military). For example America's Army has training courses resembling Hogan's Alley[2] which can be used to train police officers and has been shown in the military domain (although anecdotally) to be effective for passing courses. Furthermore commercial games such as SWAT 4 (Terdiman, 2006) can be used for training. In this game, a player can be a team leader of non-player characters (NPCs) or join teams of human players. The game as it stands has a number of scenarios that can be used as they are and the ability to modify the game potentially makes it a very good platform for special weapons and tactics (SWAT) training (Lambie, 2006). There are other environments for training as well such as OLIVE (Simon, 2005), Incident Commander (Greiner, 2005), and Angel Five (Harz, 2006). However, there is a lack of empirical study about their effectiveness in the police domain.

Table 5.1: Examples of serious games and their learning effectiveness (the list contains only games that had evidence of learning effectiveness).

Domain	Serious Game	Description
Military	America's Army (Zyda, 2005; Harz, 2006)	A popular serious game is America's Army which was built with the primary aim of recruitment. It is considered to be the most successful serious game to date. *Learning effectiveness:* anecdotal evidence showed that it succeeded in helping new army recruits to pass rifle ranges and obstacle courses.
	Ambush! (Diller et al., 2005)	Ambush! enables squads to experience and respond to ambush situations using 3D simulations. *Learning effectiveness:* the 18 subjects that used Ambush! in this study felt positive (6.72 out of 7) about its effectiveness for tactics, techniques, and procedure training.
	Tactical Language Training System (TLTS) (Johnson et al., 2004; Chathman, 2005)	The objective of TLTS is to help learners acquire communication skills in foreign languages and cultures. *Learning effectiveness:* an evaluation with seven college-age subjects reported that the game was fun and interesting and they were generally confident that with practice they would be able to master the game.
	Full Spectrum Command (FSC) (Beal, 2004)	The game simulates a Captain commanding a light Infantry company offensive operation in an urban environment. *Learning effectiveness:* the findings from 54 officers tested on the game showed that playing FSC provided tactical experiences that had potential training value.
	Virtual Iraq (Pair et al., 2006)	Virtual environments were created to treat patients suffering from post traumatic stress disorder (PTSD). *Learning effectiveness:* the initial trials created environments resembling scenes from the Iraq war. These trials involved two patients and provided anecdotal evidence to show that the environment helped to cognitively reframe their experience in a positive way and also to reduce their nightmares.
	Microsoft Flight Simulator (Herz & Macedonia, 2002)	The game has been described as the most successful use of commercial games for training. In the US Navy, all student pilots and undergraduates receive a customized version of the software. *Learning effectiveness:* a study conducted by the US Navy showed that students who use the game during early flight training

[2] The trainee is required to make a split-second decision to shoot or don't shoot in an environment which resembles urban setting where targets pop-up representing "good guys" or "bad guys" (Bennell & Jones, 2003).

		receive higher scores that those who do not.
	Dismounted Infantry Virtual Environment (DIVE) (Stone, 2005)	A multiplayer environment for urban combat training. *Learning effectiveness:* early results showed a high level of engagement and that it provided valuable after action review.
Healthcare	Re-Mission (Re-Mission, 2006)	The game is developed by a non-profit organization called HopeLab with the aim to produce "an innovative solution to improve the health and quality of life of young people with chronic illness". *Learning effectiveness:* a trial test on 375 cancer patients showed that patients who played the game exhibited an increase in the quality of life, knowledge about cancer, and ability to manage the side effects.
	Self-management of diabetic children (Stokes, 2005)	The game is developed for diabetic children to help patients improve their self-management skills. The goal of the game is to have the player keep their character's diabetes under control by monitoring blood sugar, providing insulin and managing food intake. *Learning effectiveness:* the result reported was a 77% decrease in hospitalization rates for youths given a copy of the game.
	VR Phobias (Stapleton, 2005)	Used simulations and off-the-shelf and modified games to treat various forms of phobias such as: fear of driving, fear of the dark, fear of spiders, fear of heights, fear of snakes, claustrophobia and agoraphobia. *Learning effectiveness:* they were used as part of the clinical interviews and the patients were asked while playing what they were thinking in reference to their phobia. The findings reported "a high success rate (92%) in terms of treatment of varying phobias with few (4.5%) dropping out from therapy".
Educational	Supercharged! (Jenkins et al., 2003; Stapleton, 2005)	This is a physics game designed by MIT to teach students about electromagnetism. The objective of the game is for players to navigate their spacecraft through a 3D world to reach a goal. They can place charges within the environment to help direct their spacecraft. *Learning effectiveness:* the findings from the tests conducted showed that the game managed on average to help students who played the game to score 20% better than students who did not play.
	Civilization III (Sandford & Williamson, 2005)	This is a historical game where the player has to rule a stone-age tribe to guide them to progress. *Learning effectiveness:* the findings showed that students moved away from simple 'one cause = one effect' to more complex strategies which follow "a pattern of problem identification, causal interpretations, brainstorming solutions, implementing these solutions, examining results, and repeating their interventions"
	AquaMOOSE 3D (Elliott & Bruckman, 2002)	This is a mathematical game developed to teach students about parametric equations. Students use mathematics to construct graphical forms and challenges. *Learning effectiveness:* The game was evaluated on 105 high school students and the results showed that students found the aesthetic qualities of the environment motivating. However they reported problems with navigation.
	Dimenxian[3]	This is used to teach students algebra. *Learning effectiveness:* the game reported that a case study conducted found that students enjoyed playing the game and it also helped in improving their scores.

5.3 Why Use Serious Games for Learning?

Serious games provide a platform for active learning. The contrast between active and passive learning is an issue that is discussed widely. Passive learning is regarded as suffering from principally relying on a single sensory channel (hearing) and being delivered in a manner that assumes the perceptual and intellectual uniformity of learners (Foreman, 2003). Foreman summed up the deficiencies with the typical structure of large lectures (sometimes he refers to them as stables) into five main points. The first

[3] http://tabuladigita.com/ugroups.php?s2=3&s3=0 (accessed 12/1/2007).

point raised is that the ideal learning situation must be customized to the very specific needs of the learners but in the case of lectures it is a one-size-fits-all approach which ignores the individual's learning style. The second deficiency is the lack of immediate feedback. The third is that it fails to allow active discovery and the development of new kinds of comprehension. The fourth is the lack of motivation which undermines engagement. The final deficiency is linked to its failure to ensure that the concepts and procedures are committed to long-term memory which makes them available thereafter for the analysis and interpretation of real-world experiences. Aldrich (Aldrich, 2002) cites boredom as a problem with traditional classrooms in which the ability of learners to process lecture material after 30 minutes is suspect. Becker (Becker, 2006c) cites Bruner who explained that "what the school imposes often fails to enlist the natural energies that sustain spontaneous learning."

Active participation is one of the inherent strengths of computer games. However outside the games domain, and specifically in the traffic accident investigation field in the police domain, active learning is not very easy to facilitate due to the time, cost, and safety implications. A study conducted by the National Teaching Laboratory Institute (Magennis & Farrell, 2005) reported that students who learn by doing have an average retention rate of 75% compared to an average retention rate of 5% for those who learn from lectures. Another study puts retention rate at: 90% from simultaneously seeing, hearing, and doing, 80% from doing, 40% from seeing, and 20% from hearing (Joyce, 2005). The strengths of games (or simulations) that complement active learning are (Thalheimer, 2004): aligning contexts (i.e. matching learning contexts to on-the-job performance contexts), retrieval practice (i.e. recalling information from memory), feedback (i.e. correcting misconceptions), repetition (i.e. multiple scenarios covering the same learning points), and spacing (i.e. arranging repetitions apart in time). For aligning contexts Thalheimer cites a number of psychologists who found that learners would retrieve more from memory (improvement ranges from 10% to 55%) if they were placed in the same context in which the learning occurred. The other complementing factor mentioned by Thalheimer is the retrieval practice which was found to help improve learning by an amount ranging from 30% to 100%. It was also found that providing feedback improves learning by an amount ranging from 15% to 50%, repetition improved learning by an amount ranging from 30% to 110% or more, and spacing improved learning by an amount ranging from 5% to 40%.

Section 5.3.1 describes the importance of motivation and engagement in computer games and argues for the need to add fun to serious games despite fun being perceived with ridicule and frivolity. Finally section 5.3.2 describes how cultural inclusion of computer games has affected learners' characteristics.

5.3.1 Motivation and Engagement

Motivation and engagement are at the heart of computer games. Motivation is the reason behind someone's actions or behaviour, and engagement is to attract someone's interest or attention[4]. In the

[4] Defined by the Compact Oxford English Dictionary.

healthcare domain, Watters et al. showed that children who had completed a game called 'Bronkie the Bronchiasaurus', to teach children about asthma and aid them to learn more about managing it, understood the impact of their decisions and made better choices than those who did not have access to the game (Watters et al., 2006). Huang et al. (Huang et al., 2006) cite two studies (Sankaran & Bui, 2001; Sachs, 2001) that have shown there is a positive relationship between motivation and performance. More studies are cited by Beedle and Wright (Beedle & Wright, 2006).

Engagement has been argued as one of the reasons why the military has turned to serious games (Susi et al., 2007). Furthermore highly engaging games, argued Becker (Becker, 2005), will also be found to meet Thomas Malone's intrinsic motivation[5] criteria for engaging learners. These are challenge, curiosity, fantasy, and control. Challenge relies on engaging the player's self-esteem through meaningful goals (Habgood et al., 2005). In a survey conducted by the Entertainment Software Association (ESA) in 2001 about the four main reasons for gameplay, challenge came second to fun with 72% of participants saying games are challenging (Kirriemuir & McFarlane, 2004). The curiosity to see what happens encourages the player to keep playing (Kirriemuir & McFarlane, 2004). Habgood et al. also added that cognitive curiosity is aroused when players discover their knowledge is incomplete and inconsistent. Fantasy allows a player to evoke "mental images of physical or social situations not actually present" (Malone & Lepper, 1987). Finally control gives the player a sense of empowerment and self-determination (Habgood et al., 2005).

Prensky (Prensky, 2001) identified a number of characteristics that make games engaging, as shown in Figure 5.2. These characteristics cover what is needed for a game to succeed in motivating and engaging players to the level where they become oblivious to distractions (Habgood et al., 2005; Kirriemuir & McFarlane, 2004). Another advantage of having a motivating and engaging game is that it widens its appeal to players with different learning styles (Becker, 2006c). Becker examined how games appeal to five styles: Howard Gardner's theory of multiple intelligences, the Keirsey temperament sorter, Felder's index of learning styles, Kolb's learning styles, and The Gregorc system of learning. The examination listed examples of how games are very successful at capturing the desired audiences without being deliberately designed with learning styles in mind (e.g. SIMs, Half-Life II, Halo, and Grand Theft Auto). Becker's work highlighted the need for more research to find out if games actually influence the players' learning styles.

The benefit of learning styles in general has been questioned in a report which looked at 71 different learning styles (Coffield et al., 2004). The report found that the value of matching teaching and learning styles is highly questionable. The question about how to accommodate learning styles in eLearning was put to a panel during The eLearning Producer Conference and Expo 2005 (Brandon, 2005). Ruth Clark argued that the effort spent on learning styles is the biggest waste of resources in eLearning because the

[5] Intrinsic motivation pushes people to engage in an activity for its own sake rather than it being imposed by external factors which is extrinsic motivation (Denis & Jouvelot, 2005).

WHY GAMES ENGAGE US

Games have **interaction**. That gives us ***social groups***.
Games have **goals**. That gives us ***motivation***.
Games are a form of **fun**. That gives us ***enjoyment and pleasure***.
Games are a form of **play**. That gives us ***intense and passionate involvement***.
Games have **rules**. That gives us ***structure***.
Games are **interactive**. That gives us ***doing***.
Games have **outcomes and feedback**. That gives us ***learning***.
Games are **adaptive**. That gives us ***flow***.
Games have **win states**. That gives us ***ego gratification***
Games have **conflict/competition/challenge/opposition**. That gives us ***adrenaline***.
Games have **problem solving**. That sparks our ***creativity***.
Games have **representation and story**. That gives us ***emotion***.

Figure 5.2: Why games are engaging (from (Prensky, 2001)).

cognitive commonalities outweigh the differences and therefore the effort should be spent on that. The focus on the use of learning styles seems to be on how to use them to aid the design stage. There is also another possible role for them during the testing and evaluation phases of serious games. They could be used as a diagnostic tool to help identify the causes of success or failure based on the participants' styles.

Another important reason for using serious games is because they are fun and, despite sometimes being perceived as frivoulous, this remains important because it generates the energy needed to keep the learner engaged (Dieleman & Huisingh, 2006). The 2001 Entertainment Software Association (ESA) survey revealed that 87% of the most frequent computer and video game players said the number one reason for playing games is because they are fun (Kirriemuir & McFarlane, 2004). In another survey conducted by ESA in 2006, which looked at the top four reasons parents play video games with their children, fun came second with 75% (ESA, 2006). The first reason, with 79%, was because they were asked. To create a great entertaining game, Garneau (Garneau, 2001) lists fourteen forms of fun that can aid the design process such as: beauty (i.e. pleasing the senses through, for instance, graphics and sound), intellectual problem solving (i.e. finding solutions to problems), competition (i.e. showing one's superiority), discovery (i.e. exploring the unknown), and advancement and completion (i.e. progressing towards the ultimate goal of finishing a game). In serious games, however, fun must be treated with caution and the designer must work towards balancing the fun element against learning (Roussou, 2004).

5.3.2 Cultural Acceptance and its Impact on a Learner's Characteristics

There are other forces contributing to serious games being considered for education. One of these is the increase in the number of people playing video games (Becker, 2006b). This has led to an increase in the cultural inclusion and tolerance of video games. Figure 5.3 shows some of the other facts reported by ESA. Cultural inclusion is highlighted by 61% of the parents who believe that games are a positive part of their children's lives. In the UAE the estimated video games market size in 2005 was 15 million US dollars which was ahead of other sectors such as racquet sports and accessories (6 million) and behind sectors like amusement park/outdoor playground equipment (32 million) and fitness equipment (29 million) . For technology adoption overall, the number of mobile subscribers for 2005 was very high

(nearly 100%) and internet and broadband penetration for the same year reached 51% . These are positive indicators towards technology adoption in general and games tolerance in particular. We can speculate that the impact technology and games are having on digital natives elsewhere is also affecting the people in the UAE despite the cultural differences that might exist in comparison with the USA data cited in Figure 5.3.

- Frequent game purchaser age is 40 years.
- The average age of players is 33 years.
- 69% of American heads of household play computer or video games.
- The average length gamers have been playing games is 12 years.
- The time parents are present at the time games are purchased or rented is 89%.
- The parents who believe games are a positive part of their children's lives is 61%.
- The time children receive their parents' permission before purchasing or renting a game is 87%.

Figure 5.3: ESA 2006 (ESA, 2006).

This widespread use of games in particular, and technology in general, has impacted on the characteristics of the digital natives (or game generation). Aldrich (Aldrich, 2002) asserts that the digital natives demand engagement on "multiple levels simultaneously, in a fast-feedback, graphical, high stimulation, extremely immersive, and user-centric environment". The importance of engagement has also been stressed by Prensky (Prensky, 2001) who states that the difference between game design and curriculum design is in the focus – curriculum design focuses on content whereas game design focuses on engagement. Prensky also listed ten cognitive style differences (see Table 5.2) between the digital natives and the digital immigrants.

Table 5.2: 10 cognitive style changes listed by Prensky (Prensky, 2001).

Digital Natives	Vs.	Digital Immigrants
Twitch speed		Conventional speed
Parallel processing		Linear processing
Graphics first		Text first
Random access		Step-by-step
Connected		Standalone
Active		Passive
Play		Work
Payoff		Patience
Fantasy		Reality
Technology-as-friend		Technology-as-foe

This surge in game utilization is regarded as a disruptive technology since it (Lenoir, 2003): challenges the existing expertise and practice, requires new skill sets, and demands organizational change. The change in the characteristics of the digital natives is a testament to this disruption which has increased the pressure on the dominant educational delivery mechanism: the lecture. The disruption also affects the teacher by changing the learning from being teacher-centred to being player-centred (Stapleton, 2005). This changes the role of the teacher when using games from being an agent transmitting knowledge to becoming a promoter who enables learning (Chwif & Barretto, 2003). Table 5.3 gives a comparison between conventional teaching and game based teaching. This table presents the shift which brings a number of challenges to the design and use of serious games which will be discussed in the next section. Overall it is quite clear from the reasons discussed in this section why it is becoming more difficult to ignore the benefits of games in general and serious games in particular. In some domains it is becoming almost unthinkable not to use a simulation for training (i.e. aviation) and others say "We know

the technology works, we've proven it over and over again, and we just want to get on with using it." – Don Johnson, the Pentagon (Prensky, 2001).

Table 5.3: Rodrigo's comparison between conventional teaching and simulation based teaching (Chwif & Barretto, 2003).

Paradigm	Conventional	Simulation Games
Teacher's Role	Agent	Promoter
Student's Role	Receptive	Active
Contents	Predominantly Theoretical	Real
Motivation to Learn	Contents Sequence	Curiosity, desire to solve a problem

5.4 Challenges Facing Serious Games

The serious games field is still a relatively new one and is facing challenges which range from the selection of a suitable topic to the method of assessment used (see Figure 5.4). Many of these challenges require interdisciplinary approaches to address them appropriately which correspondingly requires collaboration between professionals from different disciplines (e.g. subject matter, game design, game development, and instructional design) and this is in itself has been described as an awkward problem (Stokes, 2005). This section describes the challenges facing the design and development of a serious game. This is illustrated by commenting on how these challenges will be addressed in the development of a serious game for traffic accident investigation (see chapter 6). The serious game aims to provide police officers with the experience of virtually investigating traffic accidents and also aims to address the current training issues facing the Dubai police force.

Section 5.4.1 discusses the need to address why a serious game is needed in the first place by identifying the suitability of the topic, the instructional problems, and how the serious game can help. Section 5.4.2 describes the challenge of making the learning an integral part of the game and how the integration success can be illustrated. Section 5.4.3 highlights the difficulties with incorporating assessment as part of the game. Section 5.4.4 presents the development options and the issues facing each option. Finally section 5.4.5 lists the challenges facing the field in general.

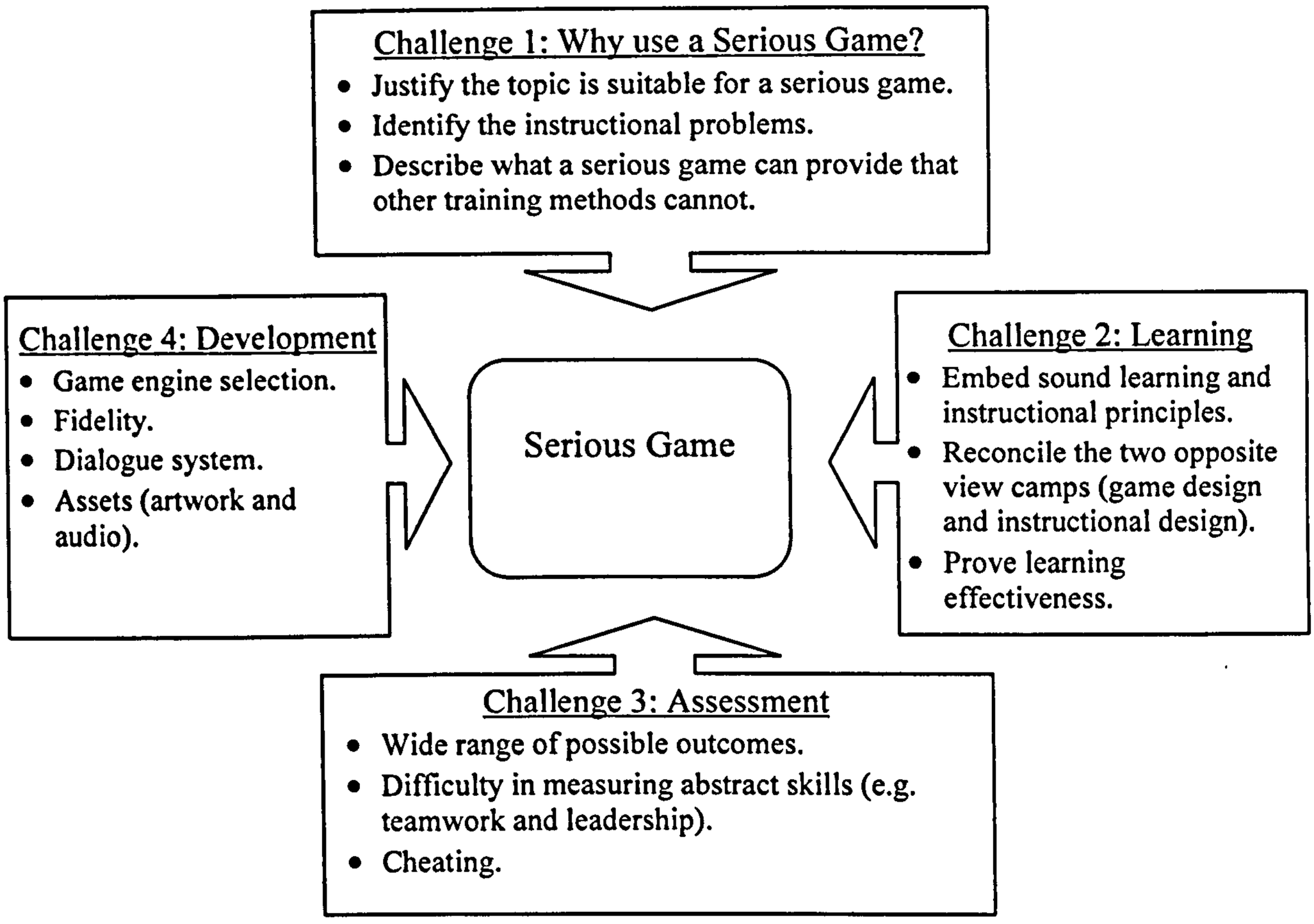


Figure 5.4: Challenges facing serious games.

5.4.1 The First Challenge: Why Use a Serious Game?

The first challenge in designing and developing any serious game is to justify its need by examining the suitability of the topic, by identifying the instructional problems, and by finding out why a serious game may be more effective than other training methods. With regards to the topic selection, Thaigi and Prensky agree on the possibility of using games to teach anything to anyone at any time (Nichani, 2001). However Prensky raised some concerns of its worthiness considering the time and cost involved and suggests the power of games should be reserved for material the learners do not want and even resist to learn because it is boring (e.g. policies) or complicated (e.g. complex software). The traffic accident investigation topic is not a boring topic and has different levels of complexity. The delivery mechanism adopted in the classroom can be classified as being boring but the field training in a real accident situation is certainly not. Serious games provide an ideal solution to this problem since the motivation and engagement are inherently present.

5.4.2 The Second Challenge: Learning

Once the instructional problems and learning objectives are identified the next challenge is to integrate them into the serious game in a way that goes beyond making the game a sugar-coating for educational purposes which was how edutainment was perceived (Kirriemuir & McFarlane, 2004). Egenfeldt-Nielsen (Egenfeldt-Nielsen, 2005) describes the problem as the lack of connection between the learning and the gameplay which very often limits the use of games as a reward for learning. He gives the example of

Math Blaster!, an educational game in which the player has to shoot down the balloon that represents the right answer and whoever pops all the balloons first wins. The problem with such an approach, he argues, is that it is based on the assumption that constant shooting of balloons will automatically lead to a conditioned response no matter the learning, context or previous experience. He argues this illustrates the disconnection that exists between the game (shooting balloons) and the learning (mathematics). What the game is doing here is providing extrinsic motivation (not really related to the game but consisting of arbitrary rewards) rather than intrinsic motivation (the feeling of mastery from completing a level). Becker (Becker, 2005) argues that this disconnected approach has led to a lack of respect for edutainment.

In the serious games field there is a general consensus about the need for building games based on sound learning and instructional principles (Mantovani, 2001; Psotka et al., 2004; Gunter et al., 2006; Mitchell & Savill-Smith, 2004; Akilli, 2006). Two issues need to be overcome for this to happen. The first issue is to prove the worthiness of instructional design models. Here, research has produced a number of accepted and well-tested models such as ADDIE (Molenda, 2003). The second issue, which is still at an early stage of research is how to match instructional design principles to game design principles. The serious games literature focuses on reporting the technical issues involved in development, or the findings of empirical studies, or a combination of both, or places the emphasis on the learning theories used for designing the serious game. The topic that does not seem to have received similar attention is a practical demonstration of how instructional design was used alongside game design in the development process. The reason why this is scarce could possibly be attributed to the separation between the two camps – game design and instructional design – as described by Becker (Becker, 2006c).

The first camp views game design principles as ones that are already employing sound principles and thus do not require instructional design principles. The second camp argues that despite the fact that games are already applying instructional principles, the "game designers must yield to the better-informed professional instructional designer" (Becker, 2006c). Prensky (Prensky, 2001) very often in his presentations and writings quotes a game designer who complains that when you introduce instructional designers to the development team, "the first thing they do is suck the fun out." It has been pointed out that this can be turned around to say that leaving instructional designers out sucks the pedagogy out of the game (Jerz, 2005). In a debate between Prensky (on the game designers' side) and Cannon-Bowers (on the instructional designers' side) during the Serious Games Summit DC 2005 (Jerz, 2005), Cannon-Bowers stressed that she did not care if her doctor had fun when learning and preferred that he trained on a solid system. Becker argues that the differences between the two camps must be reconciled before they can be combined to develop instructional games.

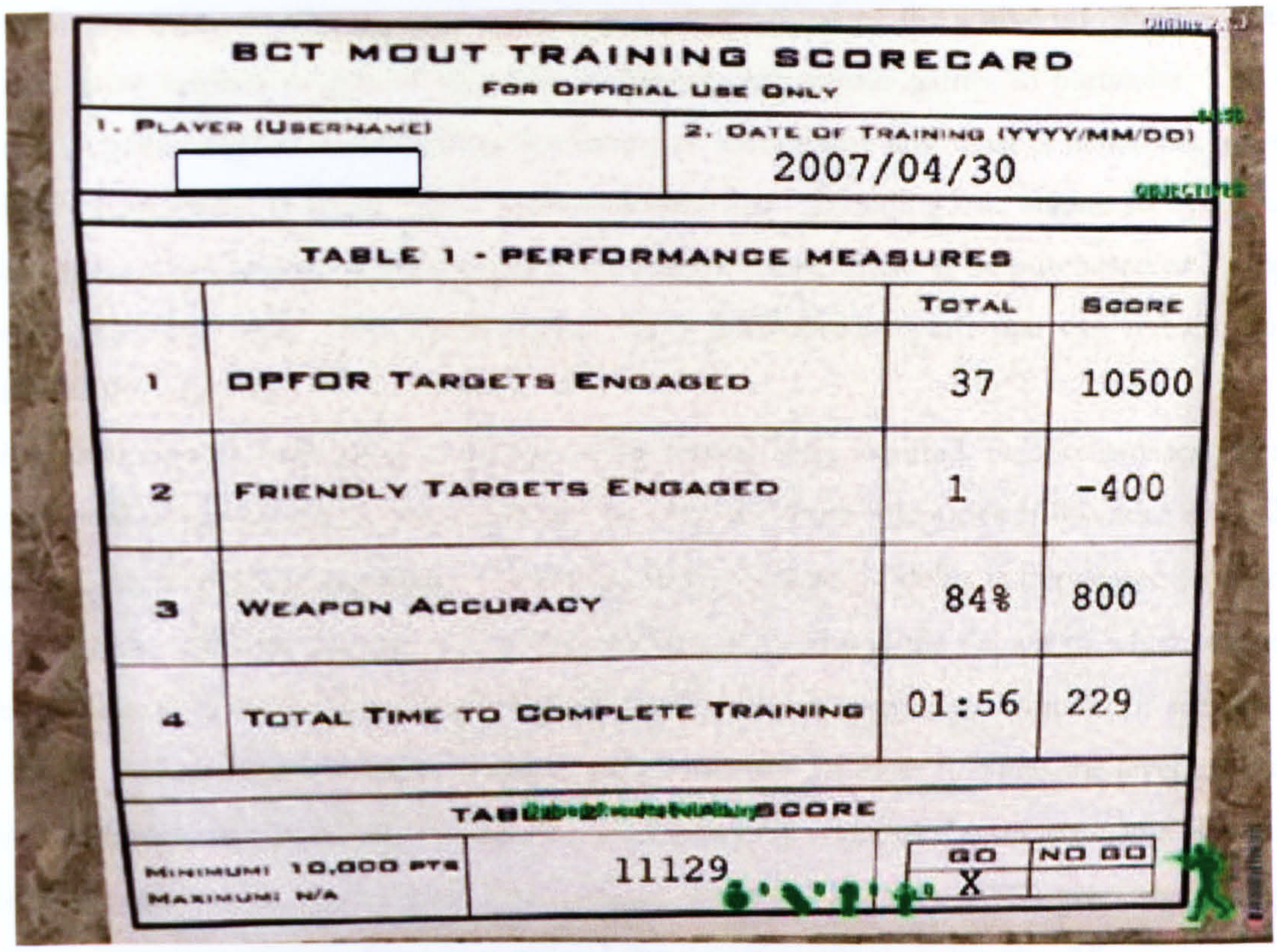
The literature shows that the reconciliation process is already underway to establish common ground between game design and instructional design. Gee (Gee, 2003) in his book "What Video Games Have

to Teach us About Learning and Literacy" has argued against those who say that video games are mindless exercises by suggesting that good video games have 36 learning principles built into them. Another proponent of video games as learning tools is Prensky (Prensky, 2001; Gee & Prensky, 2006). He identified 10 cognitive style changes (see Table 5.2) in the digital natives which challenges the current education and training methods and argues for alternatives. Aldrich (Aldrich, 2005) presented a model in which he split serious game design into three types of elements: game, simulation, and pedagogy. He argues that the careful use of all three produces an appropriate educational experience. This work could provide the common ground to aid the reconciliation between the two camps. In fact it has already started to produce instructional design models specifically developed for serious games, such as CRAFTE (Charsky, 2006) which made use of Aldrich's elements. Aldrich's elements will be used in section 6.4.2 to help with the instructional design of the serious game for traffic accident investigators (SGTAI)

5.4.3 The Third Challenge: Assessment

The assessment of learning in serious games presents another challenge that has to be addressed. The future growth of the serious games industry depends on it according to Kevin Corti of PIXELearning (Chen & Michael, 2005). Researchers have identified a number of assessment issues facing serious games. One of these issues arose because serious games rely less on memorization of facts and therefore traditional methods may not appropriately reflect the learning gained (Chen & Michael, 2005). The other issue concerns the wide range of possible solutions due to the open-ended nature of serious games which entail different levels of knowledge transfer (Iuppa & Borst, 2007; Chen & Michael, 2005). Iuppa & Borst also described the issue of measuring the improvements of abstract skills such as teamwork and leadership. Chen & Michael added the problem of identifying what is cheating in the context of serious games. To meet these issues three main types of assessments have been used by serious games developers (Chen & Michael, 2005): completion assessment, in-process assessment, and teacher evaluation.

Completion assessment is the simplest form of assessment and measures whether or not the learner managed to complete the serious game. Figure 5.5 shows an assessment example from America's Army after completing a Shoothouse training mission in which a player has to clear a house by making split-second friend or foe decisions. The problem with completion assessment, argue Chen and Michael, is that it falls short as it cannot distinguish between whether the learner learned the material in the game or just learned to beat the game. In-process assessment relies on logging and tracking the learner's actions. Teachers can then use this to aid the assessment process. Chapter 6 will present a graphical approach of presenting data such as navigational patterns. The third assessment type is teacher evaluation which relies on a combination of completion assessment and in-process assessment.



BCT MOUT TRAINING SCORECARD

FOR OFFICIAL USE ONLY

1. PLAYER (USERNAME)	2. DATE OF TRAINING (YYYY/MM/DD)
	2007/04/30

TABLE 1 - PERFORMANCE MEASURES

		TOTAL	SCORE
1	OPFOR TARGETS ENGAGED	37	10500
2	FRIENDLY TARGETS ENGAGED	1	-400
3	WEAPON ACCURACY	84%	800
4	TOTAL TIME TO COMPLETE TRAINING	01:56	229

TABLE 2 SCORE

MINIMUM: 10,000 PTS MAXIMUM: N/A	11129	GO	NO GO
		X	

Figure 5.5: A sample of America's Army score sheet after completing the Shoothouse training mission.

5.4.4 The Fourth Challenge: Development

After the serious game is designed the next challenge becomes the development. The two main development options are either to build from scratch or to reuse game engines. The advantage of the first option is that the team has full control over the source code. The disadvantage, and what has been argued as being a prohibiting factor, is cost (Gaudiosi, 2005). Cost can also be an issue with the second option. However, the wide range of game engines available (see section 2.3 for a survey) means the cost range varies from free to six figures plus royalties. The other factors pushing towards the second option are: the graphics capability, the availability of scripting, the small learning curve, and the AI, physics and networking. Section 2.4 presents a survey showing how important these factors are to projects that use game engines. Customers are also favouring the option of companies using existing game engines rather than building one from scratch for their projects, as was highlighted during the 3Up/3Down panel session in the Serious Games Summit 2007. Roger Smith from the US Army Program Executive Office for Simulation, Training, and Instrumentation (PEO STRI) said that if two companies bid for a project, the one that says it is going to build the serious game from scratch will lose.

After deciding on the development path of using a game engine, which this thesis focuses on, the next critical question is deciding on which engine to choose. The 3D engines database on DevMaster.net lists

282 engines[6]. The variety of engines available and the lack of transparency make it difficult to choose the right engine (Carey, 2007). Furthermore, the decision is complicated by the scarce information available that compares game engines in general and their suitability for serious games in particular. The other major issue with game engines is concerning the future of that engine and what is becoming known as "the RenderWare problem" (Carless, 2007). Carless argues that "[b]uilding one engine so deeply into a development process can be risky if the company in question is ripe to either be purchased or its support potentially dwindle over time." This thesis argues for a development path that can reduce the very critical nature of this decision.

The other development challenge is to decide on the fidelity level required. Fidelity is described as the level to which serious games aim to emulate reality and has different categories (Alexander et al., 2005): physical fidelity, functional fidelity, and psychological fidelity. Physical fidelity is the degree to which the game looks, sounds, and feels like the real world. Functional fidelity is the degree to which the serious game behaves like a real situation. Psychological fidelity is the degree to which the serious game replicates the psychological factors experienced in a real situation. In these fidelities the level experienced can be either low, high, or somewhere in between. In low fidelity some of the serious game elements are abstracted from reality to be emphasised. For instance, Prensky (Prensky, 2001) gives the example of teaching someone to set time and temperature for baking under different altitudes. Here it is acceptable to lower the fidelity by emphasising the time and temperature elements and removing all the irrelevant elements to this learning objective such as choosing the ingredients and creating the mix. In high fidelity the serious game tries to emulate reality as close as possible. Low fidelity serious games are good at teaching general principles and insights (Thiagarajan, 2001) and very successful for beginners as they reduce the amount of detail that might confuse the learner (Prensky, 2001). High fidelity serious games are very reliable at transfer of training and are suited for teaching step-by-step procedures (Thiagarajan & Thiagarajan, 1997). It has been argued that the level of fidelity required is situation specific and has no right answer (Prensky, 2004). For instance, Figure 5.6 shows a study which compared the effect of low fidelity and high fidelity characters on presence (Vinayagamoorthy et al., 2004). The study reported lower presence for characters with higher fidelity when they are placed in a virtual environment where there are repetitive textures (e.g. buildings, tiles, and billboards). The presence felt was increased when non-repetitive textures were used. However, overall, participants in environments with cartoon like characters reported higher presence level. Another study reported that high fidelity graphics managed to focus learners' attention initially but was less important for longer periods (Beal, 2004).

The last two challenges facing the development and related to fidelity are the dialogue system and the assets. The development of a dialogue system that is capable of producing something equalling natural language conversation is very difficult and the error rate remains a prohibiting factor for their use in pedagogical applications (Iuppa & Borst, 2007). Even in computer games the conversations are

---

[6] Accessed on 4/10/2007.

conservative and use linear approaches such as branching trees (Aldrich, 2004). The assets challenge is much easier to deal with due to the existence of many free and commercial sites for textures, models, audio, etc. However some of these still have to be modified to suit the scenario developed and the game engine's format.

5.4.5 Other Challenges

There are other challenges facing serious games in general which are not mainly design and development issues. During the Serious Games Summit 2006 a panel session was held to investigate what is wrong

(a) The cartoon form characters.

(b) The higher fidelity characters.

Figure 5.6: Characters showing two different levels of fidelity (Vinayagamoorthy et al., 2004).

with serious games (Terdiman, 2006). Ben Sawyer raised the problem of the domain's perception which is looked at as a failure and a joke because it has failed to produce a large library of finished games. Henry Kelly, president of the Federation of American Scientists, pointed out that the problem is with the direction the serious games is focused on which often targets government-funded institutions (e.g. schools or military). Kelly argued that government are often sceptical about projects with abstract goals. He added that the lack of easily measurable standards for success or growth makes it difficult for outsiders to judge if the projects work. Paul Gee, a professor of learning sciences at the University of Wisconsin-Madison, painted a more urgent case that needs a quick solution. Early evidence and managing expectations were attributed as problems that have hindered the VR field (Jerz, 2005; Stone, 2005). Others have warned that the current early evidence seems to present only small-scale studies (Sandford & Williamson, 2005). The 2006 session inspired another panel session during the Serious Games Summit 2007 called 3Up/3Down. In this session each panel member had to describe 3 positive

and 3 negative things about the serious games domain. Table 5.4 summarizes the points made, some of which enforce the points made in the previous subsections.

Table 5.4: The 3 positive and 3 negative points made in the 3Up/3Down session during the Serious Games Summit 2007.

Panellist	3 Up	3 Down
Richard Van Eck	• Being recognized as a domain on its own. • Growing acceptance. • Critical mass in K-12 with textbook publishers showing interest.	• Fractionalized voice, inconsistency, no new models. • What is needed is instructional fidelity not surface fidelity. • Problem with standardized tests.
Jesse Schell	• Academic interest explosion in the field. • Wii is having tremendous effect on showing games are for everyone. • Increased broadband penetration which is a viable delivery for serious games.	• Need to confront whether it works (i.e. more examples needed). • No clear guide of techniques of how to produce a serious game. • Gatekeepers do not believe in this technique.
Roger Smith	• Games are sometimes better than current teaching methods (e.g. Ambush!). • Military funding multiplayer game to be inserted into real command and control to plan a battle. • The cost of art assets is pushing towards maintaining an art repository.	• Limited licensing options. • IT security policy imposed on networks and desktop applications are hindering accessibility (i.e. ports are blocked). • When you have FPS hammer everything looks like 3D nail (i.e. not all problems require 3D solutions).
Doug Whatley	• Perception is changing and the field is starting to be seen as legitimate. • Games are becoming more acceptable. • Modeling and simulation is moving out from what it used to do to being used for training and operational purposes.	• Success changes everything, we have lost our courage (e.g. SCORM is good but was not designed for games). • Serious games need to be made sexy for new talent. • Serious games companies should be real companies.

5.5 Theoretical Basis for Learning in Serious Games

The aim of this section is to provide a theoretical background on how learning occurs while playing computer games. The reason for doing so is to have a better understanding of the issues that can undermine the integration of learning objectives into the game (see section 5.4.2) and to ensure that the issues that have undermined learning in previous generations are considered. The use of educational games is divided into three generations (see Figure 5.7). The figure highlights the learning theories that apply to each generation.

5.5.1 First Generation

The first generation started with edutainment which relied heavily on behaviourism theory. Behaviourism is based on a stimuli-response pattern for conditioning behaviour to become automatic. This was illustrated by the Math Blaster! game example described in section 5.4.2. This theory suffers from disconnection between the game and the learning. The cause of that is possibly related to the

fundamental problem with behaviourism which was identified in the early 1920s. The problem highlighted behaviourism's inability to explain the thought process behind behaviour and gave rise to cognitivism theory (Mergel, 1998). Cognitivism was utilized alongside constructivism in the second generation of educational games.

5.5.2 Second Generation

The second generation employed cognitivism in order to make the learner the centre of attention and it shows interest in the learning content, settings, and differences between learners. Dark and Winstead describe cognitivism as being "focused on how information is organized, structured, and conceptualized" (Dark & Winstead, 2005). It is primarily used in lectures for information transmission. In multimedia this theory is believed to have found that different modalities (text, pictures, sound, etc) provide better learning (Mayer, 2001). Egenfeldt-Nielsen cites the example of a project named Plato which aimed to use this theory to teach maths instead of relying on behaviourism theory. What it did was to replace abstract exercises such as 2+2 by something like "if you have 2 bananas and get 2 bananas more how many do you then have". This led to a significant positive effect on achievement and attitudes towards maths. The other example he gives is of a game called Rocky Boots (see Figure 5.8) which he argues managed to successfully integrate the learning content and the game. It was designed to teach basic maths and programming concepts. It allows the learner to connect different symbols (and, or, not, etc) to create digital logic circuits. The game won several awards. Despite this success cognitivism theory was criticised for its failure to integrate the affective (feelings and emotions) and social (socialization and

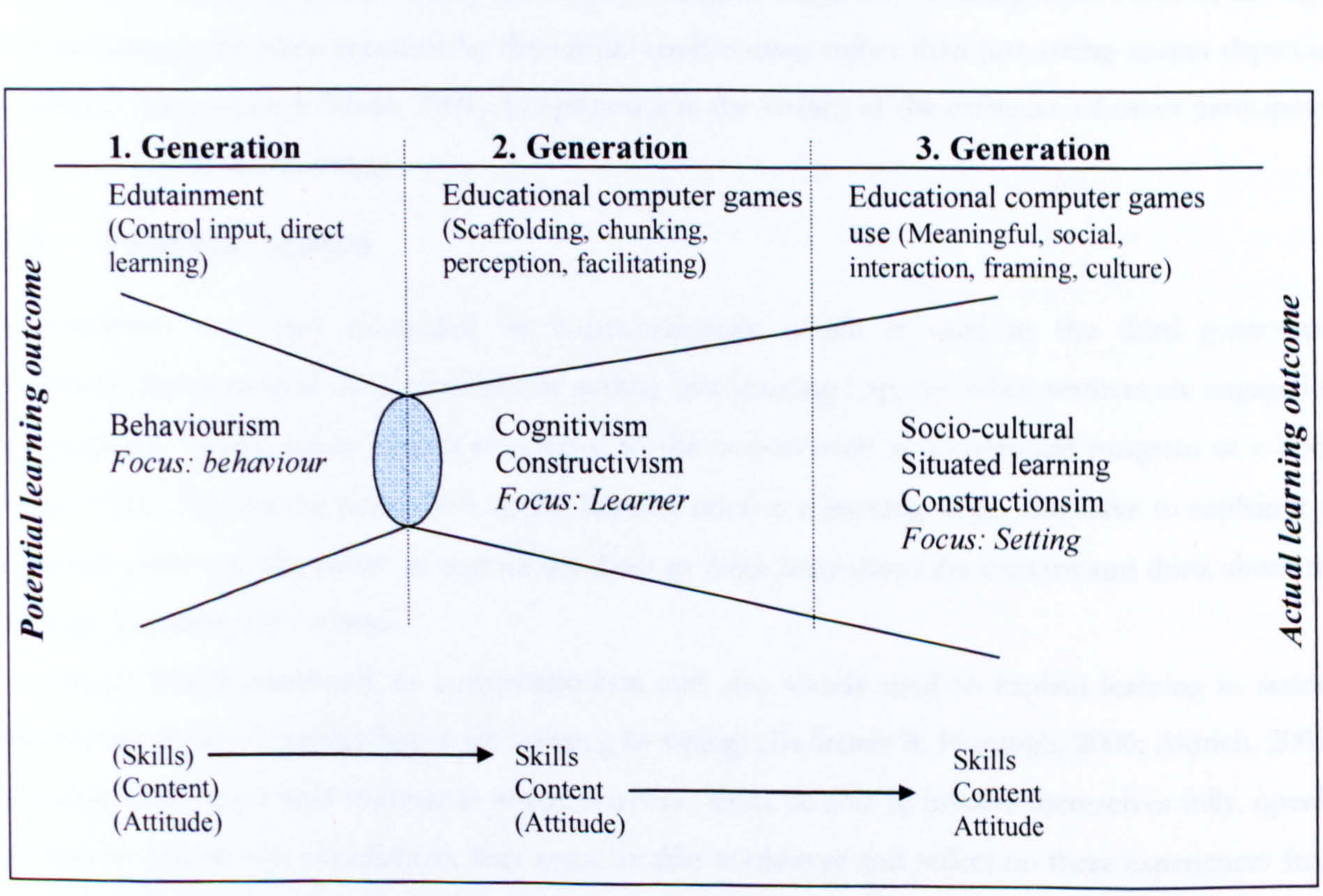


Figure 5.7: The three generations of educational games (Egenfeldt-Nielsen, 2005).

societies) domains with the cognitive (identifying and analyzing a problem, applying past learning, etc) domain, according to Dark and Winstead (Dark & Winstead, 2005).

The second theory used in the second generation is constructivism (or learning by making). It aims for the learner to construct knowledge rather than it being acquired as in cognitivism. Mantovani (Mantovani, 2001) explains that in constructivism learners learn best when building their own understanding of the content by interacting with it.

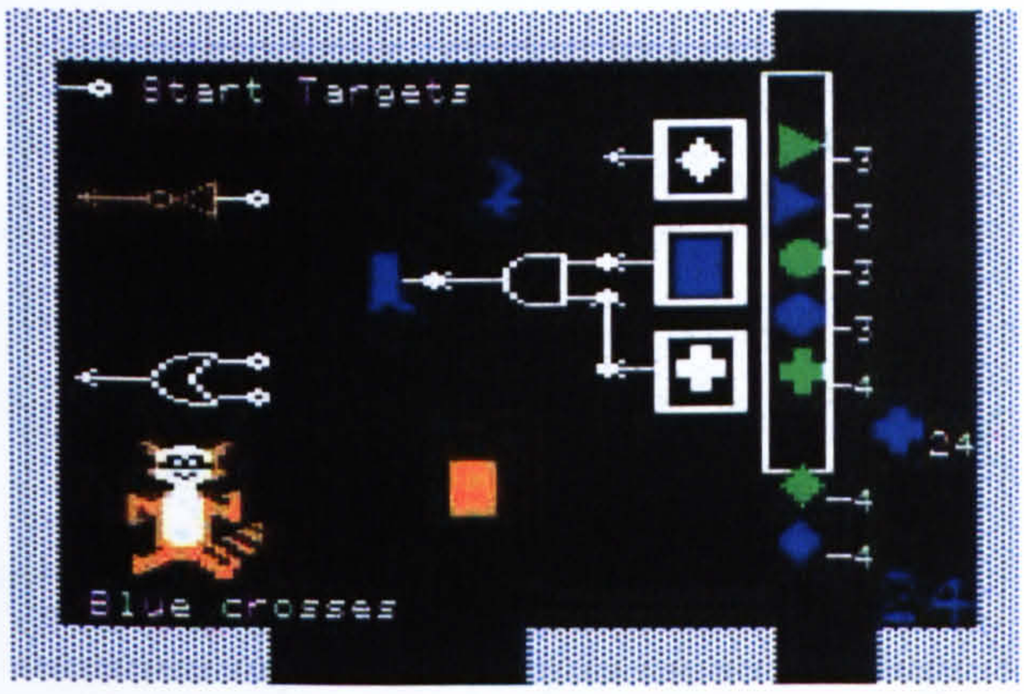


Figure 5.8: Rocky Boots (Egenfeldt-Nielsen, 2005).

The idea is that learners construct their own understanding of the world in the form of rules and mental models which they use to make sense of their experiences (Corti, 2006). The goal is to immerse the learner in a virtual world similar to the real world and allow learning to take place in a natural way. Immersion has received considerable attention in the virtual environment field and is defined as "a psychological state characterized by perceiving oneself to be enveloped by, included in, and interacting with an environment that provides a continuous stream of stimuli and experiences" (Witmer & Singer, 1998). The second phenomenon of interest to the virtual environment field, and based on placing a learner in a virtual environment, is presence. Presence is divided into two types: personal presence and co-presence. Personal presence "refers to the psychological sensation of 'being there', that is, having a sense of being in the place specified by the virtual environment rather than just seeing images depicting that place." (Casanueva & Blake, 2001). Co-presence is the feeling of the existence of other participants in the same virtual environment.

5.5.3 Third Generation

Constructivism was then succeeded by constructionism which is used in the third generation. Constructionism extended constructivism by adding that learning happens when learners are engaged in constructing a "public entity" which is external to themselves such as a computer program or a book (Papert, 1991). The crucial factor here is that learners reinforce learning when they have to explain it to others (i.e. make a public entity) as that forces them to think hard about the content and think about the best ways to convey it to others.

A theory that is connected to constructionism and also widely used to explain learning in serious games is experiential learning theory (or learning by doing) (Dieleman & Huisingh, 2006; Aldrich, 2005). Kolb describes experiential learning as where learners "must be able to involve themselves fully, openly, and without bias in new experiences; they must be able to observe and reflect on these experiences from many perspectives; they must be able to create concepts that integrate their observations into logically sound theories; and they must be able to use these theories to make decisions and solve problems"

(Feinstein et al., 2002). This type of learning adds doing to hearing and seeing. It focuses on concrete experience which is well-suited to computer games (Egenfeldt-Nielsen, 2005). Egenfeldt-Nielsen contrasts it to the lack of experience-based learning in a classroom when learning about history for example. This learning is usually based on reading or hearing about abstract concepts which the experiential learning challenges. He compares this setting to the experiences gained from playing Grand Theft Auto 3 and SimCity 4 where the learner is "part of a living, breathing, simulated universe with very concrete self-sustaining experiences".

Experiential learning consists of four stages according to Kolb's learning cycle, as shown in Figure 5.9. In the first stage the learner is involved in an activity – concrete experience (CE). In the second stage he reflects on the experience – reflective observation (RO). In the third stage he uses the observations to formulate a 'theory' based on his own concrete experience to see if it can work – abstract conceptualization (AC). In the final stage the learner uses the theories formulated for future decision-making and problem solving – active experimentation (AE). Armitage (Armitage, 1993) has also pointed to the suitability of this model to explain learning in simulations. She also highlighted three issues with this model. The first issue is that it does not encompass any external input which commonly happens prior to the CE phase in the form of lectures. The second issue is regarding Kolb's emphasis on full involvement which she argues would require the student to actually implement decisions in a real place. The third issue is the lack of explicit account of feelings (i.e. affective domain). To remedy these issues she proposed combining Kolb's theory with Binsted's whole cycle learning theory. Binsted's theory is based on encompassing external inputs as part of the learning process and takes into account that feelings are part of the learning process. It is comprised of three processes: reception of input, discovery, and reflection. The input can be in the form of external sources (lectures, books, etc) or learners themselves indicating what they think or know already. The discovery process requires the learner to take some action in the outer world and receive some feedback. The reflection happens in the learner's inner world and involves the learner's existing skills, knowledge, and feelings. Despite these issues Kolb's theory remains very useful in aiding the understanding of how the learner enters the cycle and how that correlates to the serious game training session. This theory will be used to aid the design of SGTAI in section 6.4.1.

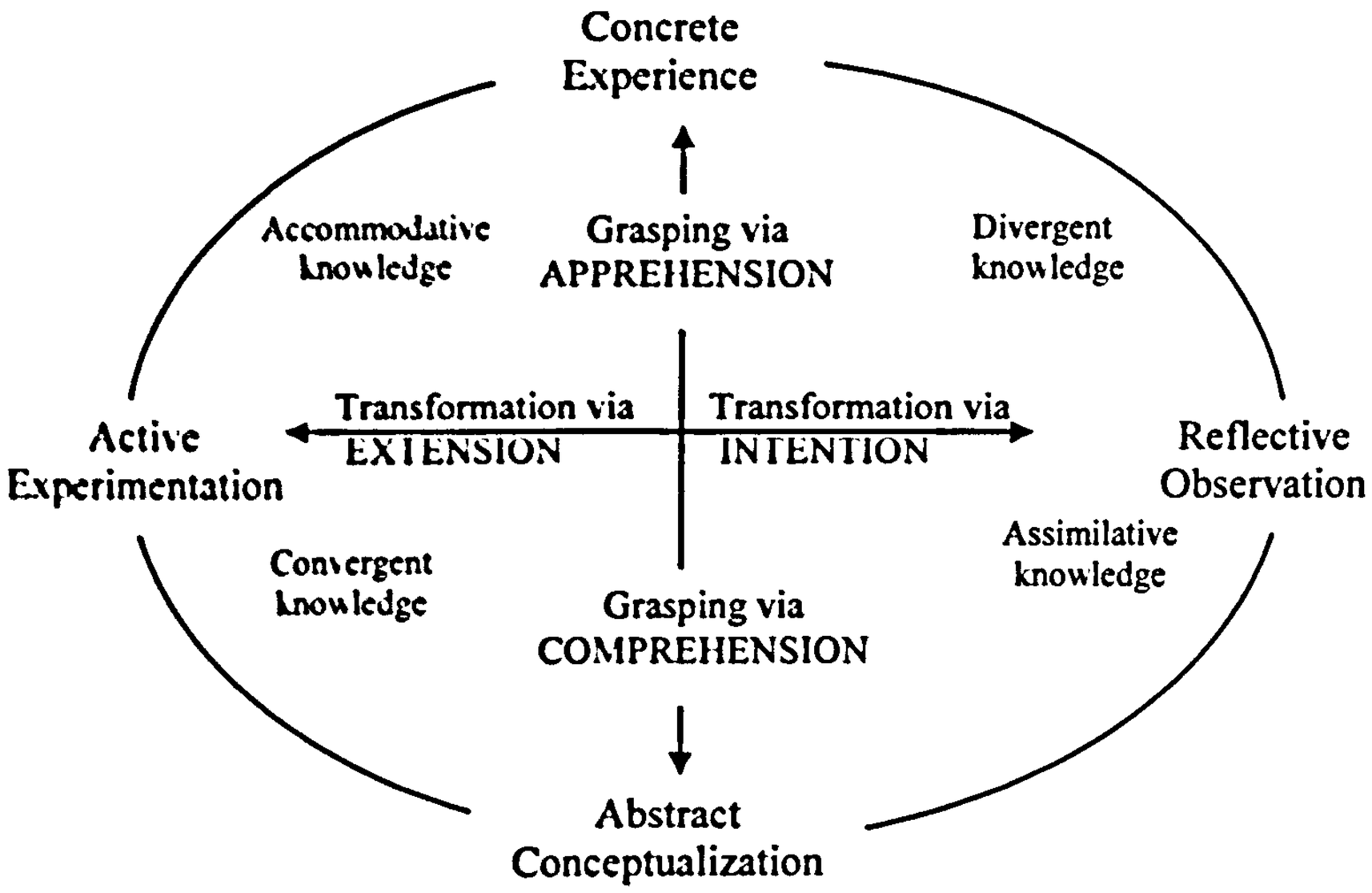


Figure 5.9: Kolb's experiential learning cycle (Chee, 2002).

The second learning theory used in the third generation of educational games is situated learning theory. This theory "suggests that learning is contextual, embedded in a social and physical environment"[7]. The emphasis is on providing a setting that is close to reality which Ogle (Ogle, 2002) argues is suited to virtual environments. Egenfeldt-Nielsen (Egenfeldt-Nielsen, 2005) states that situated learning has criticized the assumption of information transfer. It argues that information only becomes part of everyday life when learners see it grounded in context.

The third theory described as part of the third generation of educational games is socio-cultural theory. This theory emphasises the need to scrutinize the tools used as mediating activities. Egenfeldt-Nielsen gives the examples of reading, writing, or hearing which use language as a tool. In a similar manner games can be used as tools to mediate learning through discussion, reflection, and analysis in a social context. He points to Civilization III which was used by Squire to teach history. The game was found to facilitate discussion, reflection, facts and analysis facilitated by the surrounding classroom culture and the student's identity.

Another recent learning theory which can be described as falling into the third generation category is full-cycle learning proposed by Aldrich (Aldrich, 2002). This theory suggests that learning starts at an initial understanding then moves to testing that knowledge and finally ends at building a more refined understanding. The cycle comprises of four steps: understand a system, have a goal, receive feedback, and update knowledge. Aldrich also produced three types of elements which can be combined to aid the

[7] http://en.wikipedia.org/wiki/Situated_learning (accessed 30/1/2007).

process of designing a serious game: simulation (e.g. discovery, practice, and feedback), game (e.g. exaggerations, competition, and challenge), and pedagogical (e.g. learning objectives, scoring, and debriefing) (Aldrich, 2005). These elements will be used in section 6.4.2 to design SGTAI.

5.5.4 Which Theory to Choose?

The question that still requires more research is how to decide on which theory to base a serious game on. Mantovani (Mantovani, 2001) argues that the current theories are not yet capable of providing a reliable basis upon which to build up practice (i.e. design, assessment, or teaching), because their concern was to offer conceptual frameworks of learning rather than provide concrete guidelines to inform practice. Additionally there is no evidence to point to one particular learning theory to be sufficient on its own in explaining why learning occurs in serious games. Egenfeldt-Nielsen (Egenfeldt-Nielsen, 2005) points this out in his three generations of educational games. He asserts that "each generation is carried forward to the next, but de-emphasised." Becker (Becker, 2006c) also points out the difficulty and argues that retrofitting one learning theory onto a successful game is possible but it is entirely a different problem to go the other way. She cites the example of the movie industry that has been around for 100 years but has no sure-fire formula to create blockbusters. Despite the problems with theories in enabling the construction of suitable teaching methods, they remain a helpful tool to gain insight into the more practical methods to explain how the different elements of the serious game are going to influence the learner. Chapter 6 will describe how the learning theories are combined with instructional theories to create the learning foundations for SGTAI.

5.6 Summary

This chapter has provided some examples demonstrating the effectiveness of serious games for training across a number of domains on a variety of skill sets. The reason why serious games have been found to be effective is due to the inherent traits upon which they are built (i.e. games and computer games). Motivation represents one of these traits and the challenge when using it in serious games is how to provide intrinsic rather than extrinsic motivation. The motivation must be meaningful to the learning objective and must provide the feeling of mastery rather than some arbitrary reward. Motivation is also one of the reasons why games are engaging (see Figure 5.2). The other factors that make games engaging include interactivity, conflict, competition, problem solving, rules, outcome and feedback, and fun. All of these factors have to be balanced during the development process against learning, time, cost, and technology to produce an effective serious game.

Although there is no sure-fire formula on how to combine all the above factors to produce an effective serious game (Becker, 2006c), some argue for the use of instructional design alongside game design to help ensure the learning objectives are an integral part of the game to improve its effectiveness (see section 5.4.2). The way instructional design is combined with game design in the development of

SGTAI will be described in chapter 6. The chapter will also demonstrate how the assessment challenge (see section 5.4.3) is addressed.

6. A Serious Game for Traffic Accident Investigators

6.1 Introduction

This chapter demonstrates the use of the game space architecture (GSA), described in chapter 3, in developing a serious game for traffic accident investigators (SGTAI) in the Dubai police force. There are three aims in this chapter. The first aim is to develop a serious game (SGTAI) to address the issues facing the current training practices employed by the Dubai police force. The second aim is to statistically measure the learning effectiveness of SGTAI. The third aim is to put the game space architecture into practice[1] by developing a real world application which is much more complex than the sample games described in chapters 3 and 4.

Section 6.2 discusses the field study conducted in 2004. This was an understanding of the problems of the traffic investigation training used by the Dubai police force before building SGTAI. The section also highlights the applicability of a serious game for addressing the current training practices. Section 6.3 presents a walkthrough of the virtual experience provided by SGTAI. This walkthrough is used throughout the remaining sections to highlight different issues such as design, development, and assessment. The building of SGTAI using GSA is detailed in Appendix C. Section 6.4 describes how the learning objectives are embedded in SGTAI with the aid of instructional design to ensure that the game is not merely used to make learning fun. Section 6.5 details an experiment conducted in 2006 for 56 police officers from the Dubai police force. The experiment used SGTAI to train police officers. The two questions the experiment aimed to answer were: how effective is a serious game in training traffic accident investigators, and does the effect differ between novice and experienced investigators. The section also describes the experiment design, the measures used to evaluate performance, and the performance findings. Section 6.6 discusses the effectiveness of SGTAI in achieving its learning objectives, the factors that have contributed to that, the limitations, and the implications for policy makers, educators, and researchers.

6.2 Training Traffic Accidents Investigators in the Dubai Police Force

All new police officers recruited by the Dubai police force go through the same training process, which consists of lectures and on-the-job training (designed for their specific rank category e.g. sergeant and lieutenant). The training is administered by Dubai Police Academy which also accepts recruits from other Gulf Cooperation Council (GCC) countries, the Republic of Yemen, and Palestine. The duration of the training varies from 6 months to 4 years, based on the rank category. During this training, and

[1] The serious game could have been developed using a typical game development approach but here it is developed using GSA to demonstrate the architecture's scalability.

also after graduation, local recruits are assigned to police stations and departments in which they receive further on-the-job training. Section 6.2.1 describes the problems faced by the current training practices and section 6.2.2 presents the learning objectives for SGTAI.

6.2.1 Problems with the Current Training Practices

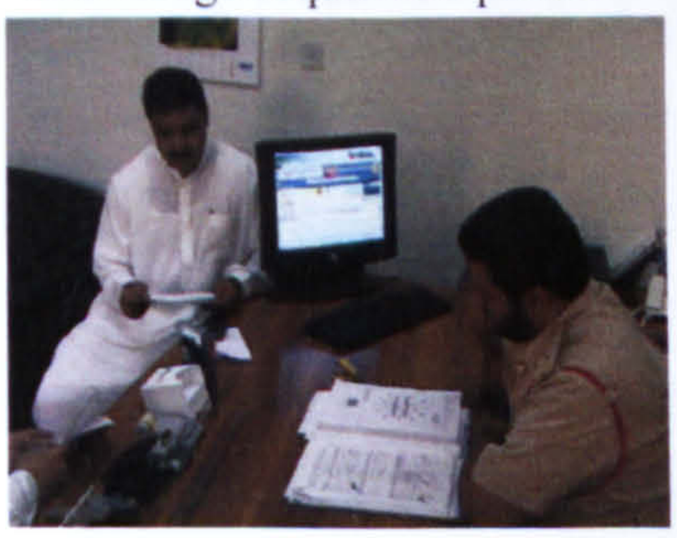
To better understand the traffic investigation field, the author conducted a field study (see BinSubaih et al., 2005a). The field study was divided into two phases: knowledge acquisition (see Figure 6.1) and preliminary experimentation (see Figure 6.2). The main objectives of the knowledge acquisition phase were to better understand the investigation process and to identify the instructional problems facing current training in the Dubai police force, which consists mainly of lectures and on-the-job training. The objective of the preliminary experiment was to examine the suitability of using serious games to teach

Interviews, role playing, and elicitation sessions.

(a) Kickoff meeting describing knowledge acquisition phases

(b) Interview

(c) Discussion



(d) Discussing assessment criteria with quality managers

(e) Elicitation session

(f) Role-playing session

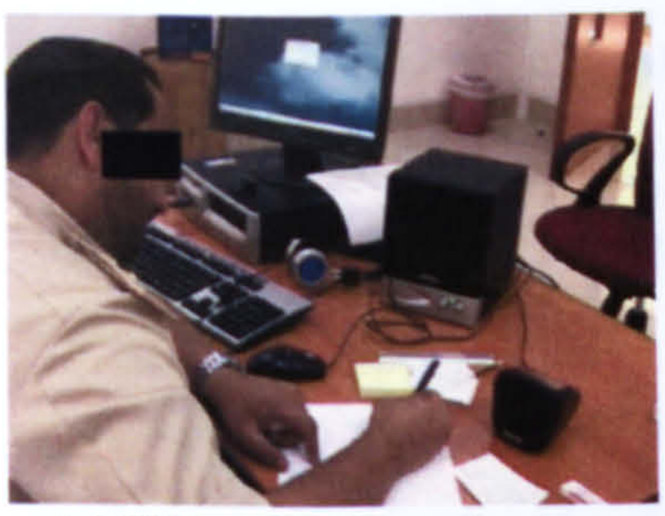
Travelling to accidents.

(g) Inspecting damages to a motorbike

(h) Photographing damages

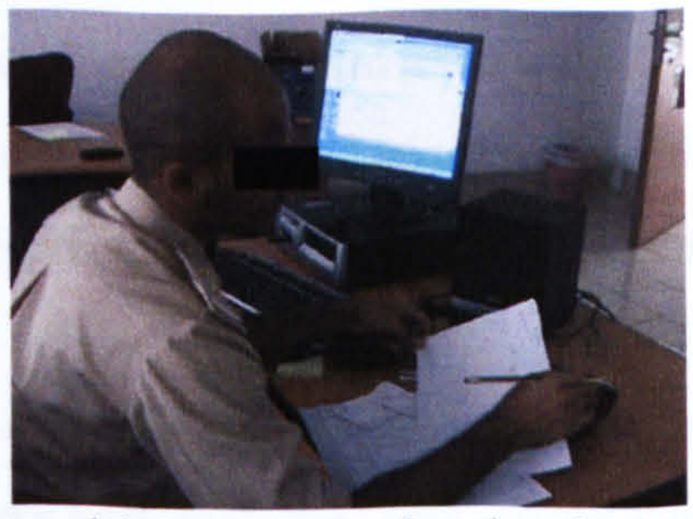
(i) Collaboration between the police and tow truck operators

Figure 6.1: The knowledge acquisition part of the field study.



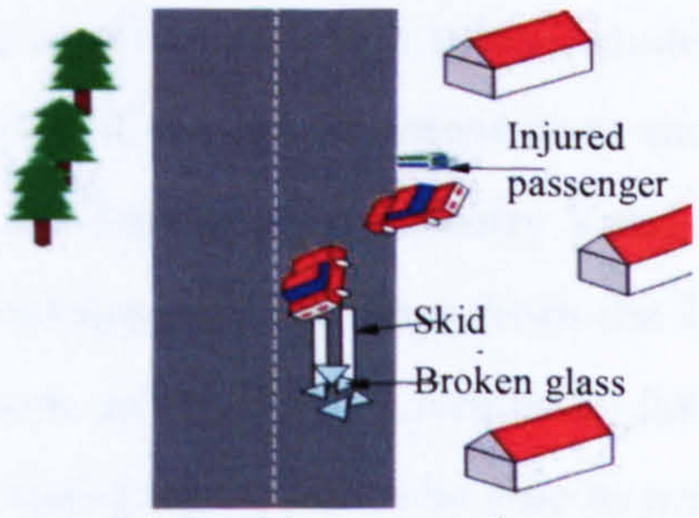
(a) Filling the report for the virtual experiment

(b) Examining the scene in a tabletop experiment

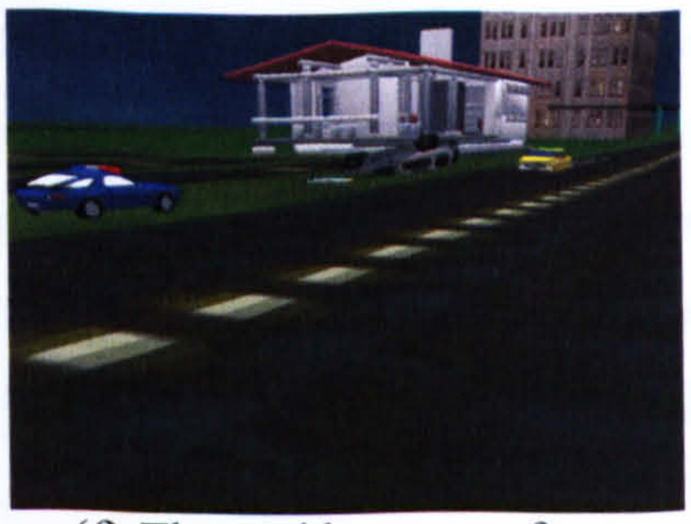


(c) Making the drawing for the virtual experiment

(d) The injured location



(e) Accident scene layout



(f) The accident scene from wider angle

Figure 6.2: The preliminary experimentation part of the field study.

traffic investigation. The preliminary experiment compared the use of a multiplayer serious game against the use of tabletop training. The results helped in identifying what SGTAI must focus on and in getting a feel for the acceptance of such technology in the Dubai police force.

The field study confirmed the well-documented problems with using only lectures for teaching practical skills (Zhou & Reed, 2005; Foreman, 2003; Aldrich, 2002). For example, lectures lack interaction and engagement which are important (Sankaran & Bui, 2001; Sachs, 2001). Furthermore the time allocated for the traffic investigation course was not sufficient to cover all the various accident types. The field study also found that the on-the-job training suffered from issues such as impracticality, varying levels of exposure, and lack of uniform assessment. The real world environment hinders repeatability and exploration, two elements which are very important in any training environment. In particular, in a real traffic accident, exploration is very difficult to achieve. Issues such as the possibility of a traffic jam meaning that a road has to be cleared as soon as possible, the bewildering heat during the day in Dubai, and the intolerance of the people around and of those involved in the traffic accident that want to get away, make it very difficult for an investigator to do his job. For a novice, the pressure of such problems, including the fear of embarrassment in front of the public and his colleagues, induces him to avoid exploration. In addition, in the real world it is impossible to reproduce a situation in an identical manner so that the same tasks can be practiced again and again.

A further problem is the varying level of accident exposure that the various officers are subject to. Accident types and frequency differ from one area to another. Due to the fact that a new investigator is assigned within a jurisdiction to a particular police station and a particular patrol unit during on-the-job

training, he might only be exposed to a limited range and number of accidents. The third issue is the lack of uniform assessment. The experienced investigator uses his own subjective judgment to decide whether a new recruit has completed the training and the lack of objective metrics can undermine this judgment.

Serious games have been found to be effective at addressing the above issues because they provide an active form of learning (see section 5.3). Serious games are motivating and engaging, and entertaining (see section 5.3.1). These represent some of the characteristics required to grab a learner's attention (Prensky, 2001; Becker, 2005). In a study of 105 high school students who played AquaMOOSE 3D, it was found that the aesthetic qualities of the environment were motivating (Elliott & Bruckman, 2002). In addition, the early results from the Dismounted Infantry Virtual Environment (DIVE) (Stone, 2005) showed the game to be engaging and the initial findings from the Tactical Iraqi showed that the subjects found the game to be fun (Johnson et al., 2004; Chathman, 2005). In Ambush! (Diller et al., 2005; Roberts et al., 2006) learners appreciated the ability to be able to practice without fear.

Whilst it is important for a serious game to be engaging, motivating, and entertaining, these should not distract from the main role which is to deliver the learning objectives set for it (Zyda, 2005; Susi et al., 2007). The learning objectives for SGTAI are described in the next section and the way the design process ensures that learning is integrated in the gameplay is described in section 6.4.

6.2.2 Learning Objectives

The learning objectives for SGTAI are to provide an environment that resembles a real traffic accident investigation which is practical in nature and varies in complexity. Figure 6.3 shows a typical accident investigation path. In practice, the investigation path varies between investigators. For example, some like to start by questioning the drivers (Figure 6.3g) before examining the evidence (Figure 6.3h) whereas others like to start with the evidence.

The learners targeted by SGTAI are the officers in charge during an investigation. In the Dubai police force each patrol vehicle has two personnel, the officer in charge and his assistant who is often also the driver. SGTAI aims to provide the investigator with a single player first-person shooter (FPS) type environment. The FPS genre represents the closest match to the real-life training environment which should help improve learning (Thalheimer, 2004). The decision to use a single player rather than a multiplayer environment was made because the environment was required to be used inside and outside a classroom setting. A single player environment is more suitable as it avoids the need to provide actors. In a multiplayer version, actors are used to play the roles of drivers, operators, paramedics and other personnel to allow the investigator to experience dealing with the people involved when investigating an accident. In a single player environment the interaction with people is limited to stock replies to standard questions.

In traffic investigation training courses, there are three domains of learning: knowledge, skills, and attitudes. The focus of SGTAI is on the knowledge and skills domains. The aim is to provide investigators with the experience of going through and completing the tasks of the five investigative phases: receiving the incident call, arriving at the accident scene, conducting the initial investigation, finalizing the data collection, and completing the accident file. It has been stressed that the effectiveness of a traffic accident investigator is dependent on two factors: training and experience (Baker & Fricke, 1986). To provide a training environment and an environment for gaining experience, SGTAI provides participants with: (i) a practical and safe environment to practice away from real accidents constraints, (ii) a modifiable environment to cater for the different accident types, (iii) an environment which provides a uniform assessment, (iv) a single player environment which forces the investigator to carry out all the investigative phases – this avoids the issue where an experienced investigator takes over, as was the case for a novice investigator who did not complete a single drawing during his six months on the job because an experienced investigator always assumed the role, (v) an environment that records the interactions and which can be used to share the experiences of an aging workforce, and (vi) an environment that facilitates social interaction outside the game. Figure 6.4 shows a typical virtual traffic investigation experience provided by SGTAI, which tries to match the real experience shown in Figure 6.3.

6.3 A Walkthrough of SGTAI

This section presents a walkthrough of SGTAI based on the five investigative phases described in the previous section. The walkthrough shows how SGTAI tries to provide the virtual experience shown in Figure 6.4 which mimics the real experience shown in Figure 6.3. In addition, the walkthrough will be used in section 6.4 to help explain how learning occurs in SGTAI and how the learning was embedded using instructional principles. The process of building SGTAI is described in Appendix C.

6.3.1 Phase 1: Receiving the Incident Call

When the investigator is ready, he receives a dispatch call via audio[2] and text. Upon accepting the dispatch call the investigator is placed in his car and is automatically driven to the accident scene. While on the way to the accident scene the investigator can utilize the time to start the initial inquiry. To do that the investigator clicks on the radio icon on the right menu (see Figure 6.5) to launch the operator interface (see Figure 6.7a). The operator interface allows the investigator to inquire about the accident situation (e.g. seriousness, vehicles involved, and witnesses), request assistance (e.g. ambulance, fire engine, and other patrol cars), and debrief the operator about the progress of the investigation.

[2] A text-to-speech synthesizer is used for English and recorded messages are used for Arabic. This is the same for the communication that occurs between the investigator and others during the investigation.

(a) Receive the accident call.

(b) Travel to the accident.

(c) Contact the operation room for further assistance.

(d) Secure the scene by parking the patrol vehicle at an appropriate place.

(e) Secure the scene using traffic cones.

(f) Attend injured.

(g) Identify and question people at the scene.

(h) Search for clues.

(i) Mark clues.

(j) Take photographs.

(k) Take measurements.

(l) Draw the accident scene.

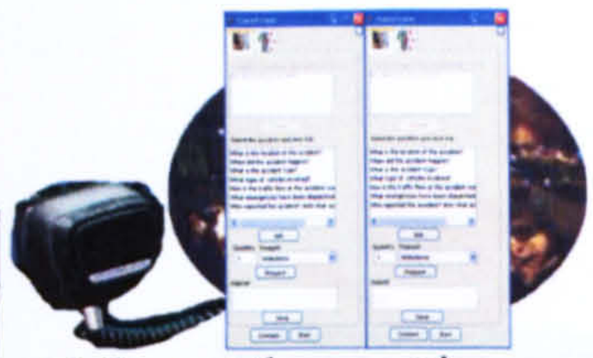
(m) Collaborate with paramedics.

(n) Collaborate with tow truck operator.

Figure 6.3: A typical traffic accident investigation experience (phase 1: a-c; phase 2: d-f, the identification of drivers: part of g, and h; phase 3 & 4: questioning the drivers: part of g, j, k, m, and n; phase 5: l).

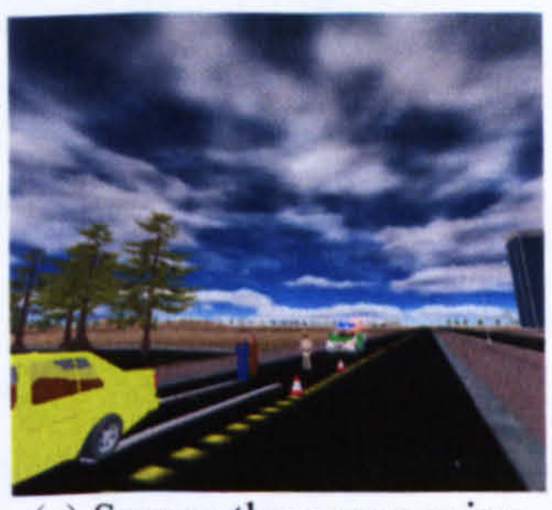
(a) Receive the accident call.

(b) Travel to the accident.



(c) Contact the operation room for further assistance.

(d) Secure the scene by parking the patrol vehicle at an appropriate place.

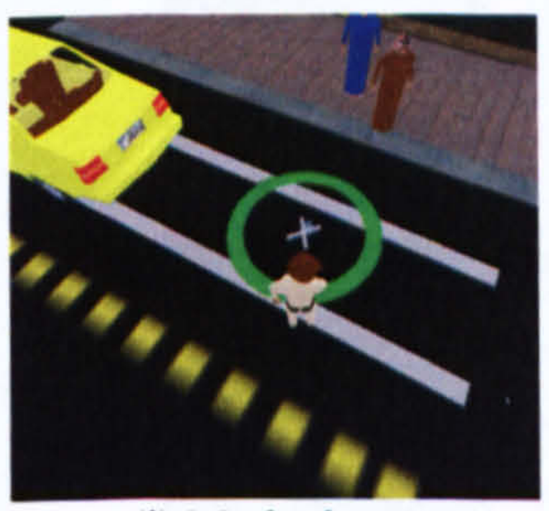


(e) Secure the scene using traffic cones.

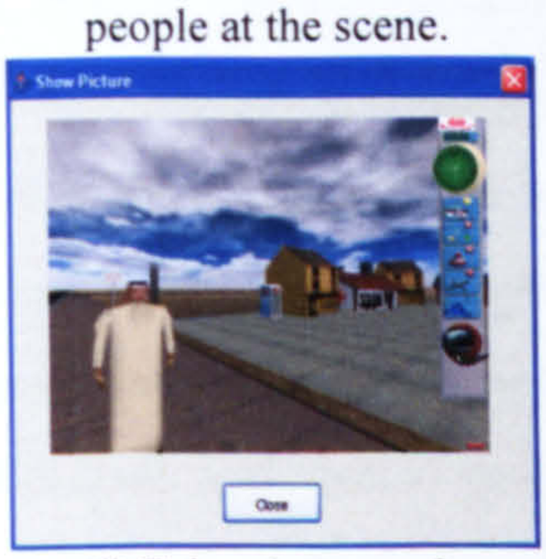
(f) Attend injured.

(g) Identify and question people at the scene.

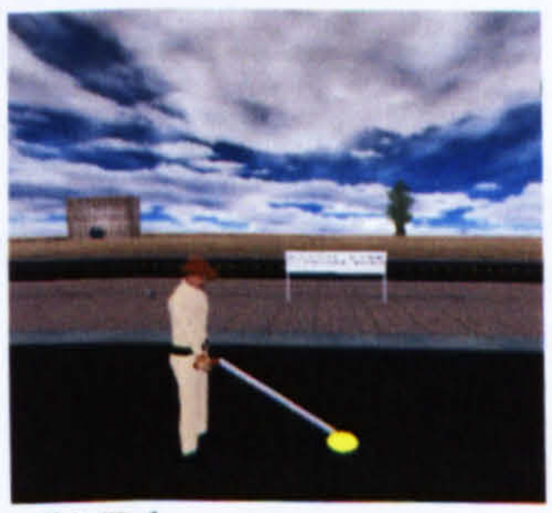
(h) Search for clues.



(i) Mark clues.

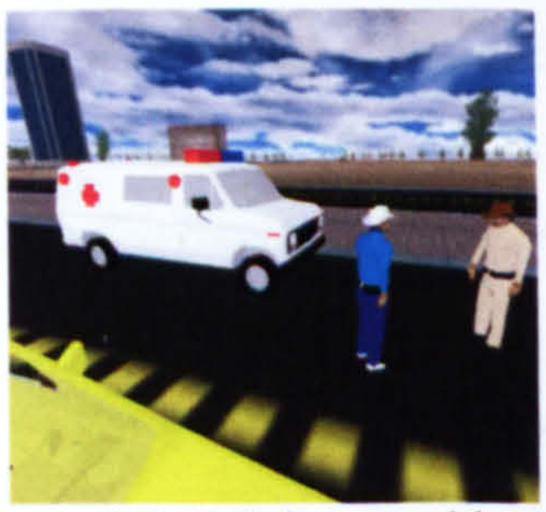


(j) Take photographs.



(k) Take measurements.

(l) Draw the accident scene.



(m) Collaborate with paramedics.

(n) Collaborate with tow truck operator.

Figure 6.4: A typical virtual traffic accident investigation experience (the drawing of the accident in (l) is completed outside the game).

6.3.2 Phase 2: Arriving at the Accident Scene

Upon arriving at the accident scene, and after the police car stops, the investigator finds himself outside the car. The state of the accident scene upon arrival is shown in Figure 6.8. The first task is to secure the accident scene to ensure his and others' safety by parking the police car in an appropriate place and by using traffic cones to help direct traffic. The recommended place to park the police car is in a way that blocks the accident lane which helps alert oncoming traffic. To do that the investigator clicks on the car and a green circle appears underneath the vehicle highlighting the selection and four arrows appear on the right menu underneath the radio button which are used for moving the car (see Figure 6.5). The investigator should also use the cones to help direct the traffic. Clicking on the cones icon adds a cone directly in front of the investigator who can then select it and move it to the appropriate place in a similar manner to moving the car. Some actions require the investigator to explain why he carried out a task (photographing, measuring, placing cones and markers, and requesting assistance). This is done to allow for a moment of reflection and also to provide the trainer with insight into the investigator's thinking (see Figure 6.6).

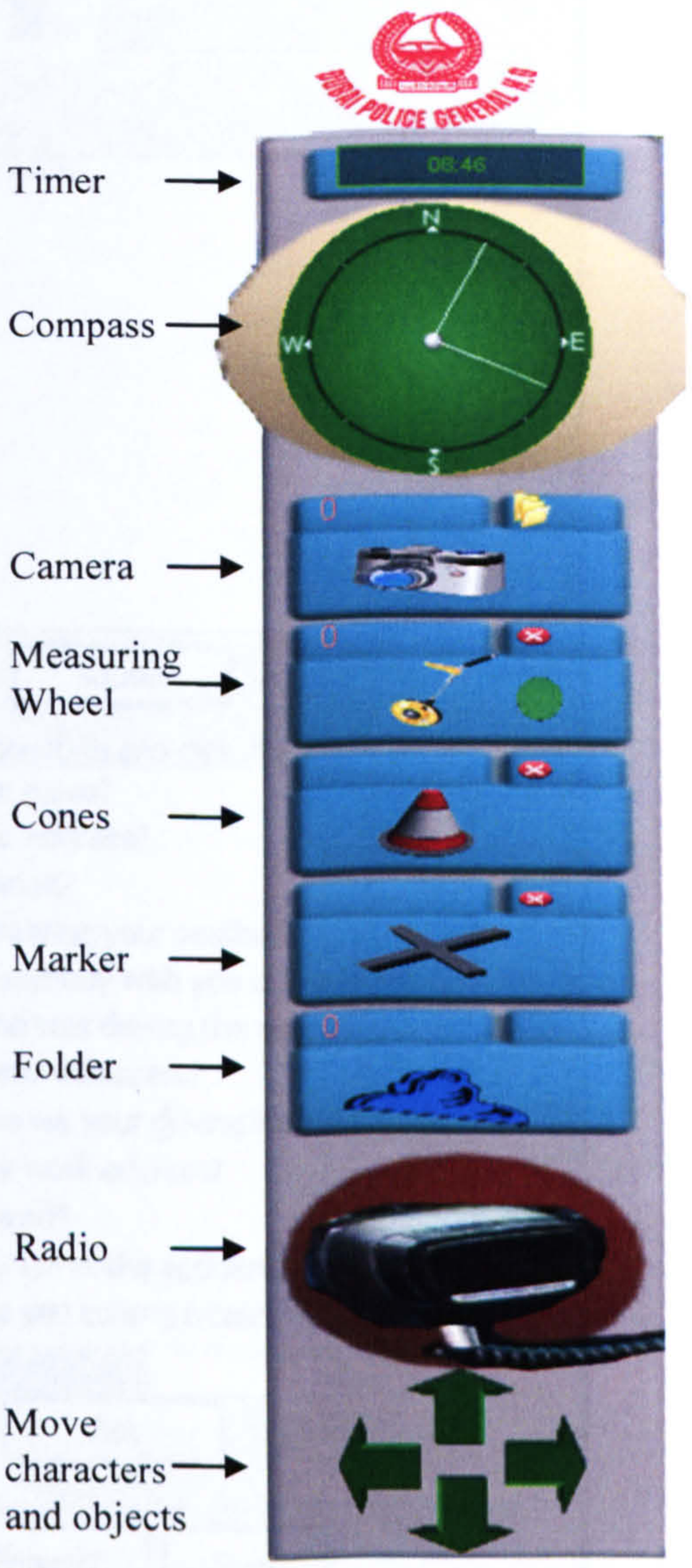


Figure 6.5: The menu.

The investigator should then search for injured people and for sources of danger (e.g. petrol leakage and people in the middle of the road). To do that the investigator can navigate around the scene and look inside the vehicles. He can also ask people at the accident scene what they saw (see Figure 6.7b). For injured people the investigator can contact the operator and request the appropriate number of ambulances (see Figure 6.7a). In the case shown in Figure 6.8 there is one slightly injured driver

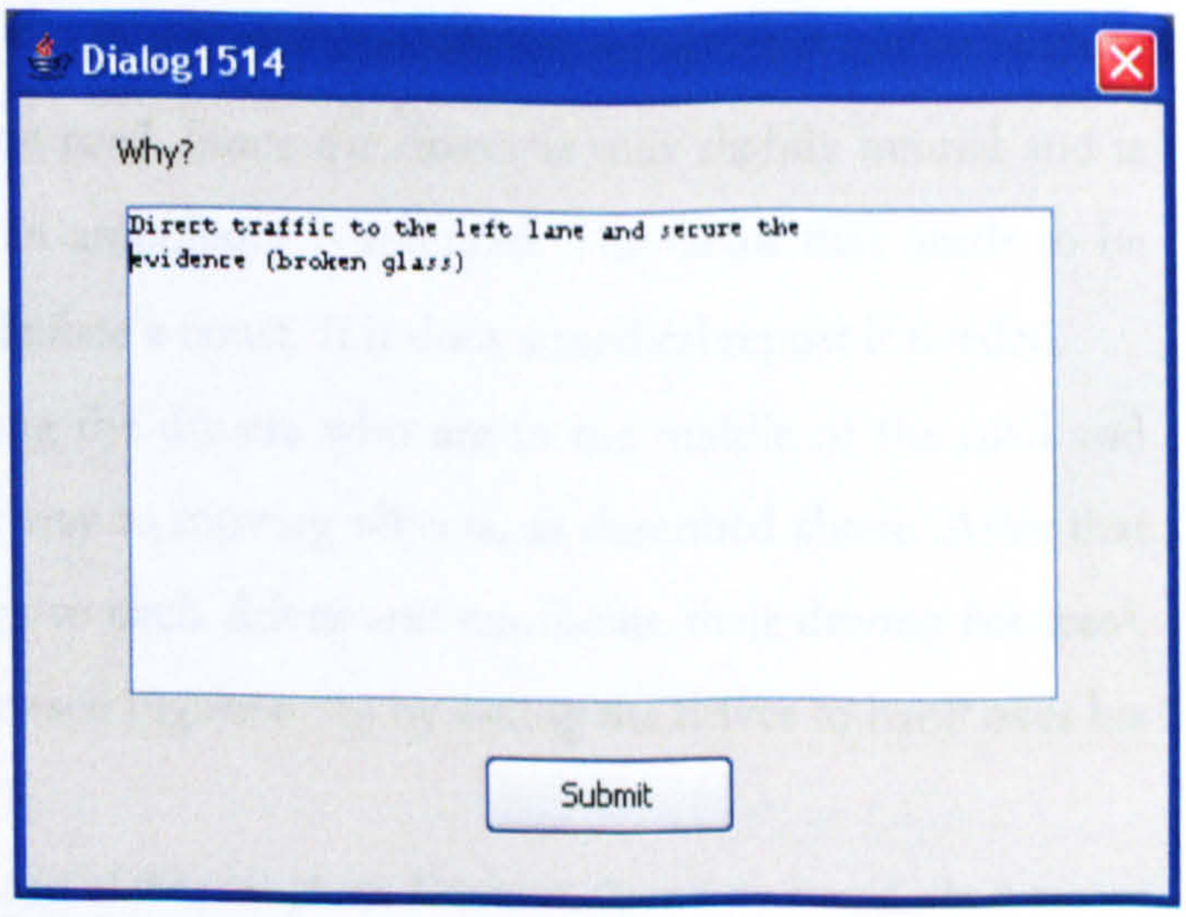
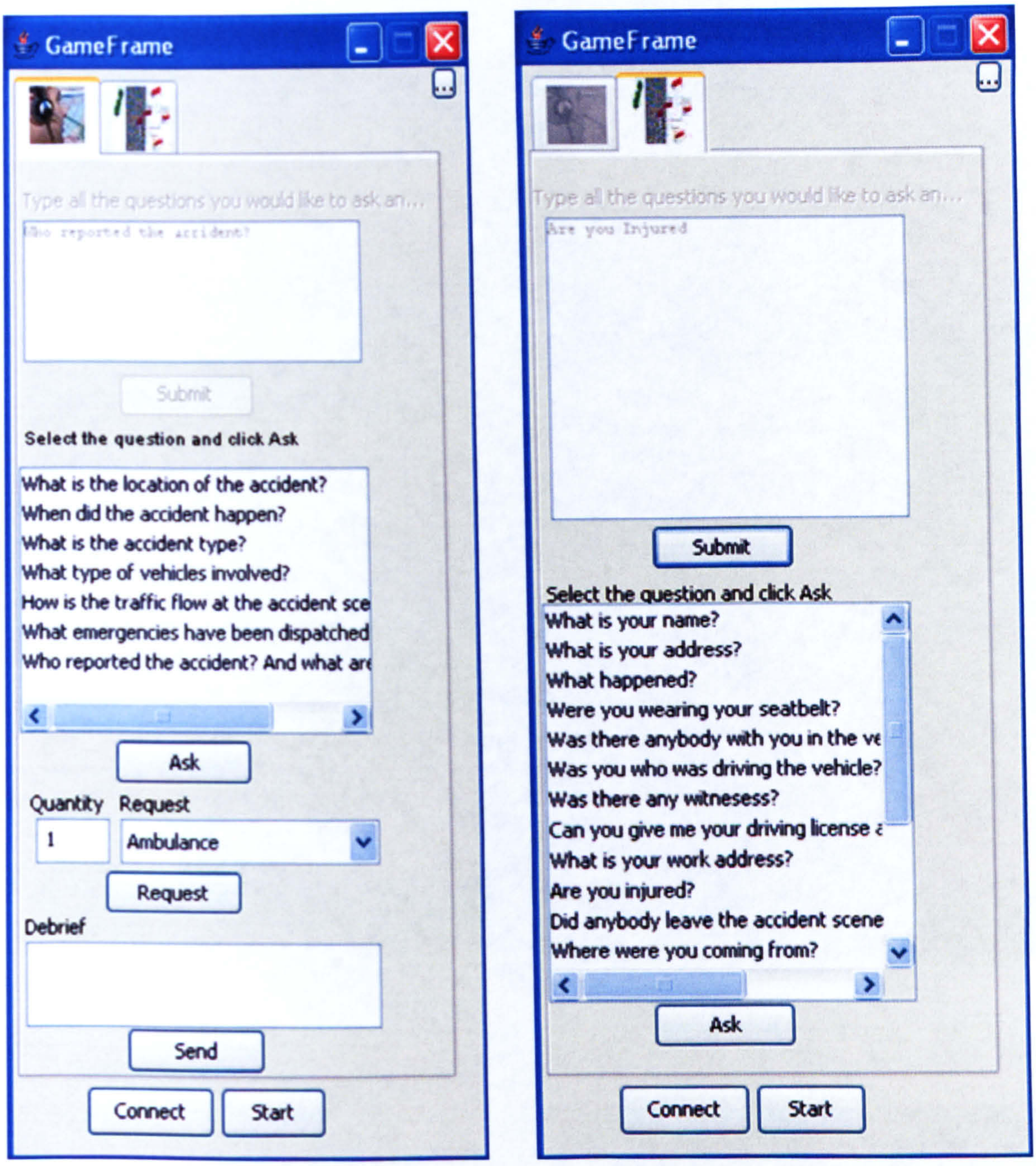


Figure 6.6: Reflection box.



(a) Communicating with operation room.

(b) Communicating with characters.

Figure 6.7: The communication interfaces.

and both drivers are standing in the middle of the road. Since the driver is only slightly injured and is standing, it is up to the investigator to decide if an ambulance is required. The factor that needs to be considered is whether the accident requires to go before a court. If it does a medical report is needed.

The investigator must also notice the risk facing the drivers who are in the middle of the road and should move them. This can be done in a similar way to moving objects, as described above. After that the investigator should identify which car belongs to each driver and confiscate their driving licences[3]. This is done through the communication interface (see Figure 6.7b) by asking the driver to hand over his

[3] According to training manual, confiscating the licenses is part of the next phase. However, there is no reason why it cannot be done in this phase since a contact has already been established with the drivers.

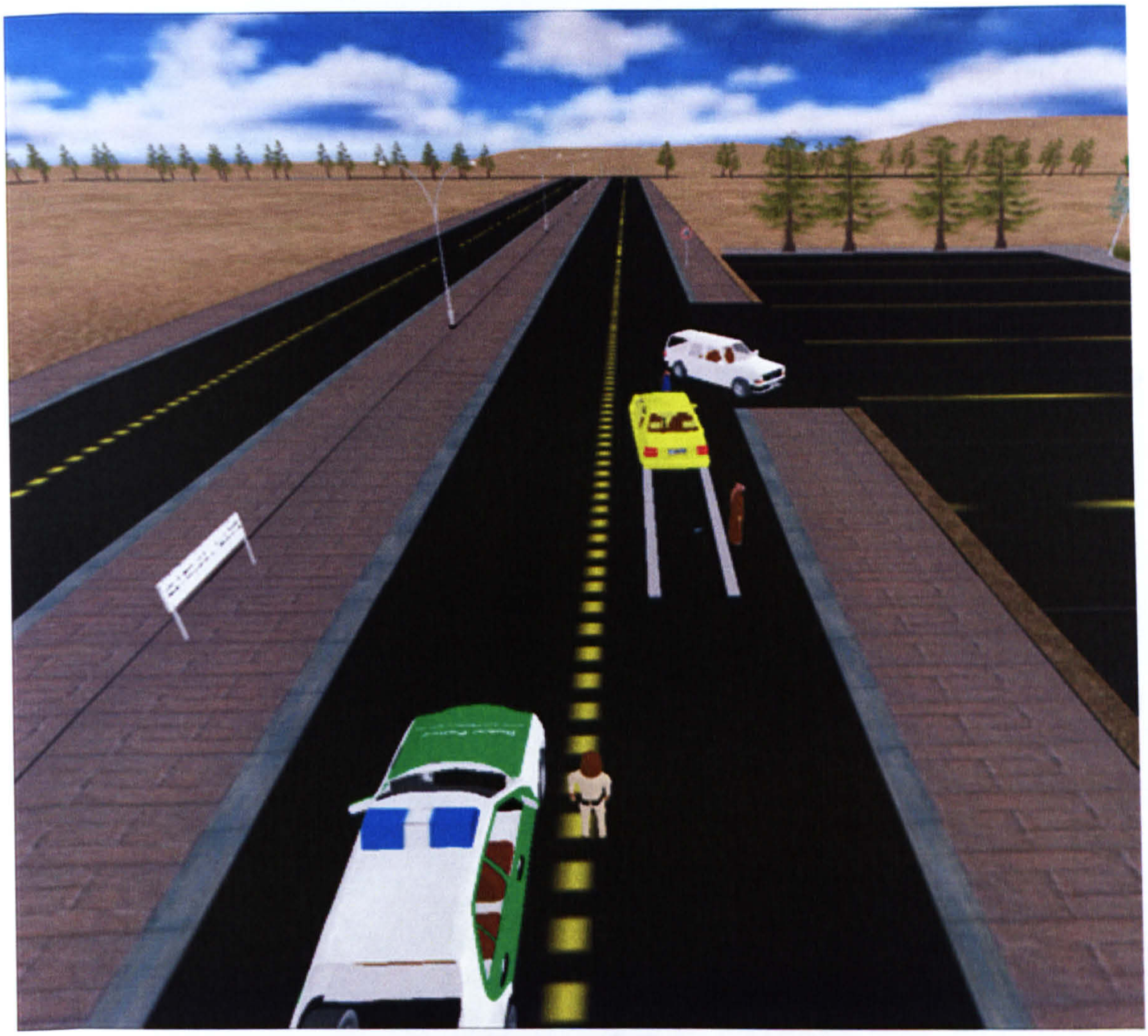


Figure 6.8: Accident scene upon arrival.

driving license. After requesting the drivers' licenses they are added to the investigator's folder shown on the right menu with a counter on top of it indicating the number of items in the folder (see Figure 6.5). After the licenses are added to the folder the investigator can click on the folder and view the licenses (see Figure 6.9).

The last step in this phase is to secure any clues. This is done by marking their positions with an 'X'. For instance there is broken glass in the middle of the road which is indicative of debris from the accident. To mark its position the investigator clicks on the 'X' sign on the right menu and the mark 'X' appears in front of him which he can move to the appropriate place. The

Figure 6.9: Driving license.

investigator is prompted to explain what he is doing. This process is similar to adding a cone. The investigator needs to do the same for marking the positions of the vehicles and for the skid marks. Figure 6.10 shows the scene after completing this phase.

6.3.3 Phase 3: Conducting Initial Investigation

There are three main tasks in this phase: questioning people, photographing the scene, and measuring the scene. The questioning task needs to establish the sequence of events that led to the accident from

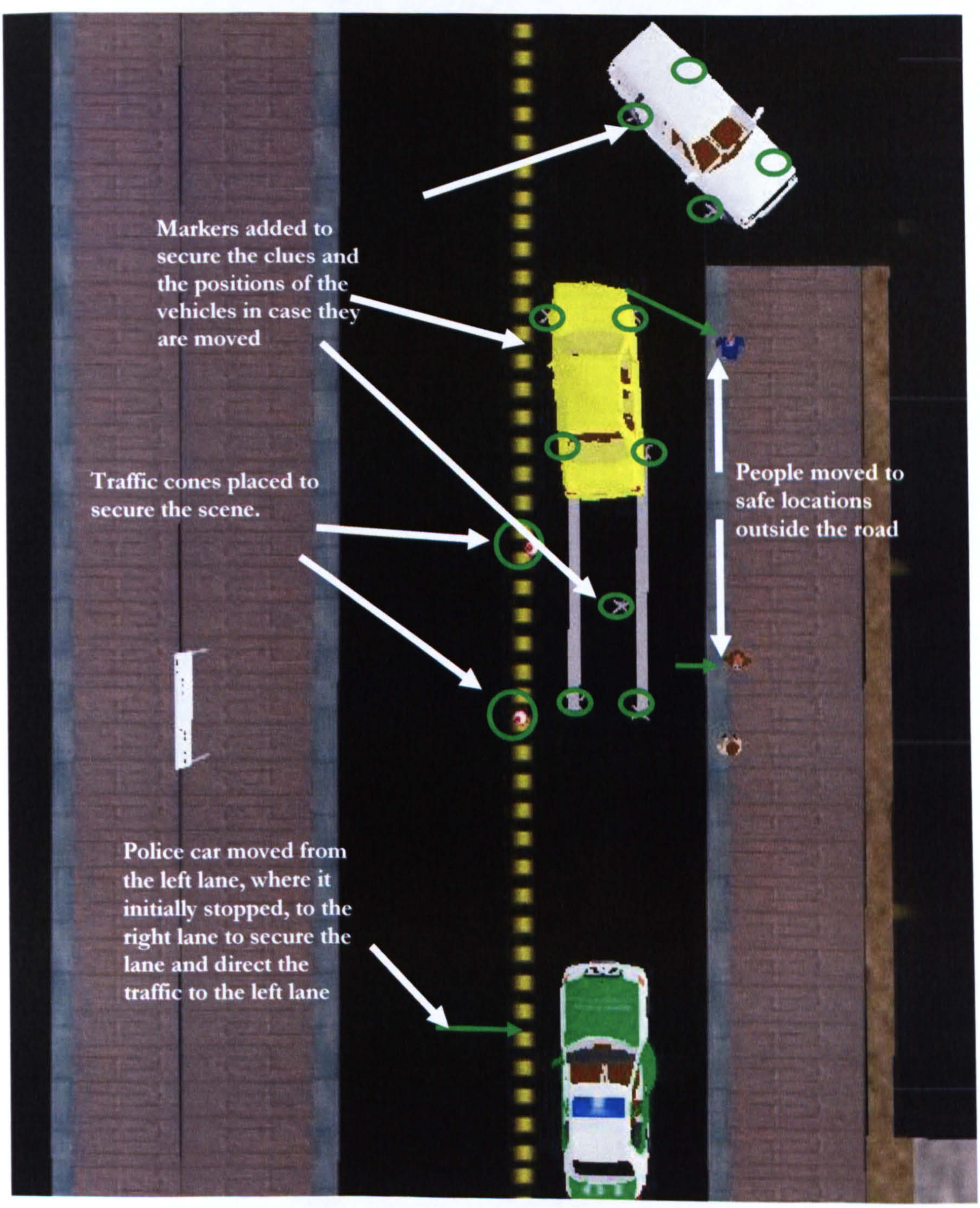


Figure 6.10: Accident scene after securing the scene and evidence (green arrows show where the police car and drivers have been moved from and to and circles show what has been added to the scene). The text, circles, and arrows are for illustrative purposes only.

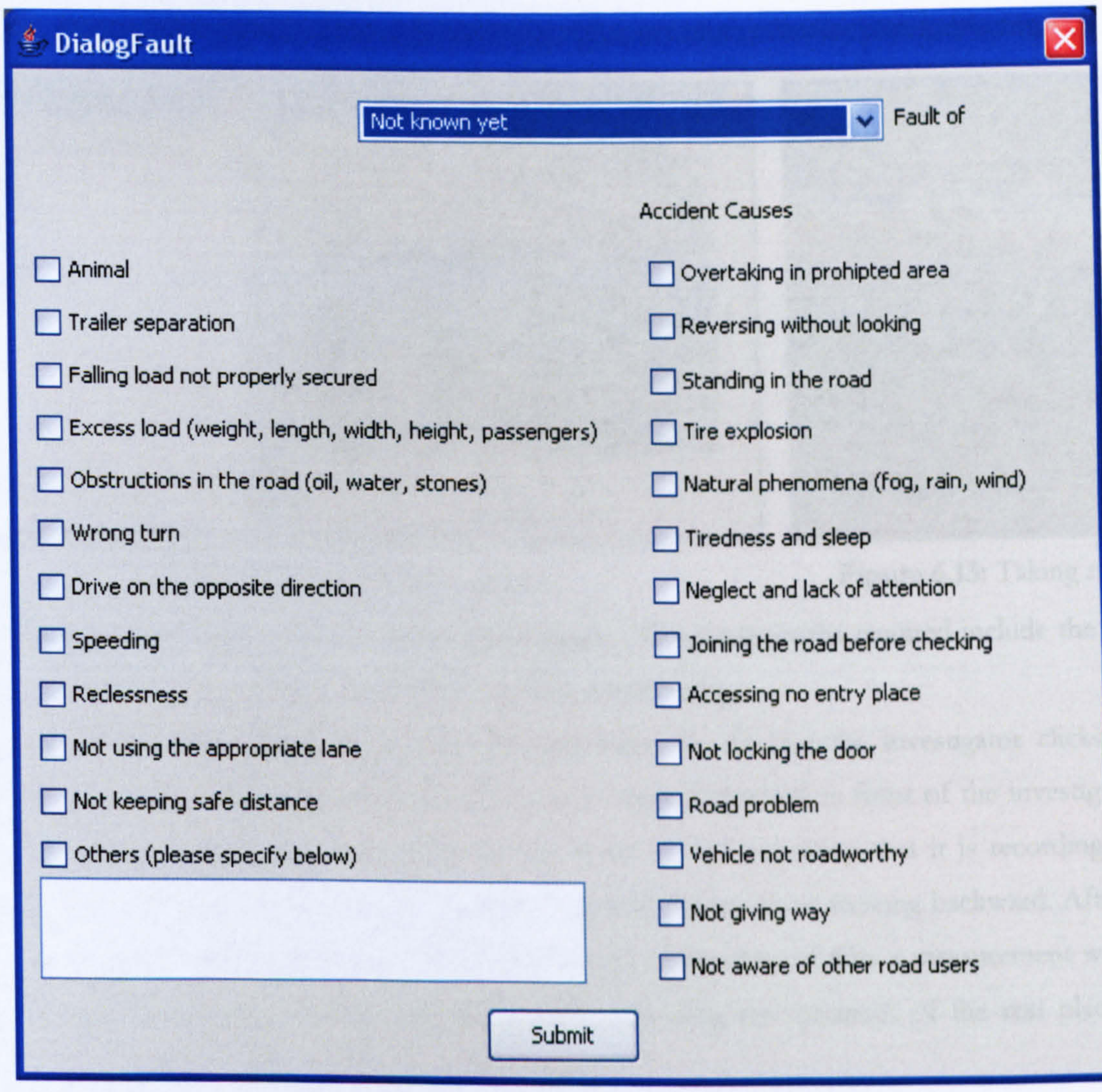


Figure 6.11: The dialog to record who is at fault.

both drivers' perspectives. It should also establish if the clues and damage to the vehicles match with their stories. The investigator should wait until he gathers all the evidence before deciding who is at fault. At different time intervals (every five minutes) the investigator is prompted to decide who is at fault and the type of fault (see Figure 6.11). One of the options available is to select that it is unknown yet. The recorded data can help the trainer, along with the full interaction list, to pinpoint when the investigator has made up his mind, which can reveal a number of things about the investigator's decision making process. If the investigator made up his mind, based on looking at only some of the clues, this indicates poor decision making skills. Another pointer to poor decision making skills is if the investigator rushes to make a decision and then after finding out that the clues contradict his decision, he still refuses to rectify it.

The other task in this phase is to start photographing the scene. To do that the investigator clicks on the camera icon on the main menu (see Figure 6.5) which takes a snapshot of the screen and stores the picture (see Figure 6.12). The investigator is prompted with a reflection box to specify why he has taken a picture (see Figure 6.6). The counter on top of the camera indicates the number of photographs taken

Figure 6.12: Taking photographs.

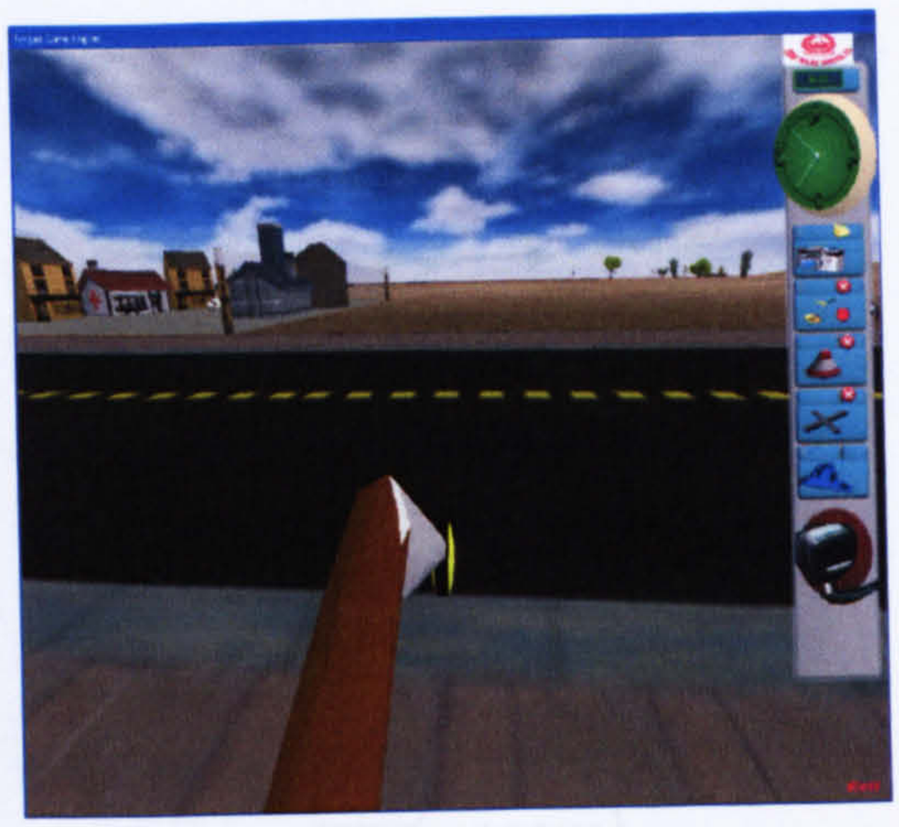


Figure 6.13: Taking measurements.

and clicking on the folder icon shows the photographs. The photographs required include the vehicles from all directions, the final rest place of the vehicles and the clues.

The last task in this phase is to take measurements. To do that the investigator clicks on the measurement icon on the right menu which places a measuring wheel in front of the investigator (see Figure 6.13) and changes the green circle on the menu to red indicating that it is recording. As the investigator moves forward the counter increases and it decreases when moving backward. After taking the measurement a reflection box appears asking for an explanation of why a measurement was taken. The measurements needed in this scenario include measuring the distances of the rest place of the vehicles and the broken glass from the accident point.

6.3.4 Phase 4: Finalizing the Data Collection

In this phase the investigator needs to pinpoint the exact location of the accident with regards to a landmark (lamppost or road sign). This is done by measuring the distance between it and between the accident point (the location of the broken glass). The other measurement required is the road width. Additionally, a few more photographs are needed (e.g. road condition, and the whole scene). If the vehicles are unable to move a tow truck must be called. This is done through the operator interface in a similar manner to requesting an ambulance. When the tow truck arrives it reports to the investigator and asks for permission to leave. When granted the vehicles are towed away. Finally the investigator should by now have collected all the necessary information to be able to assign liability based on the evidence available.

6.3.5 Phase 5: Completing the Accident File

In this stage the investigator is required to draw the accident scene. Figure 6.14 a drawing sample of the accident scene. Investigators often do initial drawings at the accident scene and if the case needs to go before a court the drawings are redone in the police station on a computer or on paper. The drawing should indicate the vehicle positions, and orientations, road type, distances, landmarks, etc. Once the

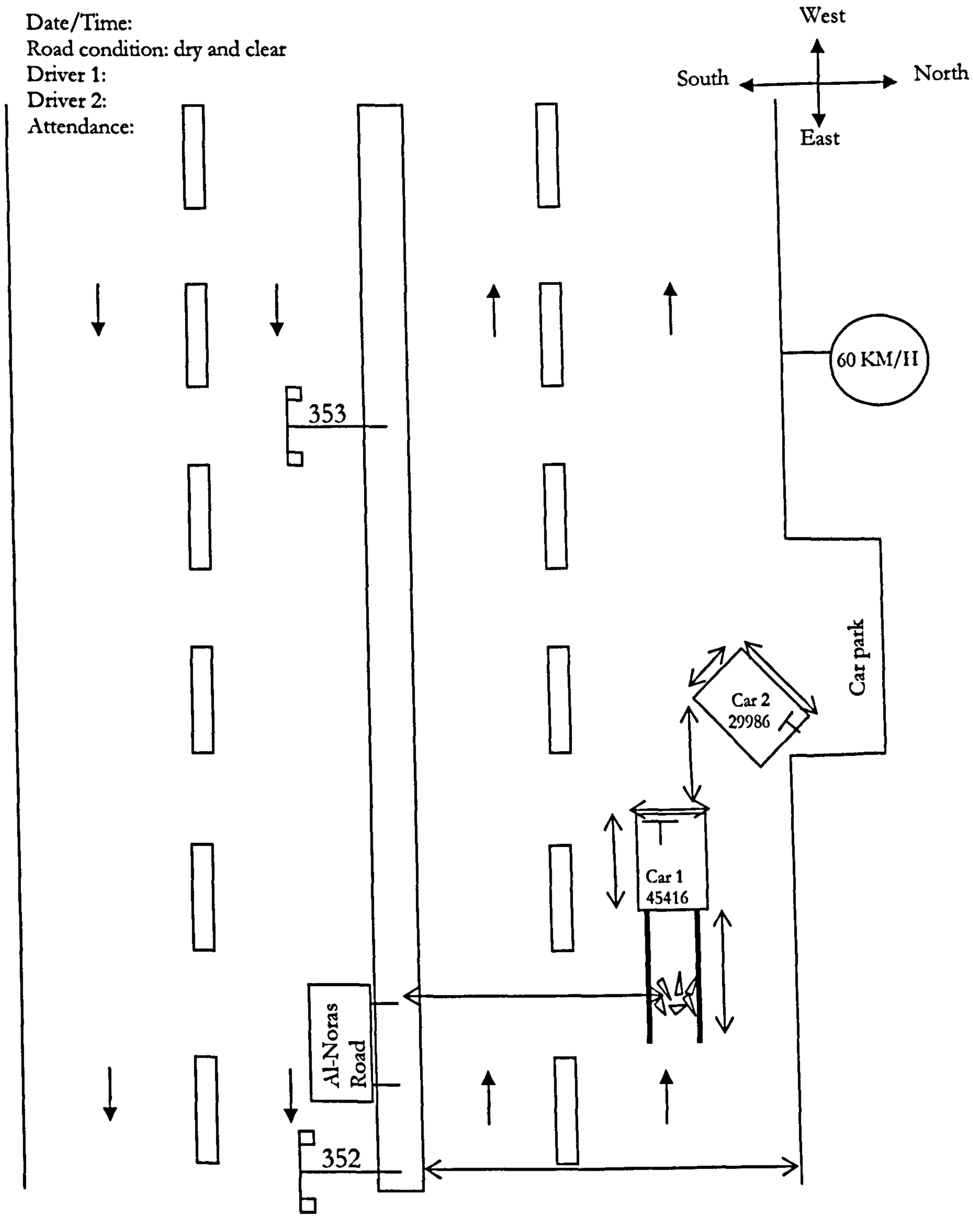


Figure 6.14: A drawing of the accident scene.

drawing is complete the investigator clicks on finish and the self-assessment wizard described in the next section starts.

6.3.6 Assessment

The self-assessment wizard calculates the scores for the following tasks: measurements, photographs, placing markers, and drawing. The screens for measurements, photographs, and placing markers list the

tasks required and a drop down list of the tasks carried out by the investigator. For the drawing list a check list of the tasks required are presented and the investigator ticks the ones carried out. Once completed a folder is created which includes (see Figure 6.19): a score sheet, pictures taken during the investigation, and a page showing all the interactions done by the investigator. The trainee can also generate images of the final accident scene. He can also generate the navigational path he has followed.

6.4 Learning Principles

The previous section provided a walkthrough demonstrating how the traffic accident investigation experience is replicated in SGTAI. In this section, the way the learning is embedded in the virtual experience (i.e. the gameplay) with the help of instructional principles is explained. First, however, learning theories are considered. Whilst no evidence could be found to point to one particular learning theory being effective at explaining why learning occurs in serious games (see section 5.5.4), experiential learning theory is mentioned in a number of serious games (Buch & Egenfeldt-Nielsen, 2006; Dieleman & Huisingh, 2006). Although Kolb's experiential theory may not be the most recent or the most widely appreciated learning theory (Nielsen-Englyst, 2003), its use in serious games shows that it has been found to help in understanding of how learning occurs. This being the case, it should aid the design process. Nevertheless, learning theories in general are known to lack the ability to provide prescriptive guidelines (Morrison et al., 2003) which means their role in the design process is usually confined to being descriptive. Instructional principles have been found to be effective at compensating for the shortcomings of learning theories (Morrison et al., 2003). SGTAI bases the instructional principles it uses on Aldrich's elements (Aldrich, 2005) (see Virtual Leader (Aldrich, 2004) for an example), which are explained in more detail in section 6.4.2. In addition, as part of the design process, SGTAI incorporates feedback loops at multiple stages in order to enhance learning.

Furthermore, it should be noted that SGTAI has also relied on preliminary experimentation to help identify what it must focus on (BinSubaih et al., 2005a) and on an iterative process of development and testing to improve different aspects of it, e.g. the graphical user interface and the voices used by virtual characters.

6.4.1 Experiential Learning Principles

Figure 6.15 shows the experiential learning principles used to build SGTAI. The assumption made is that the learner is going to enter Kolb's experiential cycle already having gone through the Dubai police college course material (hearing and seeing) and looking to put the knowledge, skills, and attitudes learnt into practice (doing). According to Kolb's experiential cycle, a learner can enter at any of the four stages (Smith, 2001). In SGTAI, the learner enters the experiential cycle at the concrete experience (CE) stage and finds himself in a virtual environment in which he goes through the investigation experience. The focus of this stage is to help the learner experience the complexity of reality. For example, during this

- Concrete experiences allow learners to understand the richness and complexity of reality (CE).
- Learning is stimulated through reflective observation (RO).
- Using previous experiences and feedback, learners construct universal principles on how to solve problems (AC).
- Using knowledge from the AC stage learners plan how to do the task differently in order to solve problems (AE).
- When using active learning, learners pay more attention, draw on prior knowledge, require deeper processing of material, and become more motivated.
- Discovery learning develops a meaningful learning which confronts learners' current ideas and aids in modifying them.
- Discovery learning also changes learners' attitudes and values by helping them to understand that learning is a process not only a set of facts and places the responsibility on learners to tackle the problem and come up with a solution.

Figure 6.15: Experiential learning principles used to build SGTAI.

phase a trainee investigator may find it difficult to perform an investigative task such as taking measurements at the accident scene. Figure 6.16 illustrates the measurement example across the four learning stages.

After completing the CE stage the learner enters the reflective observation (RO) stage in which he reviews and reflects on his experience. The focus of this stage is to stimulate the learning process. The trainee who had difficulty in performing the measurement task (see Figure 6.16) can consider his performance during this stage. In the abstract conceptualisation (AC) stage the learner draws conclusions from his experience. This stage requires the learner to be informed about his task by a trainer or from reading a manual or from other sources. SGTAI provides the learner with a self-evaluation wizard to help him evaluate his performance with model answers. His self-evaluation is also logged to be approved by the trainer.

SGTAI also logs and tracks the learner's actions and movements in the environment to help the learner and the trainer reflect on the performance. After the wizard the student also gets a score sheet which clearly marks the tasks that have been completed successfully or otherwise. This phase aims to help the learner understand the theories and philosophies that are generally applicable (Dieleman & Huisingh, 2006). For the measurement task (see Figure 6.16) this stage provides the trainee with feedback on his performance by pointing out what measurements need to be taken.

The final stage is the active experimentation (AE) stage where the learner forms the basis for the planned changes. Dieleman and Huisingh describe this phase as the ultimate phase of transformation since its objective is to manipulate the outside world through the implementation of the change. For the measurement task, the trainee would plan what he needs to do differently and apply this in the next training session. Similarly, the other investigative tasks (e.g. photographing, placing markers, and drawing) can be shown to correspond to the four learning stages.

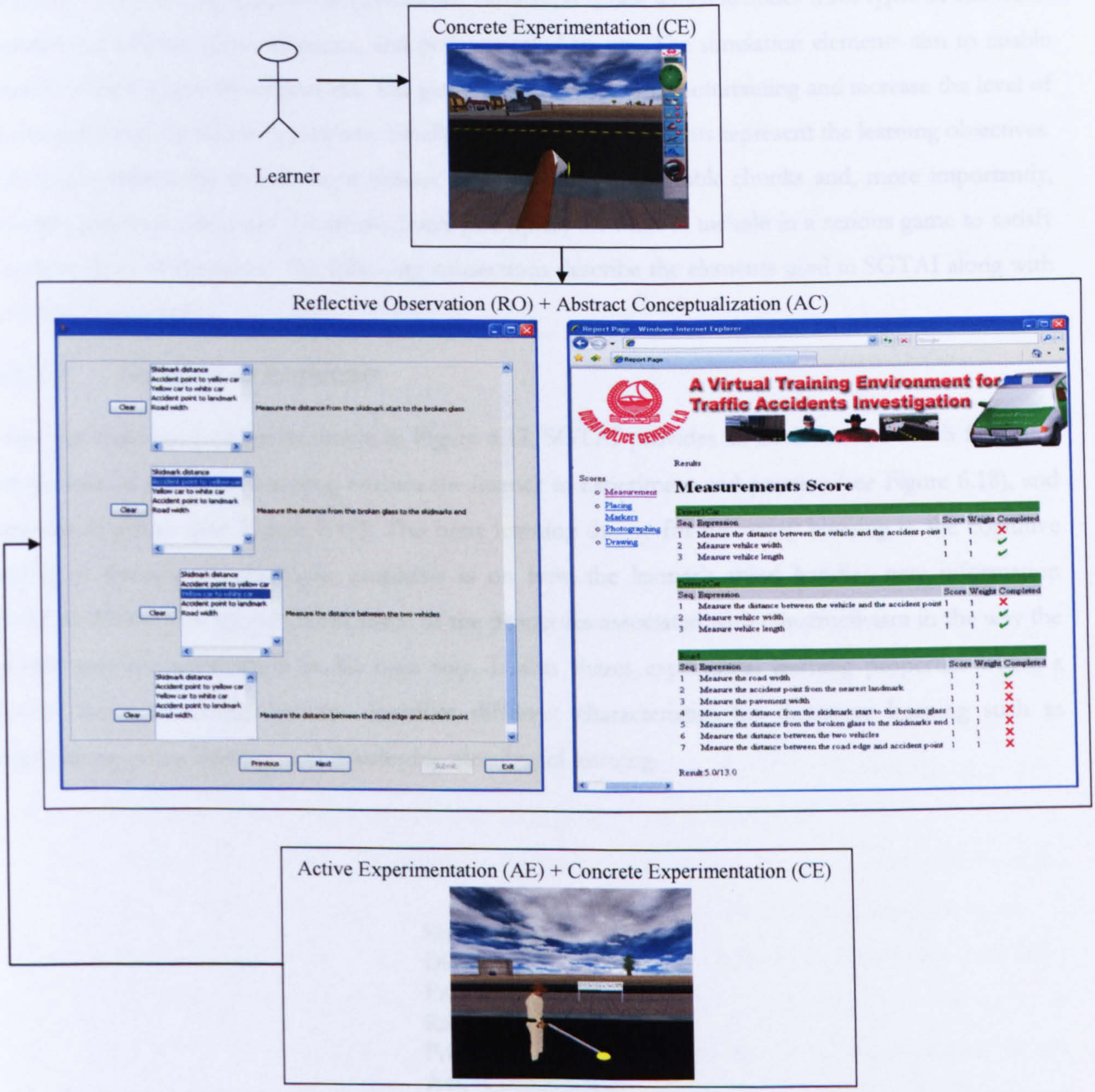


Figure 6.16: An example showing how taking measurement corresponds to the four learning stages.

6.4.2 Instructional Principles

The use of experiential learning principles is helpful in describing how a serious game facilitates learning, but it lacks, as do the learning theories in general, the ability to provide prescriptive guidance (Morrison et al., 2003). The instructional theories however are more prescriptive. Examples include Gagne's nine instructional principles, Reigeluth's Elaboration Theory, Bruner's Psycho-Cultural approach, and Merrill's First Principles of Instruction (Becker, 2006c). For SGTAI, Aldrich's elements (Aldrich, 2005) were used, despite the fact they are not described as an instructional theory. The reason why they can be used is because the elements represent how the content of game (e.g. learning objectives and background material) can be delivered (e.g. simplified interfaces, conflict, practice, scoring, and

feedback). Aldrich's perspective on educational simulation is one which includes three types of elements: simulation elements, game elements, and pedagogical elements. The simulation elements aim to enable transfer of learning to the real world. The game elements aim to be entertaining and increase the level of enjoyment from the whole experience. Finally the pedagogical elements represent the learning objectives. This perspective helps in viewing a serious game in more manageable chunks and, more importantly, provides practical guidelines (i.e. instructional principles) on what to include in a serious game to satisfy the three types of elements. The following subsections describe the elements used in SGTAI along with how they were used.

6.4.2.1 Simulation Elements

From the simulation elements shown in Figure 6.17, SGTAI provides an environment which facilitates the process of discovery learning, enables the learner to experiment and practice (see Figure 6.18), and provides feedback (see Figure 6.19). The basis learning theory for discovery learning is the cognitive model of learning in which the emphasis is on how the learner's mind handles new information (Svinicki, 1998). This model shares some of the properties associated with constructivism in the way the learner acquires information in his own way. It also shares experiential learning properties where a learner learns by doing. Svinicki describes different characteristics for discovery learning such as emphasizing active learning, and developing meaningful learning.

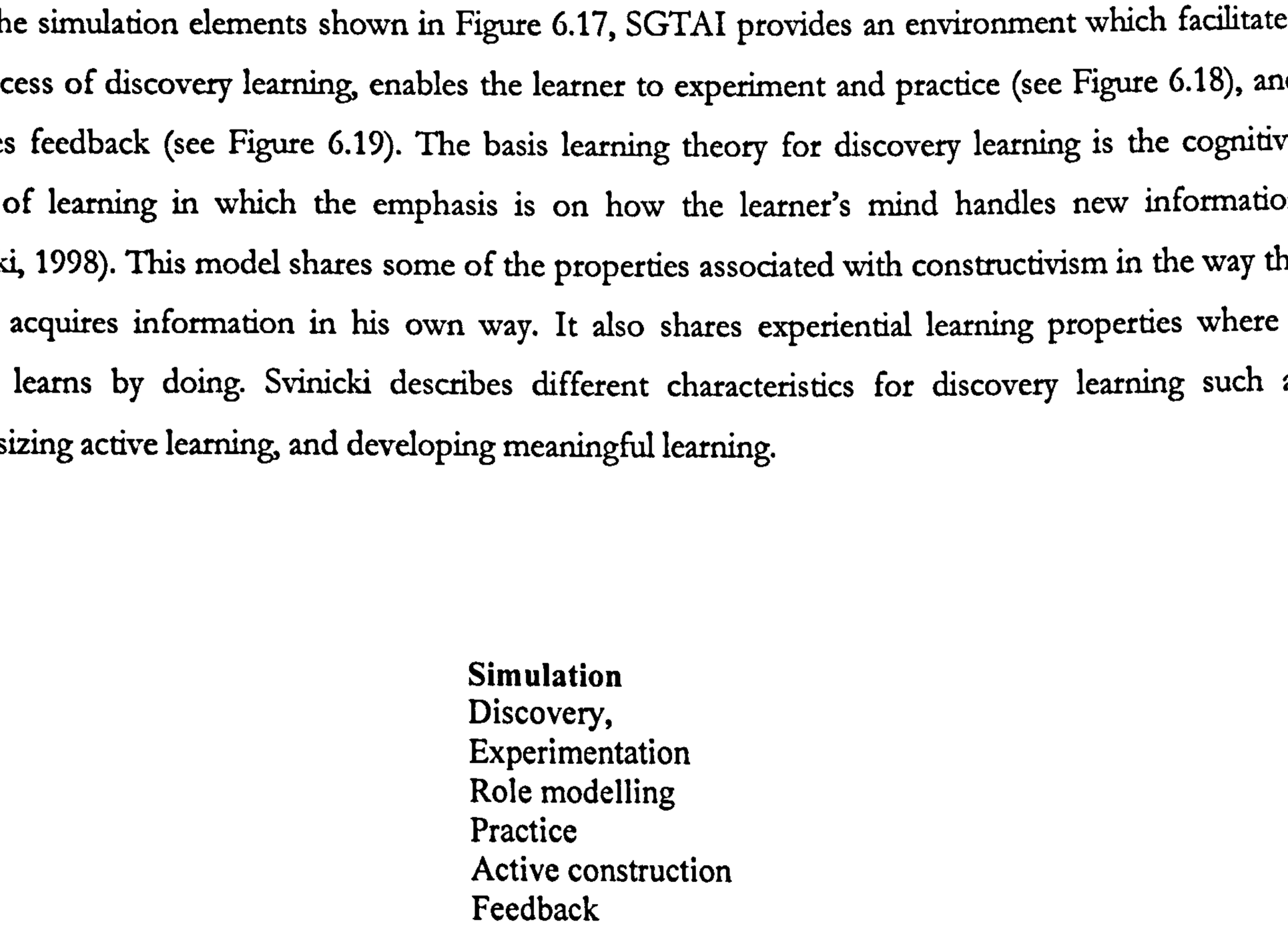


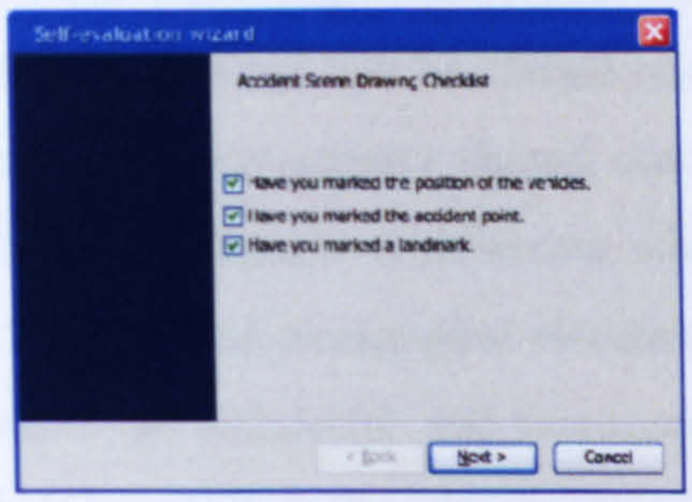
Figure 6.17: Aldrich's three elements (after (Aldrich, 2005)).

Instructional Principles upon which SGTAI was built		
Simulation elements: • Provide learners with an environment that allows them to discover learning by performing meaningful tasks. • Align the learning environment to the environment in which learners are expected to perform. • Provide an environment where learners can practice and experiment. • Physical fidelity. • Functional fidelity.	Pedagogical elements: • Identify learning objectives (e.g. measuring, photographing, etc). • Identify instructional problems. • Decide on what to simulate and the fidelity of the simulation. • Force moments of reflection. • Score and diagnose the performance. • Store libraries of successful and unsuccessful plays.	Game elements: • Use a known game genre. • Use exaggeration. • Use time and score to provide a challenge. • Use graphics and sound. • Balance fun. • Allow for multiple skill levels. • Set achievable goals.

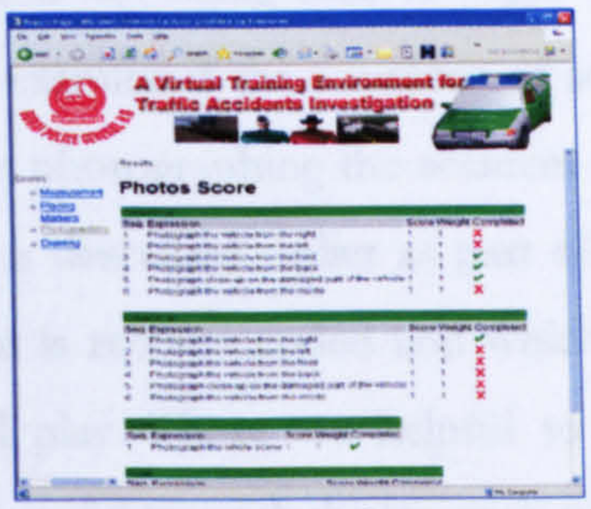
Figure 6.18: Simulation, game, and pedagogical elements used in SGTAI (Aldrich, 2005).

SGTAI follows the first principle for the active learning characteristic by enabling the learner to be actively participating, which means he is paying more attention to learning in general. The different investigative phases and tasks focus the learner's attention on the key ideas that are being examined, which should lessen the influence of distractions. The learner is also forced to draw on prior knowledge to be able to respond to the activities that require completion of tasks, which results in a deeper processing of the material. To make the learning meaningful in SGTAI the learning context is aligned to the eventual context by using the FPS genre and by providing real problems for the investigator to solve (e.g. taking the necessary measurements when an accident involves two vehicles).

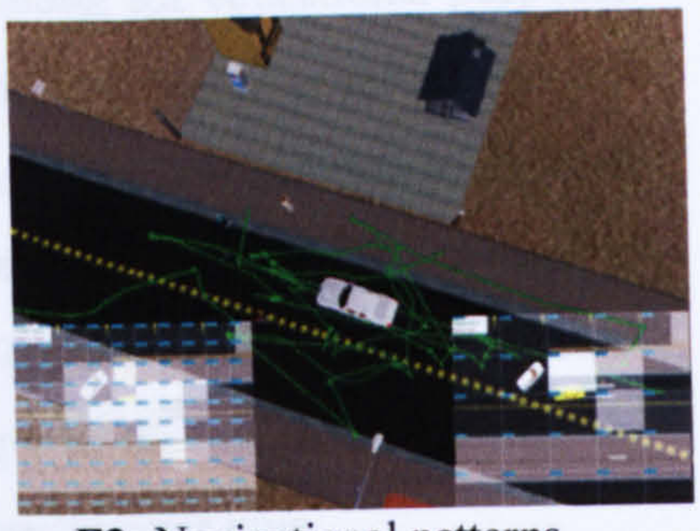
Feedback is also one of the simulation elements mentioned by Aldrich and one of the activities in the abstract conceptualization (AC) stage of experiential learning. Besides the feedback described in the AC stage, SGTAI facilitates the kind of feedback which can occur when learners interact to compare score sheets. This interaction forms part of an important activity outside the game which is referred to as social ecology (Herz & Macedonia, 2002). To do this the output of SGTAI (score sheets, all the interactions recorded, and the navigation path) can be considered as part of what the learner can construct as a 'public entity' (see section 5.5.3) and share and compare with others. Figure 6.19 shows the different kinds of feedback provided by SGTAI. The self-evaluation wizards feedback (F1) is provided during the game. Other kinds of feedback (F2, F3, F4, F5, and F6) can be used between games or outside the game (see Figure 6.20).



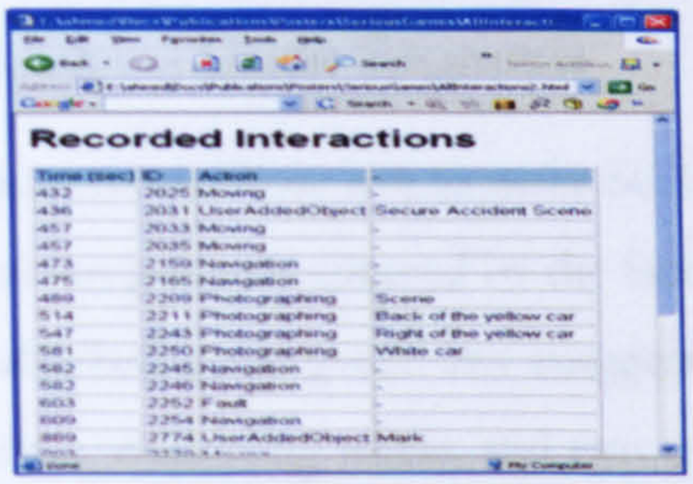
F1: Self-evaluation wizards.



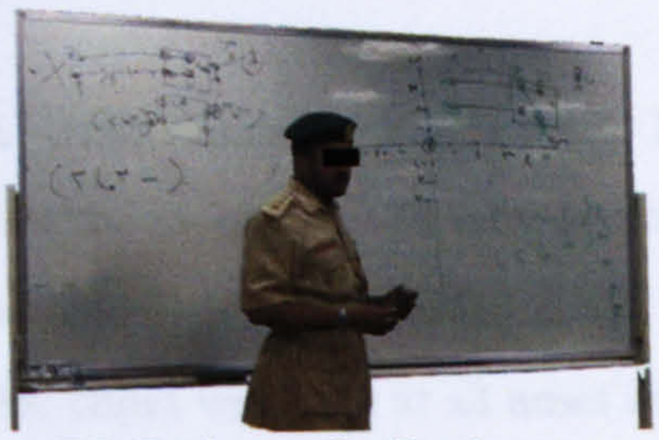
F2: Score sheet.



F3: Navigational patterns.



F4: All interactions.



F5: Trainer's feedback.

F6: Social ecology.

Figure 6.19: Feedback types.

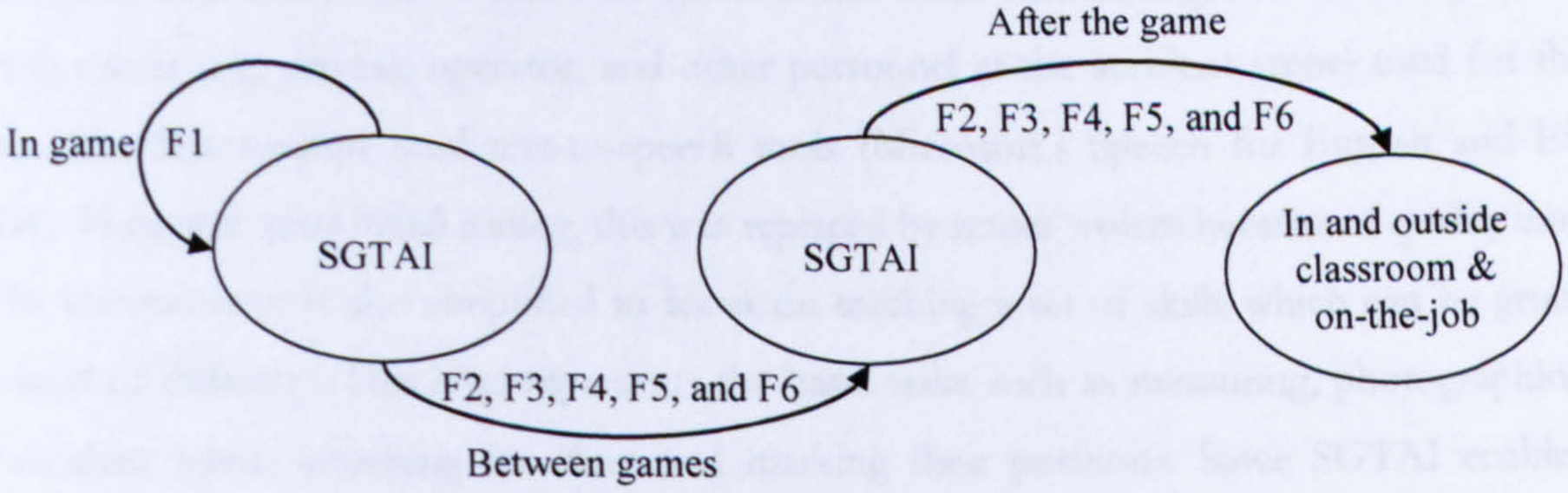


Figure 6.20: Feedback loops.

6.4.2.2 Pedagogical Elements

The pedagogical elements (see Figure 6.18) aim to ensure that the learning objectives are included in the game and describe what needs to be simulated. An example of a learning objective in SGTAI is to teach the learner how to take measurements at the accident scene. These measurements are required to be able to reconstruct the accident scene for further investigation which is often needed for court cases. To do this, SGTAI provides the learner with a way to take measurements, to record these measurements, and to evaluate his performance. Each accident scene has a model answer of what measurements need to be taken. These are used to assess the learner's performance and are presented to him to tick if completed during the self-evaluation stage at the end of the investigation. To ensure the learner is accurately marking himself, the learner's own assessment is recorded for further verification by the trainer. The measurements activities are relevant to the measurement task and therefore ensure that the learning

objective is an integral part of SGTAI and not being used as sugar-coating for educational purposes. Further evaluation can be carried out to examine if the sequence of actions followed is acceptable. For instance, the investigator should not start photographing the accident scene before securing it. Another pedagogical element is reflection which is described earlier as part of the RO stage in the experiential cycle. The final pedagogical element that is recommended and which is present in SGTAI is to store libraries of successful and unsuccessful play. These are helpful to guide the discussion during the debriefing session after the game and are useful to track the learner's progress over time.

6.4.2.3 Game Elements

The game elements aim to make SGTAI entertaining (see Figure 6.18). The first game element (i.e. using a known genre) is covered by the fact that the first-person shooter genre has been chosen since this is an established game genre. An exaggeration element is also added in the way the investigator carries the camera, measuring wheel, and two traffic cones with him at all times and he can just click on the menu for things to appear in the environment. Another element is the challenge element which exists because SGTAI requires the investigator to complete the investigation in the provided time and to achieve a high score.

The game also uses sound to make the environment more entertaining. For the non-player characters' (NPCs) voices (e.g. drivers, operator, and other personnel at the accident scene) used for the dialogue system, the first attempt used text-to-speech tools (Microsoft's Speech for English and EULER for Arabic). However, after initial testing, this was replaced by actors' voices because of quality issues.

The environment is also simplified to focus on teaching a set of skills which can be grouped into a first level of difficulty. This level represents the basic tasks such as measuring, photographing, securing the accident scene, searching for clues and marking their positions. Since SGTAI enables different scenarios to be created, the complexity can be increased by adding different accident types, traffic flow, discrepancies in the statements given, etc. Breaking SGTAI into levels means the player can achieve a sense of advancement and completion.

6.5 Evaluating SGTAI with Real Police Officers

In February and March of 2006 an experiment was conducted to measure the effectiveness of SGTAI (BinSubaih et al., 2006b) as a training tool and to analyze its suitability in addressing the issues facing Dubai police force. The two hypotheses were that SGTAI should be able to improve the performance of both novices and experienced investigators, and that novices would be able to improve their performance by more than the improvements recorded for the experienced investigators. The second hypothesis is based on the fact that the difficulty level planned for the experiment is low and thus experienced investigators would not improve by much. The improvement was measured by conducting pre- and post-training assessments. The suitability of SGTAI was also measured by the comments

received from the participants and the trainers. Section 6.5.1 describes the design of the experiment and section 6.5.2 details the performance measurements used. Section 6.5.3 reports the performance findings and discusses the causes of the performance trends exhibited. Section 6.5.4 illustrates the navigational patterns exhibited. Section 6.5.5 reports on the sense of presence recorded. Section 6.5.6 presents the comments made by participants. Finally, section 6.5.7 presents the limitations of the experiment.

6.5.1 Method

Fifty-six participants were selected randomly from traffic investigators in the Dubai police force. Two groups were required for the study: novices and experienced investigators. The average experience of participants was just under 7 years[4]. All the participants were males. Seven participants were dropped for various reasons: 2 for study leave, 1 for special assignment, 1 for sick leave, 1 felt pressurized by the experiment and requested to stop after the first training session, 1 due to simulator sickness, and 1 due to unrecorded data in the second training session. This resulted in 49 participants for the study.

The experiment design consists of two primary sessions as shown in Figure 6.21. The first session has three parts: agreeing and signing the confidentiality agreement for the experiment, followed by pre-test (see Appendix D) and first questionnaires. All participants went through the first three parts. After this the pre-test results were calculated and they were used to divide participants into two groups (A and B) with similar performance averages. Group A was the control group and group B was the one that was trained. These groups (A and B) were further divided into two groups based on their experience (novices and experienced). This resulted in four groups: novices-A (10 participants), novices-B (16 participants), experienced-A (9 participants), and experienced-B (14 participants).

The control groups have two main roles. The first role is to control the experiment stages to ensure that the pre- and post-tests are of similar difficulty levels. The second role is to use their results to measure the effect training has by comparing them against the trained groups. In session 2, the two A groups (novices and experienced) followed different routes to the two B groups (novices and experienced). The two A groups only took part in the post-test whereas the two B groups went through four parts: familiarization sessions[5], two training sessions, post-test, and a second questionnaire (Slater's presence[6] questionnaire (Slater, 1999)). Figure 6.22 shows images from the experiment.

---

[4] 6.69 years (SD=8.87 and median=1)

[5] If the participant manages to complete the task during the first 15 minutes he progresses to the next stage, otherwise he is asked to retake the training.

[6] Presence is the sense of 'being there' in the virtual environment.

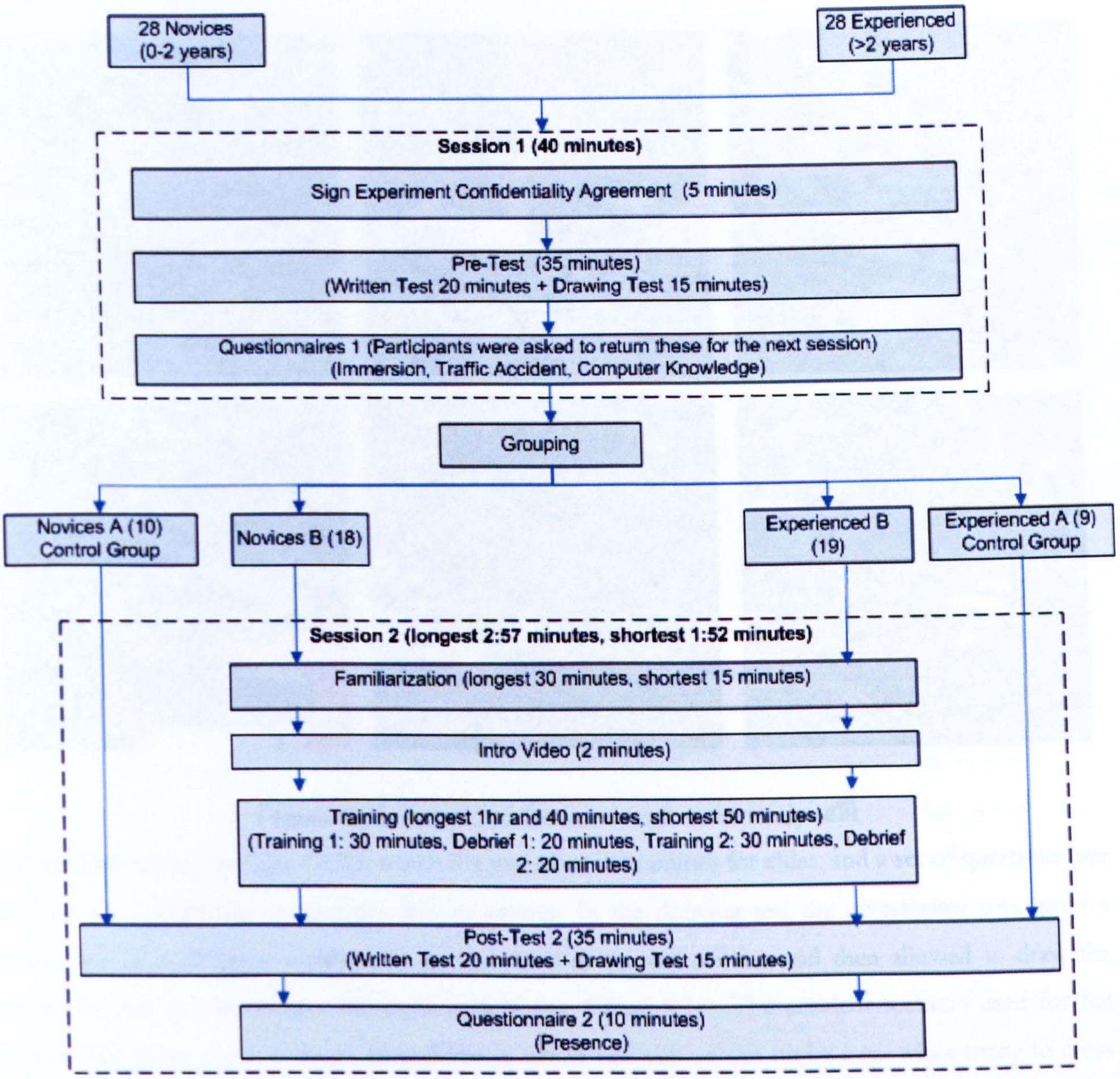


Figure 6.21: Experiment design. The numbers in each group are shown in brackets, e.g. Novices-B had 18 people in it.

6.5.2 Measurements Used

The performance of the trainee investigator was measured based on the successful completion of the following tasks: securing the accident scene (parking police car and placing cones), photographing the accident scene, taking appropriate measurements, placing markers at important clues, and drawing the accident scene. Two trainers approved the marking scheme shown in Table 6.1.

Table 6.1: Marking scheme.

Task	Mark
Parking police car	10.5%
Placing cones	7%
Photographing accident scene	17.5%
Taking measurements	21%
Placing markers	14%
Drawing	30%

The pre- and post-tests each consisted of two parts: a written test and a drawing test. The written test comprised of a short explanation of how the accident happened which the investigator reads to help understand how the accident happened, a 2D drawing of

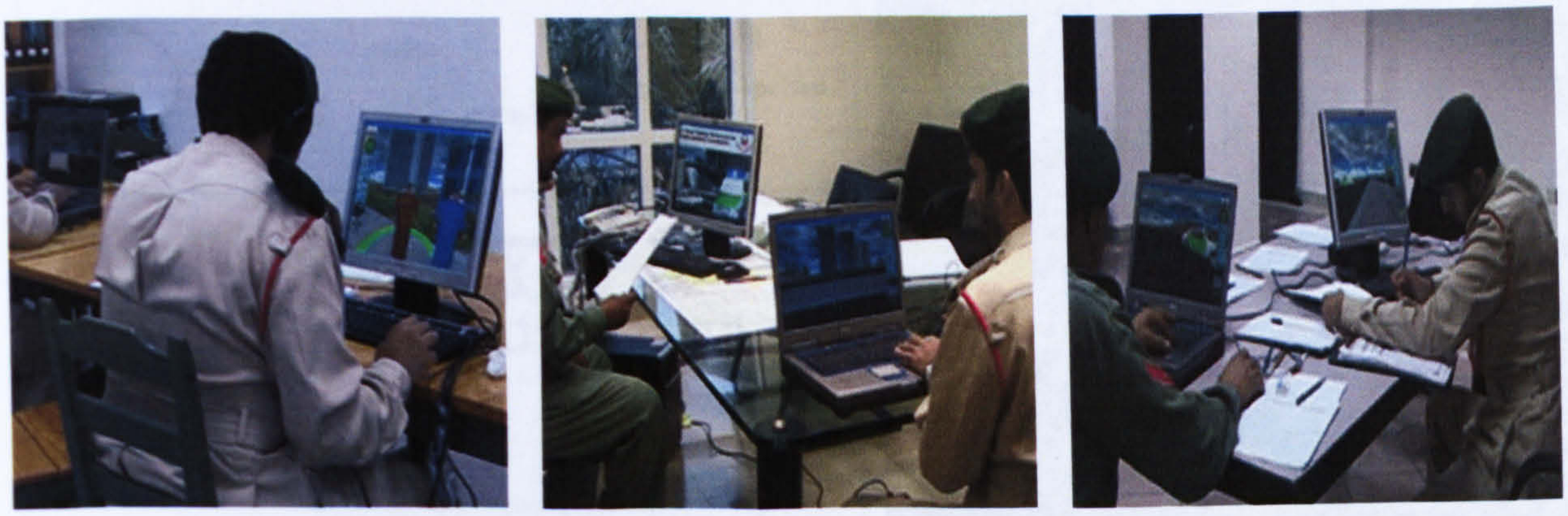


Figure 6.22: Sample of the users used in the study.

the accident scene (see Figure 6.23) which the investigator examines for clues, and a set of questions (see Appendix C) which the investigator has to answer. In the drawing test the investigator was given a description of a different accident to the one used in the written test and then allowed to draw the accident scene by examining a 3D environment (see Figure 6.24). The accident scenario used for the written part of the pre-test shows an accident in which one person was hit by a car while trying to cross the road (see Figure 6.23). The accident used for the drawing part of the pre-test also happens to involve one person being hit while trying to cross the road but the location differs (see Figure 6.24). To stop any learning from taking place, no self-assessment was conducted after the test and the marking was done by the facilitator afterwards.

The training session accident scenario involved a collision between two vehicles as shown in Figure 6.8. One of the drivers was slightly bruised. The investigator had 30 minutes to complete the investigation. Before the session started the investigator was told that he would receive two reminders: one after 15 minutes and one 5 minutes from the end. These would also remind him that the drawing should be also accomplished within the 30 minutes. After the session ended the trainee went through self-assessment wizards which show a list of tasks that had to be completed for the different tasks, and the user was asked to tick the ones accomplished. The system then generated the results. The user examined the results for 10 minutes before the next session started. All the sessions were video taped for further analysis. Participants who did not achieve a score of 70% or above were asked to take the

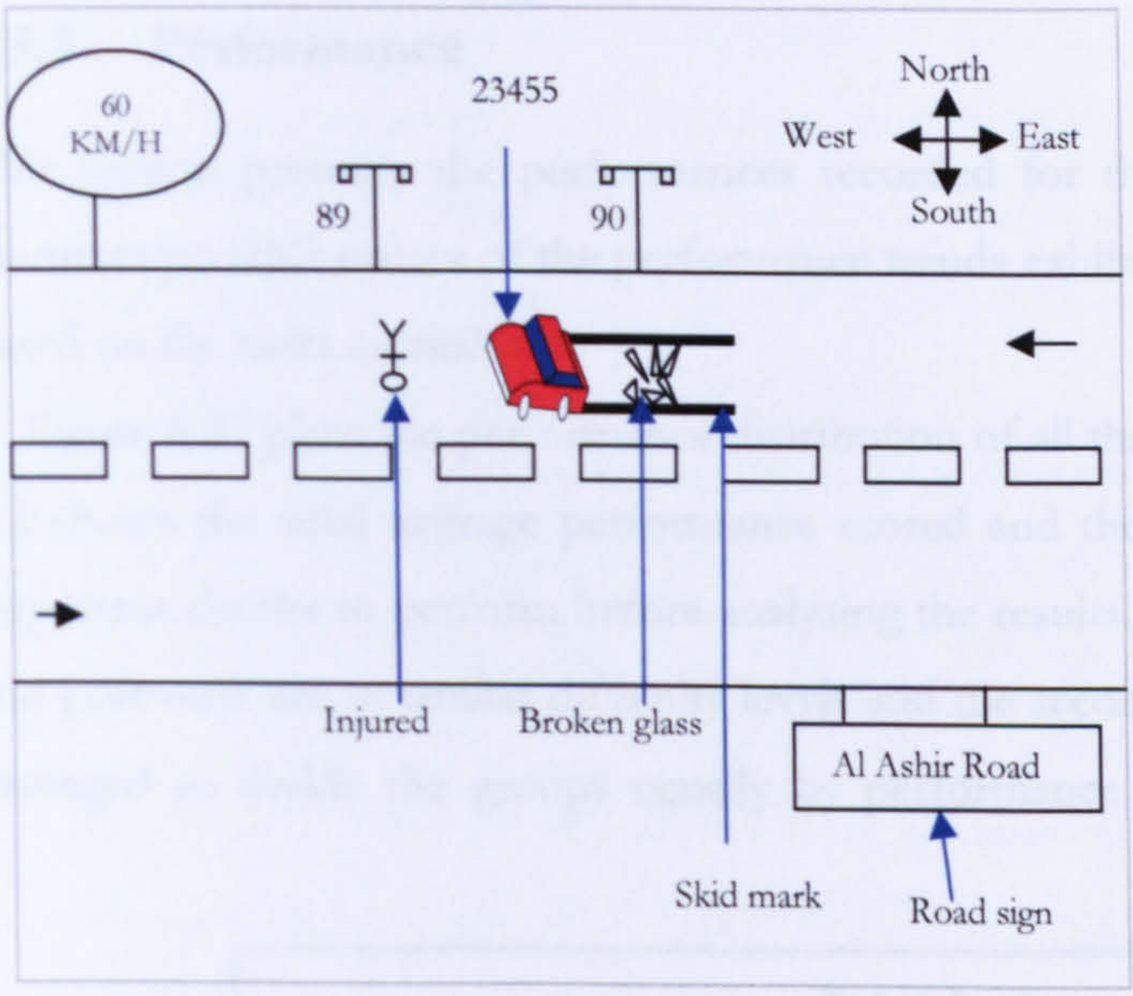


Figure 6.23: Pre-test: The drawing used in the written test (Appendix D lists the questions).

Figure 6.24: Pre-test: In the drawing test, the investigator is immersed in the environment above and then asked to draw the accident scene.

training session once more. This session proceeded in a manner similar to the first. Figure 6.25 and Figure 6.26 show the accident scenarios used for the post-test written and drawing tests respectively. The accident scenario used in the written test involves a car hit from behind while both cars were travelling on the same lane. The accident scenario used for the drawing involves a vehicle hit from the side while trying to make a left turn.

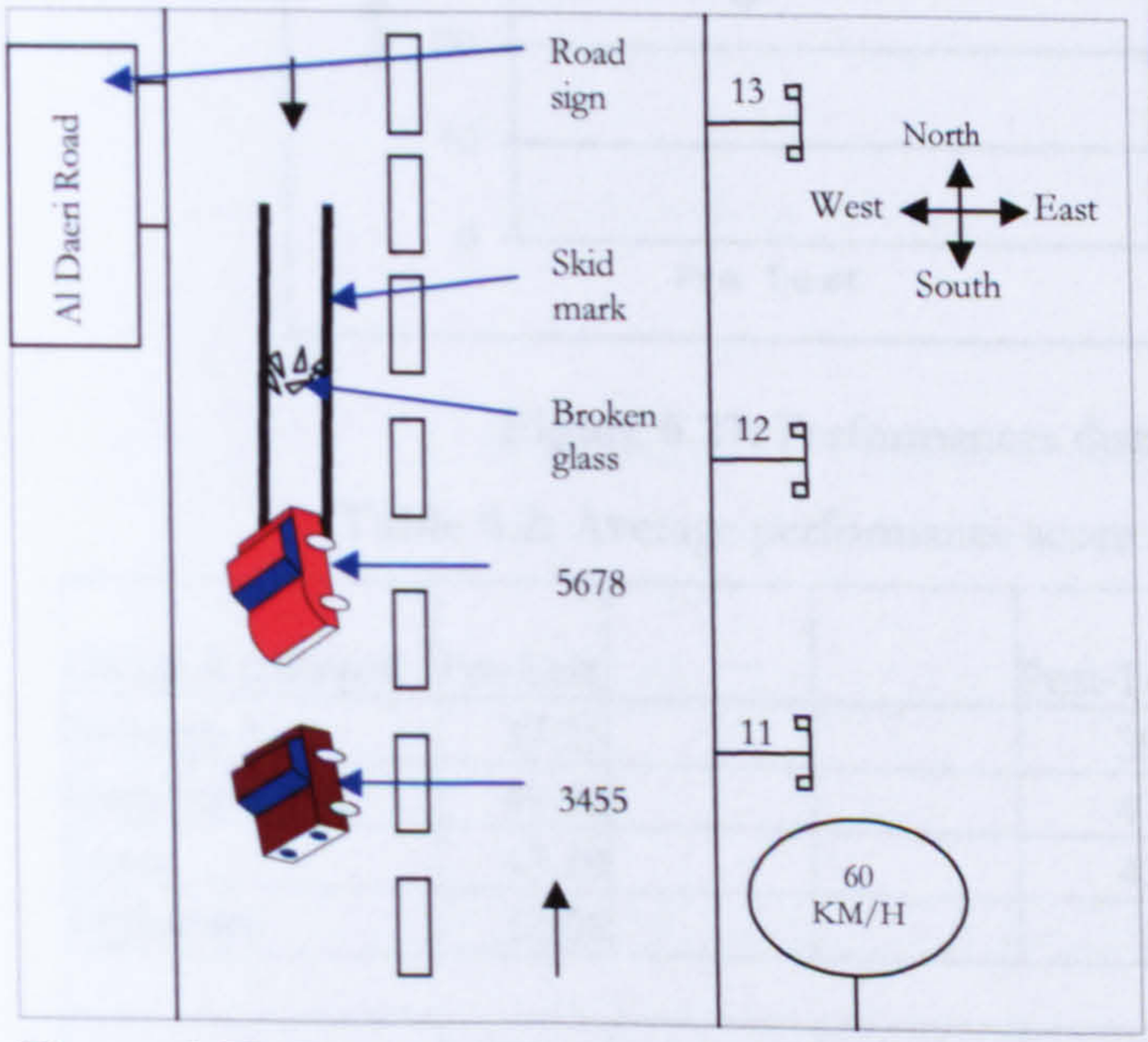


Figure 6.25: Post-test: The drawing used in the written test (Appendix D lists the questions).

Figure 6.26: Post-test: In the drawing test, the investigator is immersed in the environment above and then asked to draw the accident scene.

6.5.3 Performance

This section presents the performances recorded for the different stages of the experiment. It also discusses possible causes of the performance trends exhibited, and finally breaks down the performances based on the tasks carried out.

Figure 6.27 plots the performance distribution of all the participants for the pre- and post-tests. Table 6.2 shows the total average performance scored and the improvement that occurred. There are two important checks to perform before analysing the results. The first check needs to confirm that the pre- and post-tests are of similar difficulty levels and the second must examine that the grouping process has managed to divide the groups equally by performance. To verify the first check there is a need to

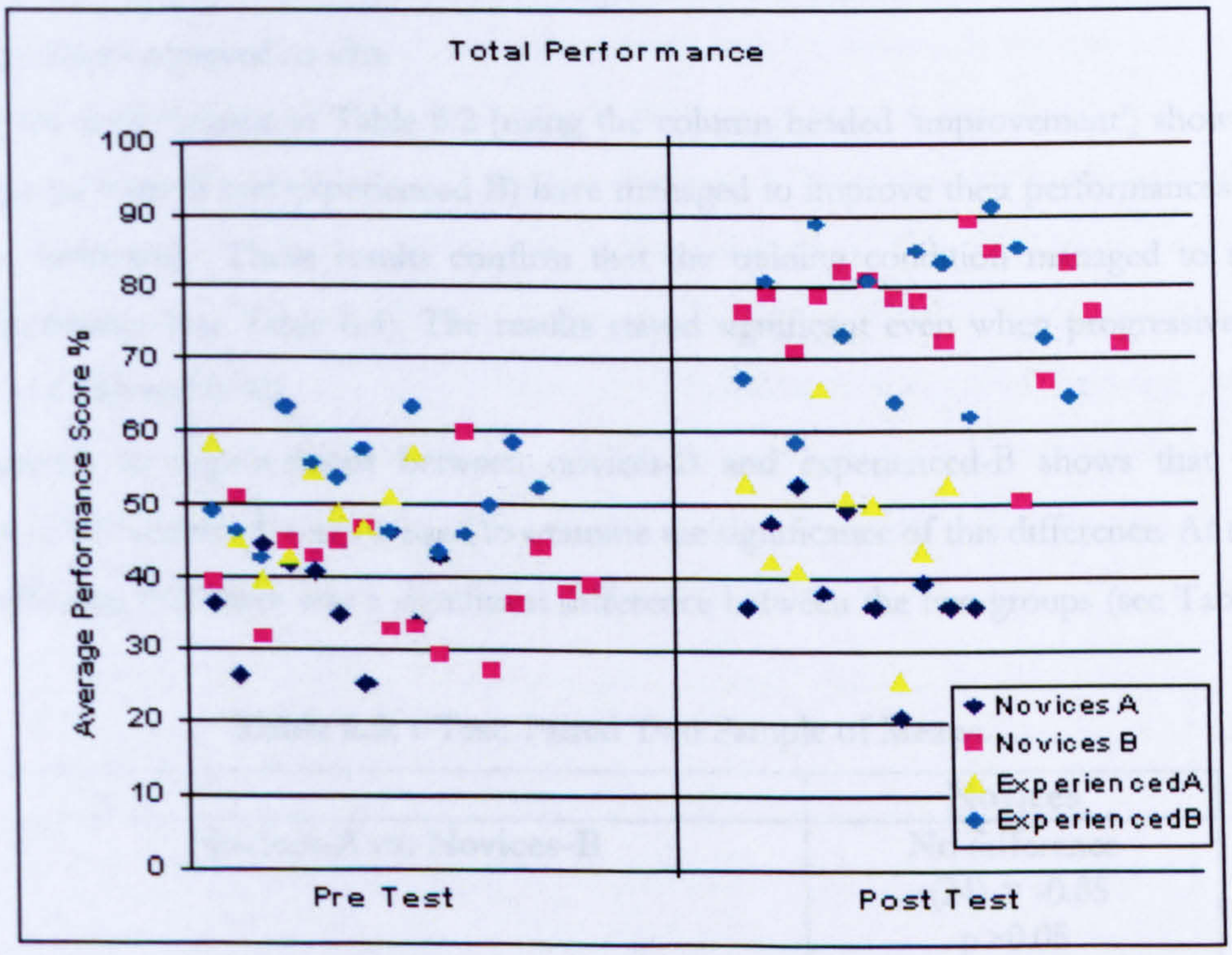


Figure 6.27: Performances distribution for both groups.

Table 6.2: Average performance score and improvement in percentage.

Group A (control)	Pre-Test			Post-Test	Improvement	Largest Improvement	Smallest Improvement
Novices-A	37.25			39.07	1.82	21.29	-7.15
Experienced-A	49.33			47.34	-1.99	23.09	6.41
Mean	43.29			43.21	-0.09	-0.09	-0.09
Difference	12.08			8.27	3.81	1.8	13.56
Group B (trained)	Pre-Test	Training 1	Training 2	Post-Test	Improvement	Largest Improvement	Smallest Improvement
Novices-B	40.04	30.97	76.11	76.21	36.17	59.61	23.52
Experienced-B	51.86	36.01	67.38	75.4	23.54	45.13	6.41
Mean	45.95	33.49	71.75	75.81	29.86	52.37	14.97
Difference	11.82	5.04	8.73	0.81	12.63	14.48	17.11

examine the differences in the average performances in pre- and post-test levels for the control groups (novices-A and experienced-A). The average performance differences in novices-A and experienced-A are 2.79% and 2.53% respectively. The t-test[7] results shown in Table 6.4 confirmed that there is no significant difference between the pre- and post-tests for both novices and experienced investigators which suggests that the difficulty level is similar.

The second check verifies the grouping process which had to divide the novices and experienced investigators into groups of equal level of performance. By comparing the pre-test results for novices-A and novices-B it was found that they scored 37.25% and 40.04% respectively for a difference of 2.79%. The t-test confirmed that there is no statistically significant difference between the pre-tests for novices-A and novices-B (see Table 6.3). Similarly there was no significant difference between experienced-A and experienced-B who scored 49.33% and 51.86% respectively for a difference of 2.53%. This shows that the grouping process achieved its aim.

Analysing the performance in Table 6.2 (using the column headed 'improvement') shows that both trained groups (novices-B and experienced-B) have managed to improve their performances by 36.17% and 23.54% respectively. These results confirm that the training condition managed to significantly improve performance (see Table 6.4). The results stayed significant even when progressively reducing alpha value[8] to 0.005 and 0.001.

The difference in improvement between novices-B and experienced-B shows that the former improved by 12.63% more. A t-test is used to examine the significance of this difference. At the pre-tests the t-test confirmed that there was a significant difference between the two groups (see Table 6.3). If at

Table 6.3: t-Test: Paired Two Sample of Means.

	Novices
Novices-A vs. Novices-B	No difference $t(24) = -0.85$ $p>0.05$ t critical two-tail= 2.06
Experienced-A vs. Experienced-B	No difference $t(21) = -0.89$ $p>0.05$ t critical two-tail= 2.08
Novices-B vs. Experienced-B (Pre-Test)	**Difference** $t(28) = -4.13$ $p<0.05$ t critical two-tail= 2.05
Novices-B vs. Experienced-B (Post-Test)	No difference $t(28) = 0.23$ $p>0.05$ t critical two-tail= 2.05

[7] A t-test is used to test hypotheses and is effective at quantifying the difference between groups (Bourg, 2006).
[8] "Alpha represents the level of significance related to the probability" (Bourg, 2006).

the post-test this gap still exists (i.e. more than the t critical) then it can be concluded that there is no significant difference, and vice versa. The t-test confirms that the significant difference found at the pre-test was nullified at the post-tests which implies that the improvement is statistically significant.

Table 6.4: t-Test: Paired Two Sample of Means.

	Novices	Experienced
A	No difference t(9) = 0.65 p>0.05 t critical two-tail= 2.26	No difference t(8) = -0.52 p>0.05 t critical two-tail= 2.31
B	**Difference** t(15) = 17.01 p<0.05 t critical two-tail= 2.13	**Difference** t(13) = 7.88 p<0.05 t critical two-tail = 2.16

6.5.3.1 Trends Analysis

Figure 6.28 and Table 6.5 show the learning trends of all the groups. The lines for novices-A and experienced-A seem to exhibit a very small improvement which verifies that the pre- and post-tests are of equal difficulty levels. However the two B groups show an interesting line of one drop and two rises. The drop occurs between the pre-test and the first training session where novices-B and experienced-B dropped by 9.07% and 15.85% respectively for an average drop of 12.46%. This is then followed by sharp rises between training 1 and training 2. Novices-B rose by 45.14% and experienced-B rose by 31.37%. The average rise is 38.26%. The second small rises recorded occurred between training 2 and post-test where novices-B rose by 0.1% and experienced-B rose by 8.02%. The t-tests show that the first drops and the first rises for both groups are statistically significant whereas the second rises are not (see Table 6.5). The causes of the exhibited trends are now discussed in greater detail.

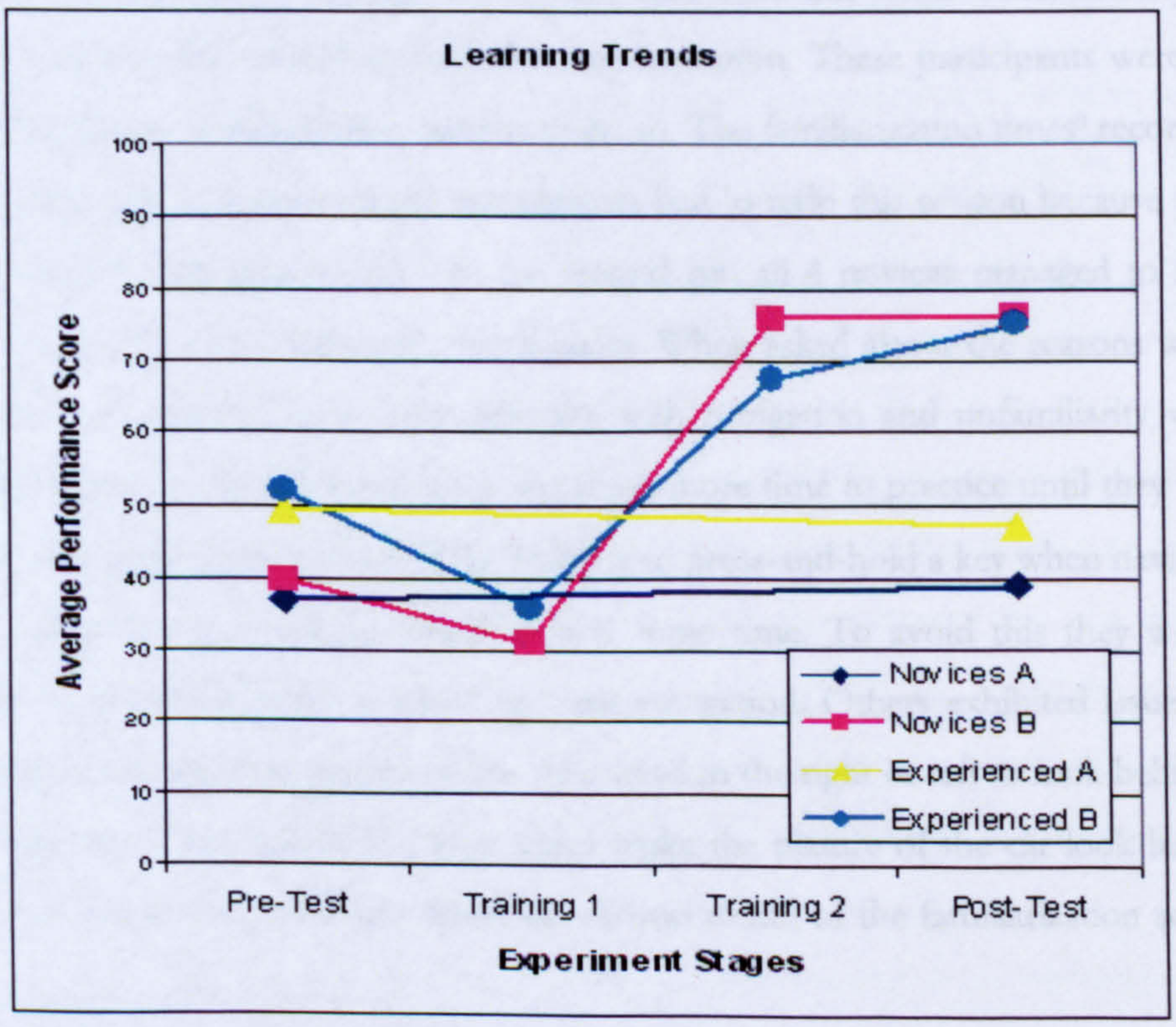


Figure 6.28: Learning trends.

Table 6.5: Learning trends.

	Novices	Experienced
Small drops between pre-test and training 1	**Difference** 9.07% t(15) = 3.74 p<0.05 t critical two-tail= 2.13	**Difference** 15.85% t(13) = 3.95 p<0.05 t critical two-tail= 2.16
Sharp rises between training 1 and 2	**Difference** 45.14% t(15) = -14.29 p<0.05 t critical two-tail= 2.13	**Difference** 31.37% t(13) = -7.77 p<0.05 t critical two-tail = 2.16
Average rises between training 2 and post-test	No difference 0.1% t(15) = -0.03 p>0.05 t critical two-tail= 2.13	No difference 8.02% t(13) = -2.159 p>0.05 t critical two-tail = 2.16

Sharp Drops

The relatively sharp drop occurred between the pre-test and the first training session (training 1). There are no definitive answers of why this occurred. However a number of factors might have contributed to it. The first factor is the unfamiliarity with the environment despite the efforts made in the familiarization stage to avoid this problem. The expectation was that the steps taken of supplying participants with a written tutorial and video demonstration days before the second session, and giving each a maximum of 30 minutes training, would have addressed this issue. Unfortunately some of the participants did not read the tutorial or view the demonstration. These participants were instructed to take some time before the familiarization session to do so. The familiarization times[9] recorded show that only 4 novices compared to 8 experienced investigators had to redo this session because they could not complete the tasks the first time round. On the second run all 4 novices managed to finish on time compared to only 4 of the 8 experienced investigators. When asked about the reasons why they could not finish on time the issues raised were difficulty with navigation and unfamiliarity with using 3D technology. These issues were addressed by giving them more time to practice until they were happy to move to the training stage. Some exhibited the inability to press-and-hold a key when navigating. Instead they opted for repetitive fast clicking which wasted some time. To avoid this they were specifically instructed to try to press-and-hold to speed up their navigation. Others exhibited issues with the 3D technology by their body movement (e.g. tilting their head to the right or left to look behind things) and by the type of questions they asked (e.g. how can I make the picture of the car look larger). Of the 4 experienced investigators that could not finish the second round of the familiarization session on time,

[9] The times for two participants were not recorded because the application was closed prematurely.

only 2 needed the full-allowed time to complete training session 1. This suggests that the effect of unfamiliarity on the sharp drop could only have affected 2 experienced participants and it did not contribute to the drop in novices performance.

The second factor is the insufficiency of time allocated for the training session compared to that allocated to the pre-test. The recorded average times to complete training session 1 for novices and experienced investigators were 25:32 minutes and 27:08 minutes respectively. Eleven novices (i.e. 68.8% of all novices) compared to eight experienced investigators (i.e. 57.1% of all experienced investigators) managed to finish the training session before time was up. These findings suggest that the probability of this factor being the cause of the drop is higher in the experienced investigators than it is in the novices.

The third factor is the varying difficulty levels of the accident scenarios, i.e. the training scenario is harder than the pre-test scenario. Since the performance achieved by the control groups confirmed that there is no statistically significant difference between the pre- and post-tests, then proving that there is no significant difference between performances achieved by the trained groups for training 2 and the post-test should suggest a reduction of the effect of this factor. This was confirmed by the t-tests which showed that the difference is not significant. The experienced participants are more likely to be affected by this factor when comparing their average to the novices.

Table 6.6 summaries the effects of the three factors on the novices and experienced investigators. For the three factors, the effect on the experienced investigators is higher than on the novices. From the three factors, it is believed that time was the most influential, followed by unfamiliarity and difficulty level. Time might have added pressure as it was mentioned as one of the causes for one experienced participant to drop out, as he felt pressurized. A physiological sensing device might have provided a stronger sense of whether or not pressure was a factor. However, further work would need to done to determine if pressure was linked to time or other experiment settings.

Table 6.6: Factors effect probability on both groups.

Factors	Novices	Experienced
Training time shortage	Lower	Higher
Unfamiliarity	Lower	Higher
Varying difficulty levels	Lower	Higher

First Sharp Rises

This section examines the causes for the sharp rises that occurred between training 1 and 2 (45.14% for novices and 31.37% for experienced investigators). The t-tests described in this section confirmed that both rises are statistically significant. The unfamiliarity factor that affected the drop might have also contributed inversely to this rise since each participant had another 30 minutes working with the system which accounts for a complete familiarization session with two runs. But since the unfamiliarity only affected 2 experienced participants during the sharp drop, this suggests that its effect is very small if not negligible.

The other two factors (time and difficulty levels) that affected the drop are unchanged – the training session time remained 30 minutes and the accident scenario used is the same. The novices' average time

for training 2 compared to training 1 increased from 25:32 minutes to 30:04 and the experienced investigators' time increased from 27:08 to 31:06. Since the same scenario for both training sessions was used, students might have memorized what needs to be done for this specific accident scenario and just applied that. In games terminology they might have learned to beat the game. However, if this was the case, it would not explain why their average performance stayed high at the post-test which uses a different accident scenario. This suggests that there is another factor that is influencing this sharp rise and it is believed to be training. The results are statistically significant and the comments were very positive which points towards this factor.

6.5.3.2 Task-based Performance

Table 6.7 shows the breakdown of performance by tasks. The objective is to examine if there are any indicators that can suggest the suitability of this training for some tasks more than others. The best average improvement for both groups (novices and experienced) occurred for the photographing task (45.08%) and the worst improvement recorded was for parking the police car. When examining each group individually it can be seen that photographing is the highest in the experienced but it comes fourth in novices. T-tests confirmed that the significant improvements for novices occurred in all tasks except parking the police car. Experienced investigators, however, only showed significant improvements in: measuring, marking, and photographing.

It was not surprising to see the photographing task top the best improved task for the experienced investigator, because of what was found during the field study (see BinSubaih et al., 2005a). In this earlier study, despite the fact that the investigator is expected to accompany and instruct the photographer to the important clues that need to be photographed, it became a habit with a number of investigators to allow the photographer to wander alone and take the photographs that he judged appropriate. The problem with this is that the photographer is not aware of the sequence of actions that led to the accident and thus cannot determine the clues that need to be photographed. One possible explanation is because of time constraints. It could also be attributed to the culture of collaboration which fosters an element of trust between the investigator and the photographer as they have most likely previously worked together on a number of occasions. As one experienced investigator revealed in the debriefing session, they initially accompanied and instructed the photographer, but with time this became a lower

Table 6.7: Task-based performance improvements in percentage.

Task	Novices-B	Experienced-B	Average
Measuring	52.83	31.83	42.33
Marking	47.1	32.75	39.93
Photographing	43.36	46.8	45.08
Placing Cones	57.5	14.3	35.9
Parking Police Car	3.73	0	1.87
Drawing Accident Scene	21.58	10.25	15.92

priority. This might provide an explanation for why the photographing task was the most improved task for experienced investigators, as they were forced to do it themselves.

6.5.4 Accident Scene Navigation

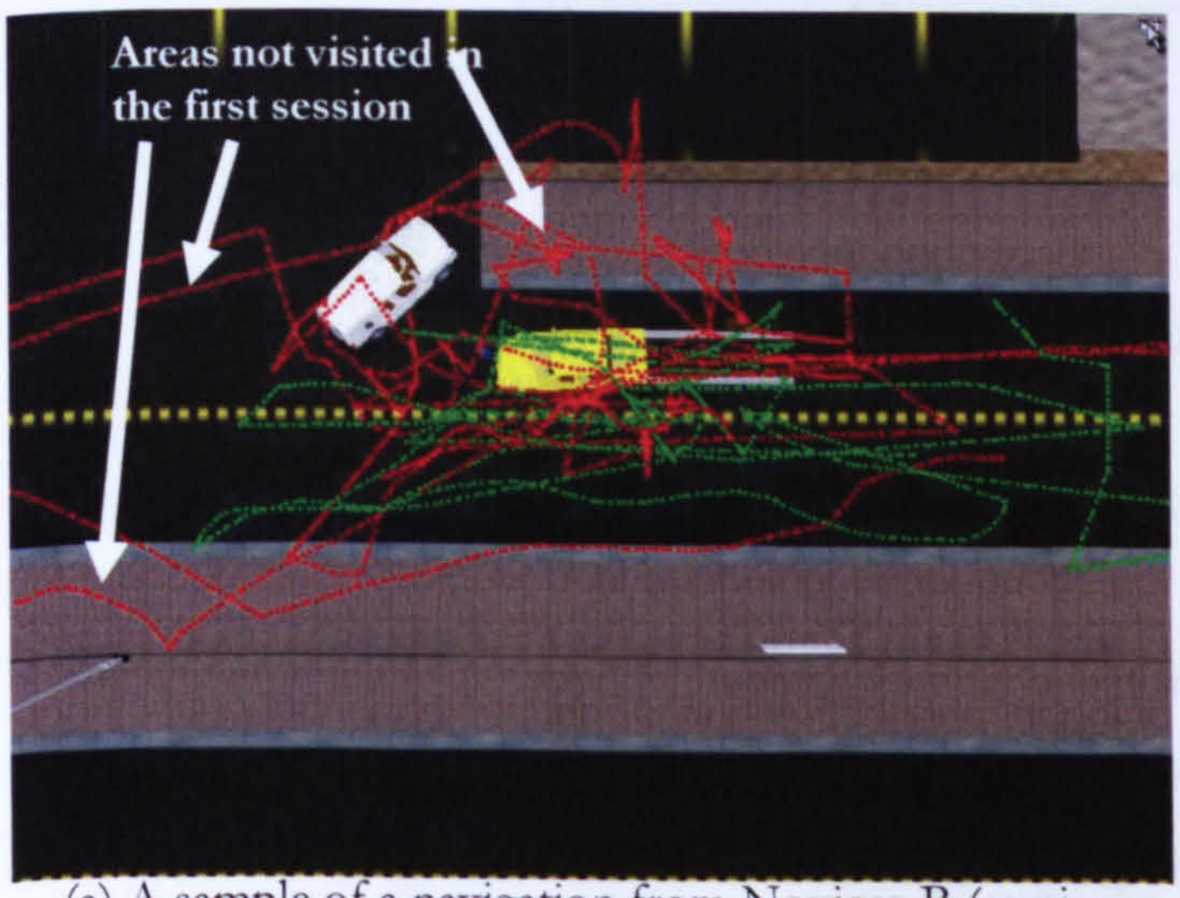
Figure 6.29 shows some of the navigational paths taken by the investigators. The two navigational patterns examined further were: distances travelled and time spent in motion. Figure 6.30 and Figure 6.31 illustrate the distances travelled by novices and experienced investigators respectively. Both navigational patterns examined showed significant difference between training 1 and training 2 (see Table 6.8). Both novices and experienced investigators spent more time and travelled more in the second training session compared to the first. This could be caused by the increase in the number of tasks completed in the second session. However time and travelled distance failed to show any statistically significant difference when comparing novices to experienced investigators.

The navigational paths also exhibited another two interesting patterns with regards to the size of the area covered and the unusual places visited. For instance, some navigational behaviour showed an increase in the area covered between the two sessions. Figure 6.29a shows that during the second

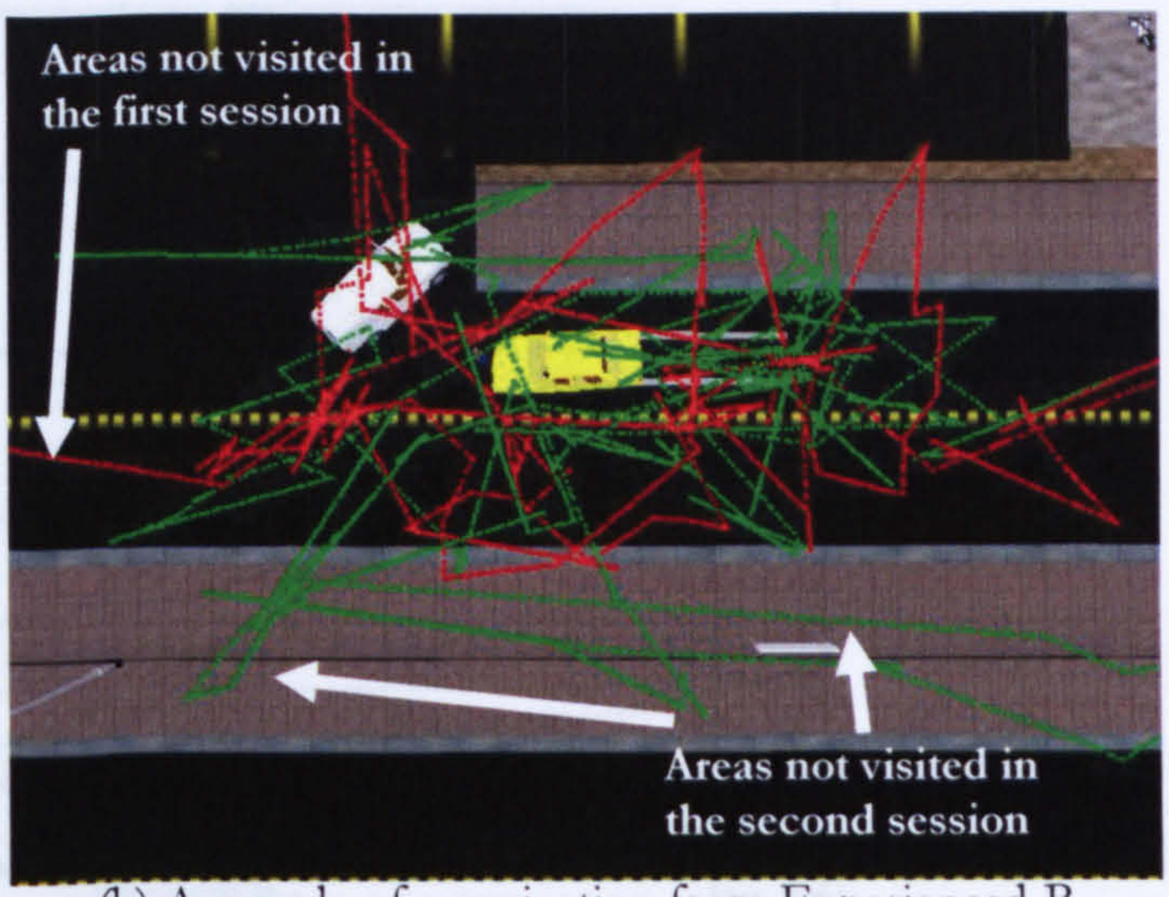
Table 6.8: The significance of the navigational patterns.

Group	**Training 1**	**Training 2**	*t-test between training 1 and 2 for each group*
Distance travelled (meters)			
Novices-B	332.91 (SD 190.73)	560.01 (SD 164.74)	**Difference** t(15) = -5.495 p<0.05 t critical two-tail=2.13
Experienced-B	380.34 (SD 163.24)	585.72 (SD 163.94)	**Difference** t(13) = -3.396 p<0.05 t critical two-tail= 2.16
t-test between novices-B and experienced-B for each training session	No difference t(28) = 0.726 p>0.05 t critical two-tail= 2.048	No difference t(28) = 0.427 p>0.05 t critical two-tail= 2.048	
Time in motion (minutes)			
Novices-B	5:26	8:08	**Difference** t(15) = -5.78 p<0.05 t critical two-tail=2.13
Experienced-B	6:30	9:14	**Difference** t(13) = -3.59 p<0.05 t critical two-tail= 2.16
t-test between novices-B and experienced-B for each training session	No difference t(28) = 1.35 p>0.05 t critical two-tail= 2.048	No difference t(28) = 0.427 p>0.05 t critical two-tail= 2.048	

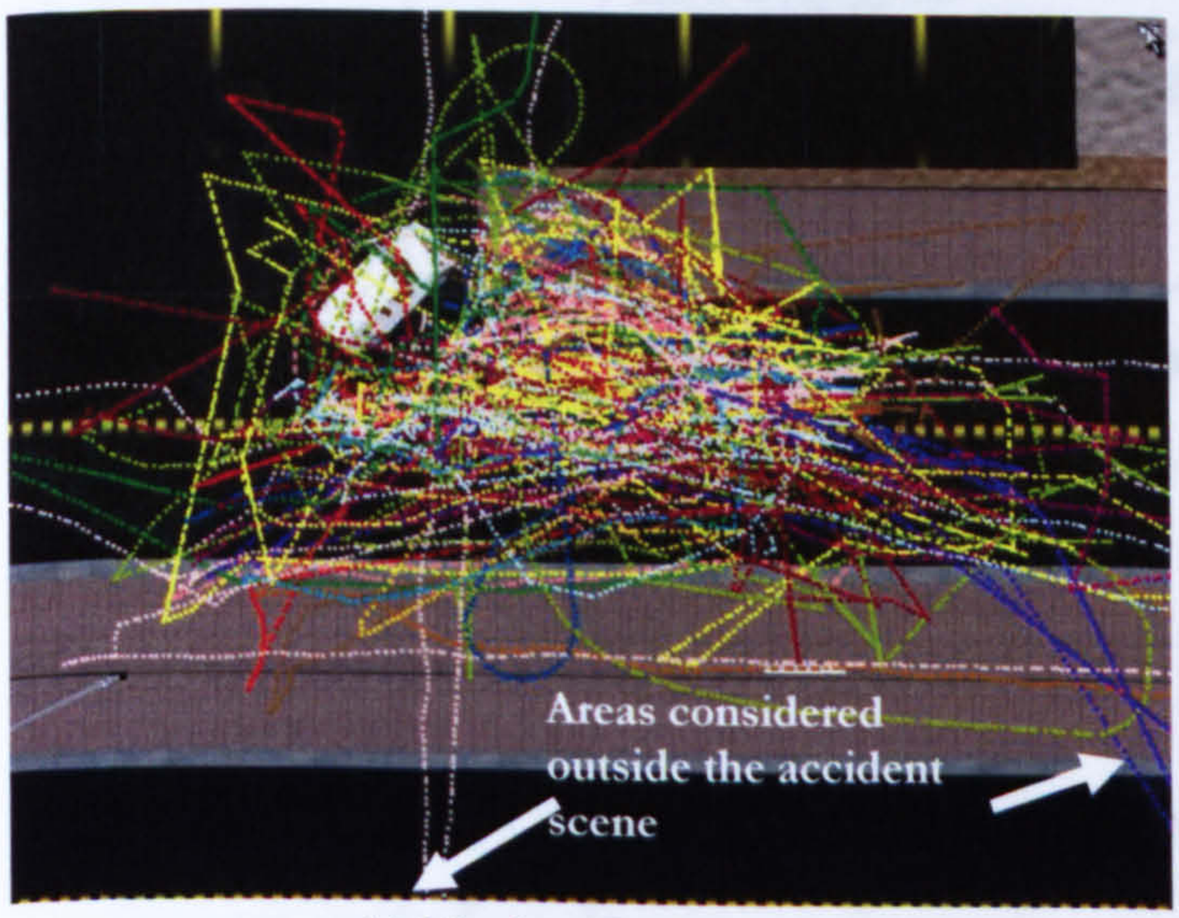
training session the investigator visited new areas in the second session. Similarly, Figure 6.29b shows new areas visited but also shows that the first session covered more areas compared to the second session. Some investigators also seem to visit areas that might be considered well outside the accident scene area as shown in Figures 6.29c, d, e, and f. During testing of SGTAI, one investigator commented that he always tries to scan areas that are considered by other investigators to be outside the immediate accident scene location because evidence can exist well outside the accident area, such as tyre marks especially in sandy areas where marks can easily be spotted. Further investigation of the causes of these patterns is required. One possible avenue to explore is to divide the accident scene into zones or hot spots and compare the differences between the investigators in terms of the visited zones and the frequency of the visits.



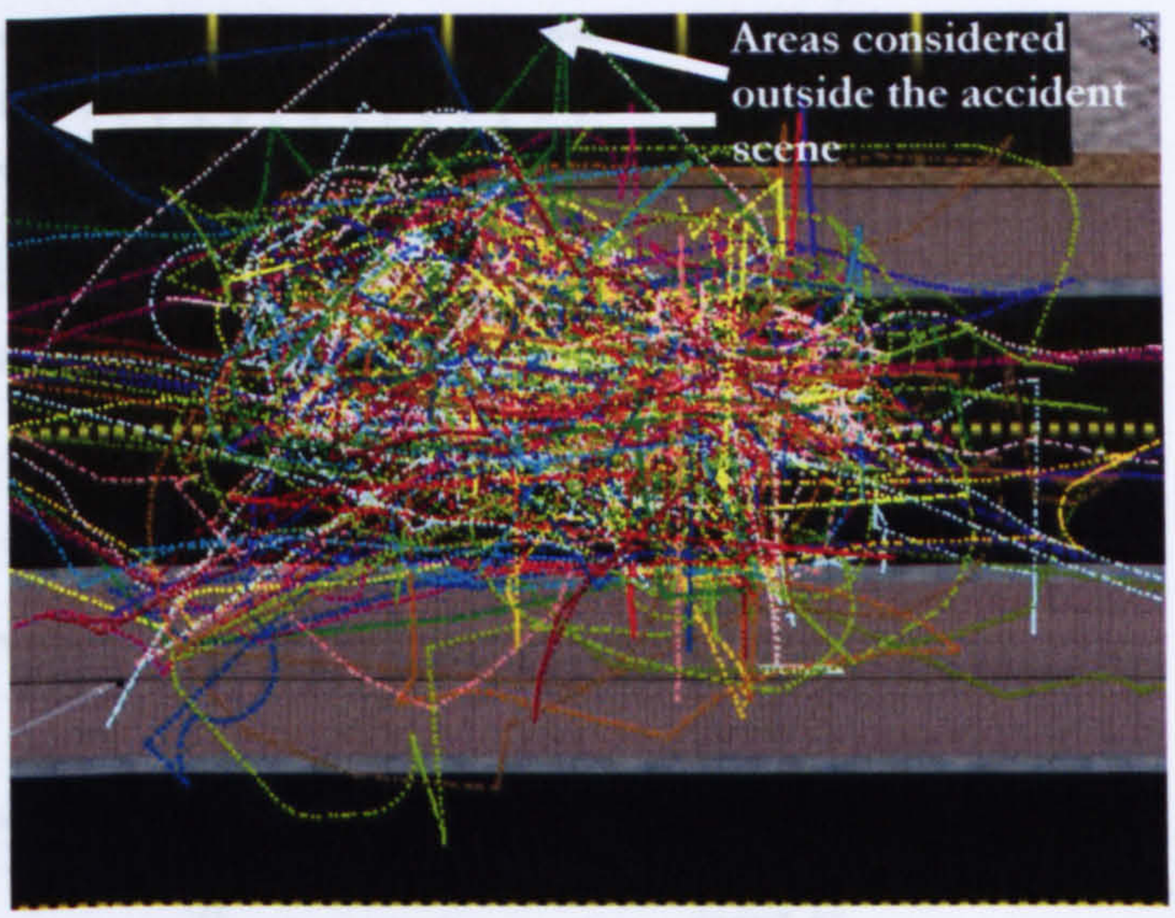
(a) A sample of a navigation from Novices-B (session 1 (green) and session 2 (red))



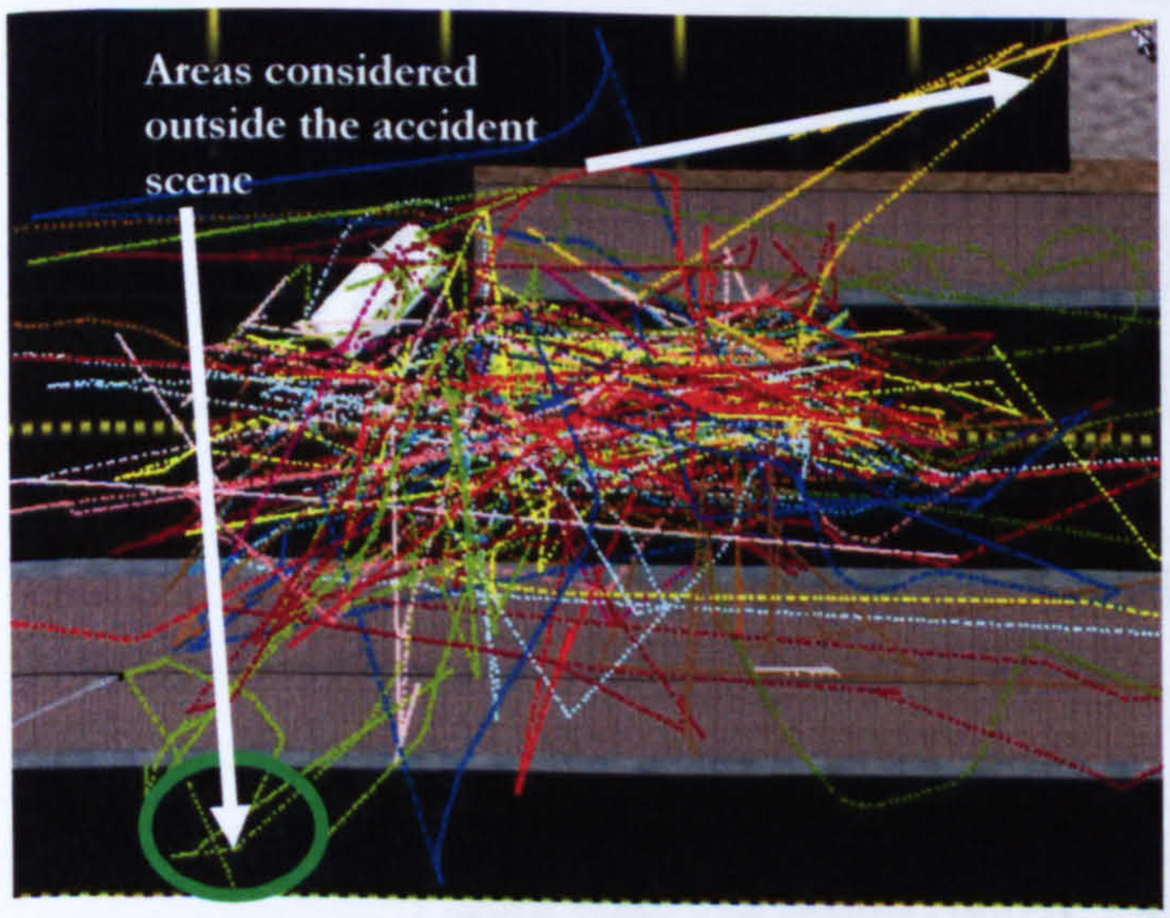
(b) A sample of a navigation from Experienced-B (session 1 (green) and session 2 (red))



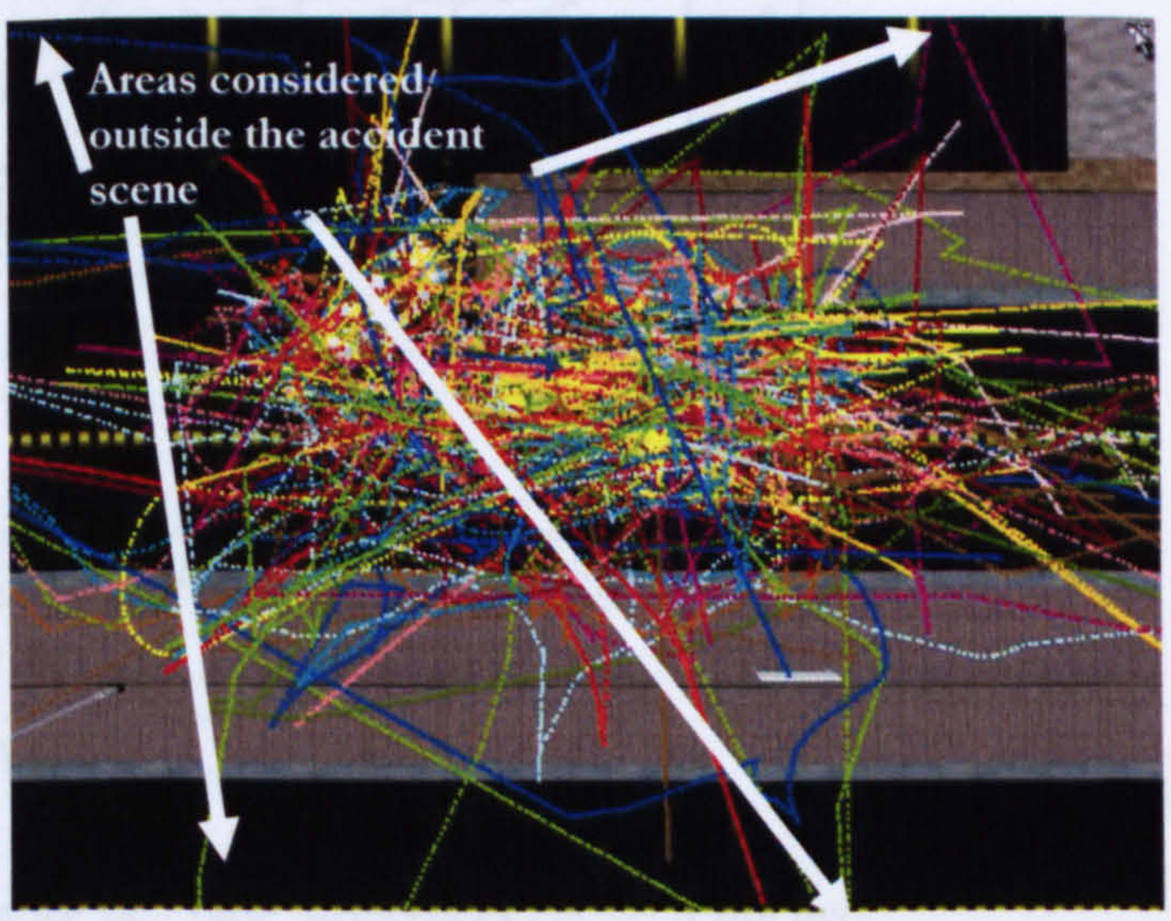
(c) Novices-B session 1



(d) Novices-B session 2

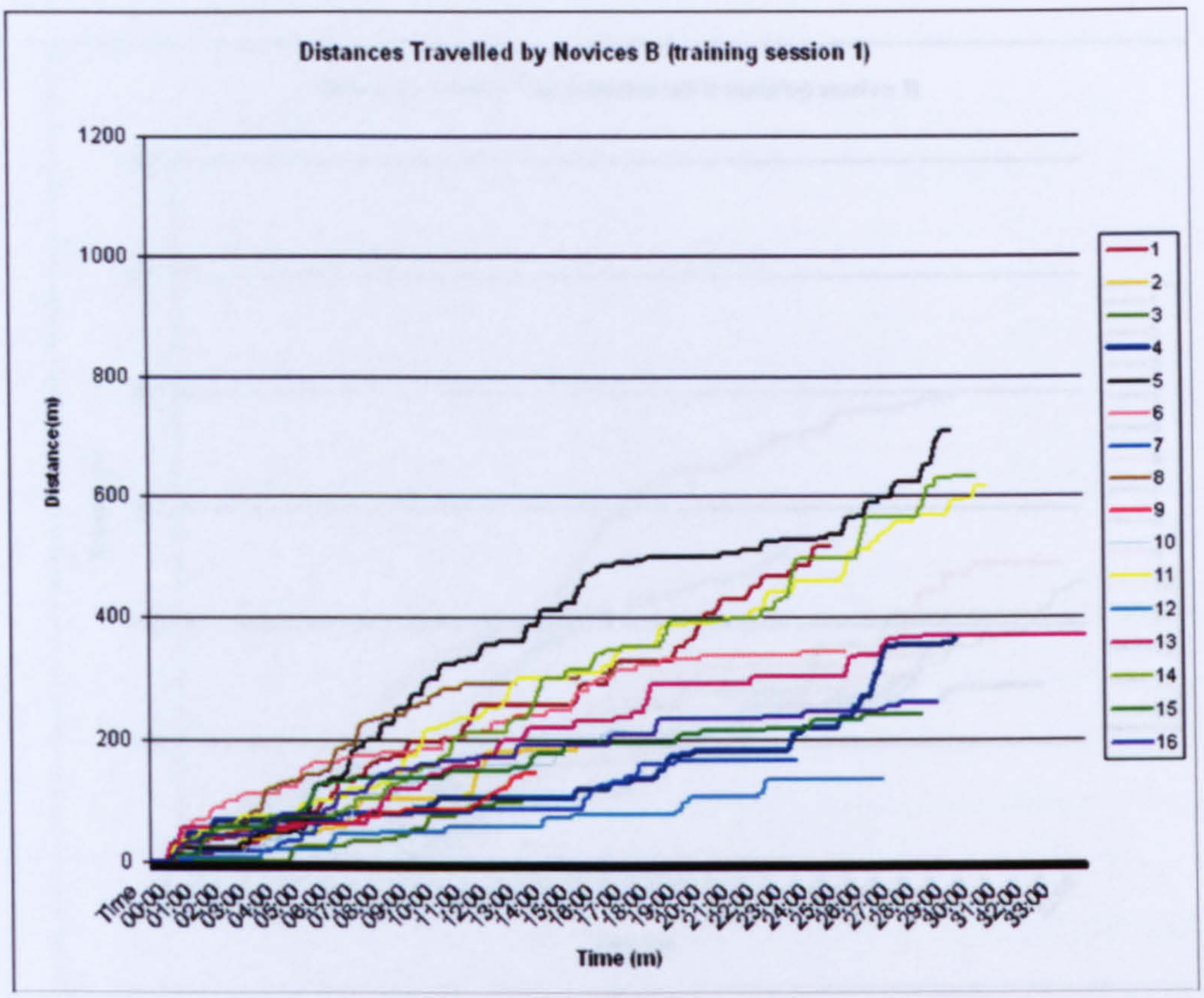


(e) Experienced-B session 1

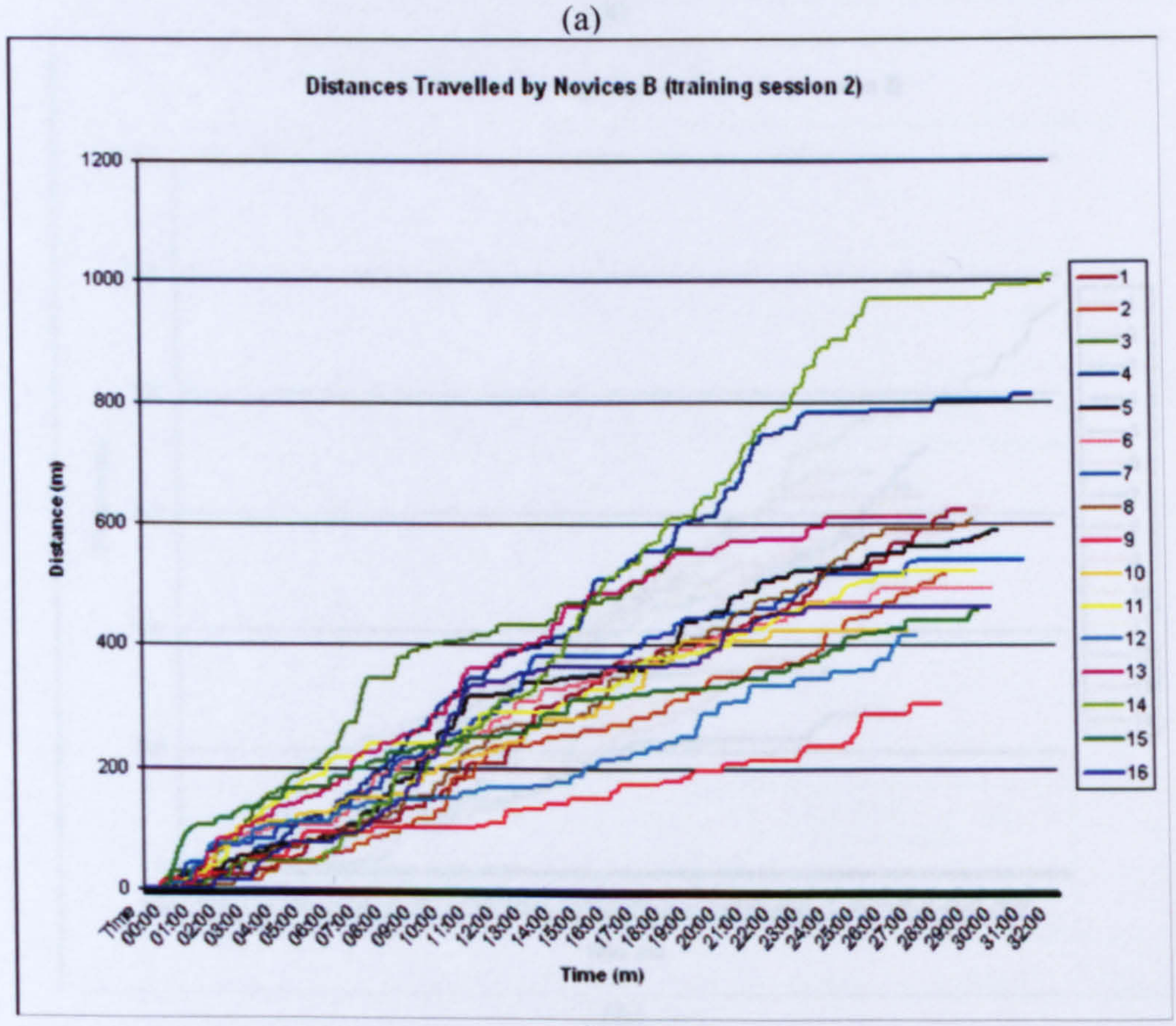


(f) Experienced-B session 2

Figure 6.29: Navigational patterns.

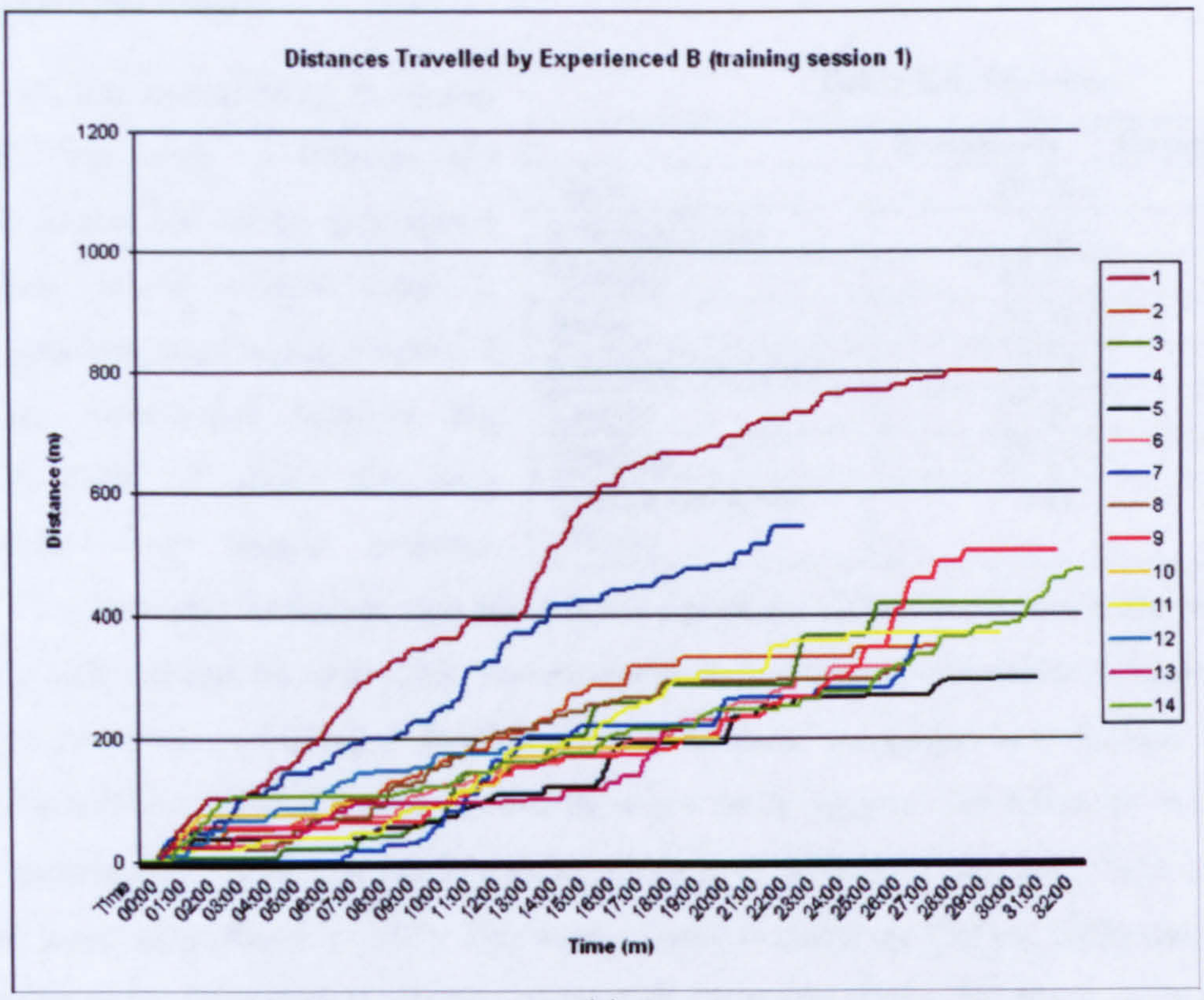


(a)

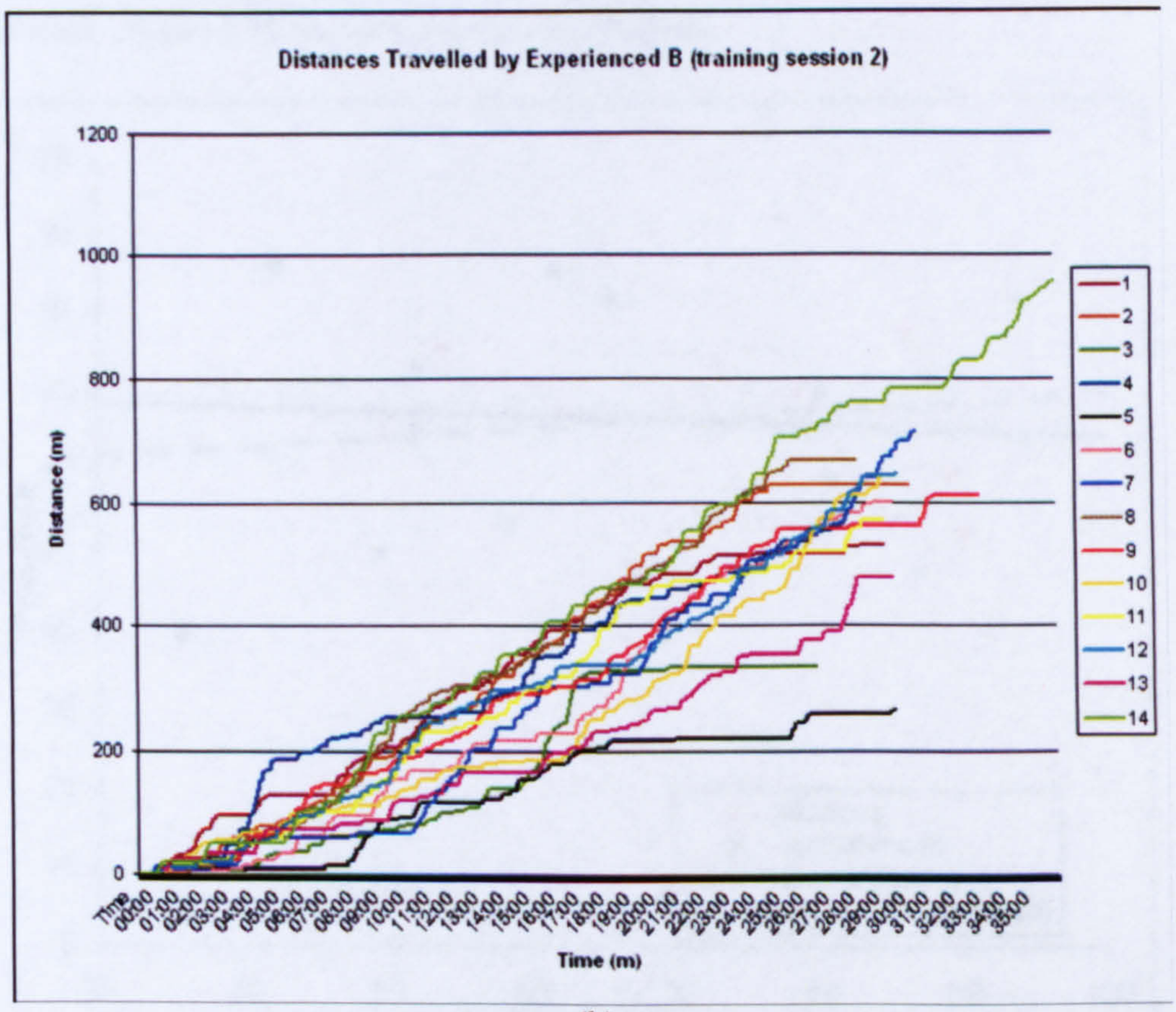


(b)

Figure 6.30: The graphs contrast the distance and time differences between the two training sessions for novices.



(a)



(b)

Figure 6.31: The graphs contrast the distance and time differences between the two training sessions for experienced investigators.

6.5.5 Sense of Presence

A presence questionnaire similar to the one in (Slater, 1999) was used to measure the subjective experience felt by the participants of 'being there' in the accident scene. It contains 23 questions with scores between 1 to 7 and one open-ended question (see Appendix D). Table 6.9 shows that both groups recorded very similar presence averages[10]. The t-test also confirmed that there is no significant difference between the two groups ($t(28)=-0.68$, $p>0.5$, t critical two-tail=2.05). To examine if there exists any correlation between presence felt and the performance achieved, a Pearson product-moment correlation test (Bourg, 2006) was performed. The test revealed that for the novices there is a small negative correlation ($r=-0.16$) and for experienced investigators there is a small positive correlation ($r=0.15$). Combining both groups the correlation becomes insignificant ($r=0.07$). This interpretation is based on (Cohen, 1988) who suggested that, for correlations in psychological research, a correlation between 0.10 and 0.29 (or –0.29 and –0.10) is considered small. Figure 6.32 shows the correlation diagram.

Table 6.9: Presence.

	Novices-B	Experienced-B
Mean	66.06	66.08
Standard Error	2.01	3.99
Median	67.71	67.43
Mode	75.71	81.43
Standard Deviation	7.78	14.92
Largest	77.57	85.71
Smallest	52.14	39.14
Confidence Level (95.0%)	4.31	8.66

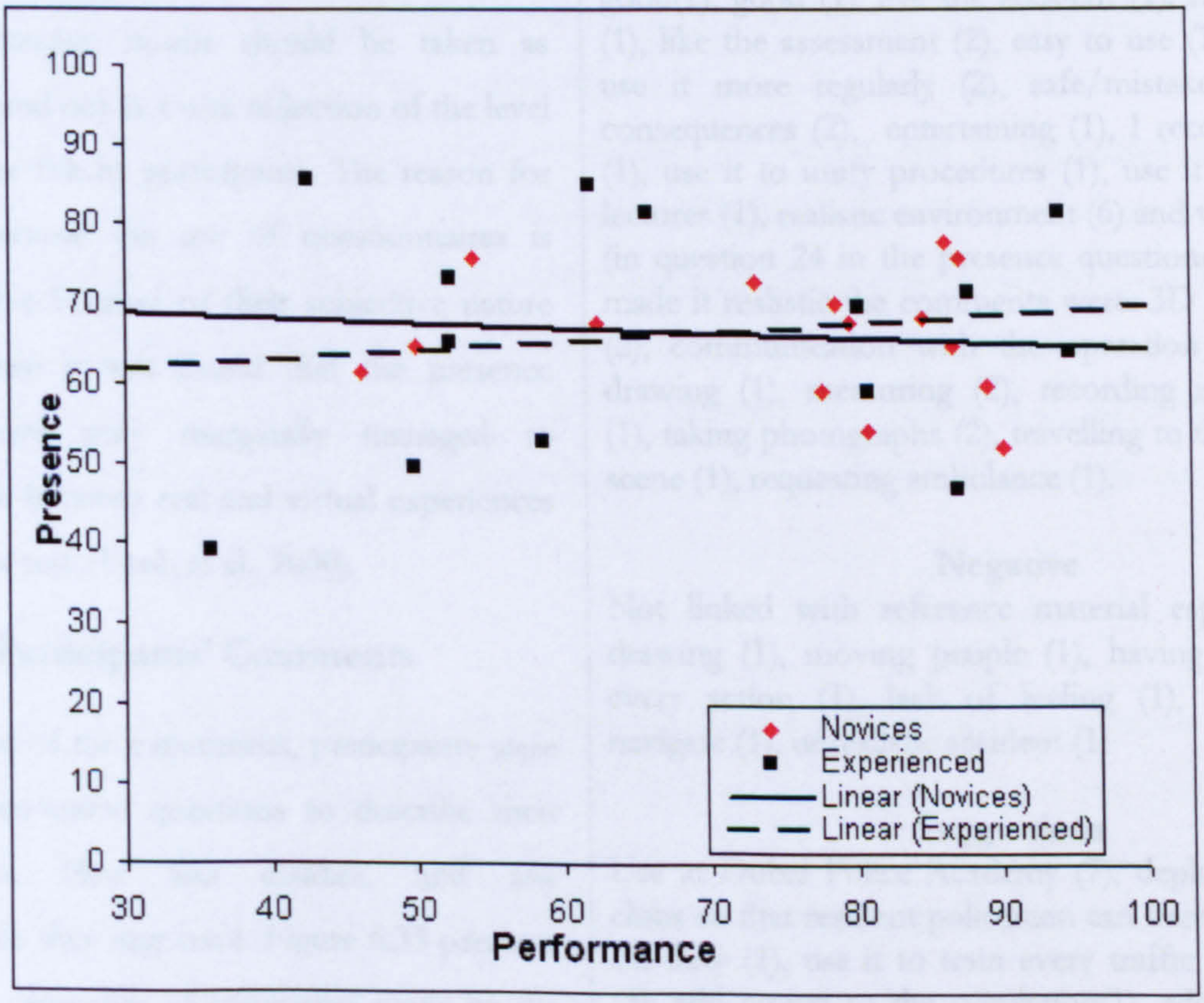


Figure 6.32: The presence and performance correlation.

[10] One novice participant failed to return the presence questionnaire.

The correlation found between presence and performance was small and not what was believed would happen. The belief was that novices (younger generation) would be more at ease with this type of environment and would feel more presence compared to the older participants. However this was not the case as the novices showed negative correlation compared to positive correlation shown by the experienced investigators. The exhibited positive correlation is much smaller than the one reported in VERTS (Youngblut & Huie, 2003) which was at 0.42. One explanation for this could be the use of dialogue boxes to ask the student to reflect after each action to justify what he did (e.g. after a measurement is taken the system asks why he performed that measurement). The reflection box may have broken the sense of presence.

One explanation of the fact that the experienced investigator's sense of presence increases with performance, instead of decreasing as for the novices, could be that the nature of their expertise is based on the 'expert recording set of schema', which guides their problem-solving, and which novices do not have (Chi et al., 1988). It seems that experts use the stimuli from the simulation as a trigger for previous memories and have a more "internal" experience thus paying less attention to the presence-breaking stimuli (e.g. reflection boxes). Novices may have a more "external" experience due to their lack of experience and pay more attention to the presence-breaking stimuli.

The presence results should be taken as indicative and not as a true reflection of the level of presence felt by participants. The reason for that is because the use of questionnaires is questionable because of their subjective nature and because it was found that the presence questionnaire only marginally managed to distinguish between real and virtual experiences in a 'reality test' (Usoh et al., 2000).

6.5.6 Participants' Comments

At the end of the experiment, participants were asked open-ended questions to describe their experience, likes and dislikes, and any suggestions they may have. Figure 6.33 presents the three categories of comments made by the participants who were trained: positive comments, negative comments, and suggestions.

Comments (Number of mentions)

Positive

Engaging (1), practical (4), semi-realistic (1), saves time (1), useful (6), teaches (10), excellent (9), very good(1), good (1), live the accident (1), no pressure (1), like the assessment (2), easy to use (1), want to use it more regularly (2), safe/mistakes without consequences (2), entertaining (1), I recommend it (1), use it to unify procedures (1), use it to replace lectures (1), realistic environment (6) and when asked (in question 24 in the presence questionnaire) what made it realistic the comments were: 3D technology (2), communication with the operation room (2), drawing (1), measuring (2), recording information (1), taking photographs (2), travelling to the accident scene (1), requesting ambulance (1).

Negative

Not linked with reference material especially for drawing (1), moving people (1), having to explain every action (1), lack of feeling (1), difficult to navigate (1), unrealistic accident (1)

Suggestions

Use at Dubai Police Academy (7), deploy at Police clubs so that resident policemen can train on it while off-duty (1), use it to train every traffic investigator (3), add crowd to the simulation(1), add traffic (1), use it to train on rare accidents (1).

Figure 6.33: Comments made by participants.

The comments indicate the experiences felt by the officers who played SGTAI. The positive comments and suggestions seem to support the findings reported in section 6.5.3 which showed SGTAI to be effective. For example, in open-ended questions 10 comments were made about SGTAI's ability to teach and 6 found it useful. In addition, 9 thought it was excellent. This is also backed by the suggestions made where 7 thought it should be used in the police academy and one thought it should be deployed in the police clubs. What was surprising was the fact that 2 comments thought the communication with the operation room helped increase the realism of the environment. This is despite the fact that it is menu-based dialogue. This is probably because it contributed to the overall investigation experience despite its lack of fidelity. The dislikes varied across issues such as the use of the reflection box, a lack of feeling, moving characters, and not linking with the course material. The positive comments are reflective of the issues raised during the experiment but the negatives are not. For instance the difficulty and the reflection box were raised much more than other issues during the different sessions of the experiment but many officers did not include them when answering the open-ended question.

6.5.7 The Limitations of the Experiment

One of the limitations of this work is not verifying that training transfer would be carried out to real accidents. The study conducted has demonstrated the amount of transfer but will that be retained and, more crucially, will it be utilized in a real accident? The subjective comments from the participants in the experiment and the anecdotes[11] from people used in testing SGTAI provided a strong indication that this may be the case. However that has not been proven.

The second concern is with the varying screen sizes of the two sets of equipment used in the experiment: a desktop PC (17-inch, 512MB RAM and 32MB graphics card) and a laptop (15.4-inch, 2GB RAM and 256MB graphics card). However the difference of 1.6 inches in screen size is too small to have had any significant effects on the amount of learning transferred or the presence level felt. In literature the screen size was found to be an issue for performance (Tyndiuk et al., 2004; Patrick et al., 2000) and presence (Laarni et al., 2005) only when one screen was more than double the size of the other.

6.6 Discussion

This section discusses the findings from the experiment conducted (section 6.6.1) and uses it to highlight the issues concerning the development process of SGTAI. The issues analyzed are: fidelity (section 6.6.2), expertise (section 6.6.3), using GSA (section 6.6.4), and the ability of SGTAI to address the current training issues at the Dubai police force (section 6.6.5). Finally, the findings along with the field

[11] One tester during a discussion said that when he travelled to an accident he started taking pictures of things he noticed in SGTAI.

study are used to identify the implications this work presents for policy makers, educators, and researchers (section 6.6.6).

6.6.1 Evaluating the Two Hypotheses

The findings suggest that there is a statistically significant improvement in the performance of both novices and experienced investigators who were trained on SGTAI compared to those who were not. These findings validate the first hypothesis of the experiment. Several reasons could help explain this positive outcome.

First, it could be argued that the training sessions promoted concentration and focused participants on the investigation topic in a way that demanded attention. It is known from the learning theory literature that increased interactivity leads to increased attention which results in a deeper information processing (Wong et al., 2007). In addition, several studies have shown that video games increase attention rate (Green & Bavelier, 2003; McFarlane et al., 2002). Another study has also shown that increased attention in serious games leads to better transfer of learning (MacNamee et al., 2006).

The second reason could be attributed to SGTAI presenting participants with a challenge which motivated them to achieve better scores. One of the factors that help motivate participants in any setting is the discovery that their knowledge is incomplete (Habgood et al., 2005). The ability to repeatedly practice away from real-life constraints means longer exposure which allows participants time to develop and refine their skills. Repetition is an important learning factor which can improve performances by 30 to 110% for initial repetitions and by 15 to 45% for additional repetitions (Thalheimer, 2004). The average improvement reported for novices-B between the first training session and the second training session in this study exceeded the suggested range of performance improvements due to initial repetitions quoted by Thalheimer. The average improvement reported for experienced-B investigators between the first training session and the second training session fell within the range of performance improvements due to initial repetitions.

The third reason could be attributed to ability of the learning foundations used (see section 6.4) to ensure that motivation and engagement are not disconnected from learning. As described in section 5.4.2, intrinsic motivation is preferred over extrinsic motivation, where intrinsic motivation relies on providing the feeling of mastery. This is provided in SGTAI through the use of a scoring system which indicates the progress made and which is linked to the completed tasks which are all related to the investigation process. The other component used to keep participants engaged is to provide them with achievable goals without making the game too easy. The average largest and smallest performance improvements reported for all participants were 52% and 15% respectively. These findings show that the game was not too easy and not too hard. Providing feedback also keeps participants engaged.

The second hypothesis, which expected novices to exhibit significant improvement compared to the improvements recorded for experienced investigators, is validated to a lesser extent than the first

hypothesis by the findings. There were significant differences in performance improvements between novices and experienced investigators who were trained on SGTAI. (This was true for alpha value of 0.05 but was not the case when alpha level was reduced to 0.005 and 0.001 – see section 6.5.3. The first hypothesis withstood these reductions which increases the confidence in the results). The basis for the second hypothesis was that the environment does not represent a high difficultly level and therefore experienced investigators should be able to achieve high scores in the pre- and post-tests. Also, the difference between their improvements and the improvements recorded for novices should remain significant. A possible explanation is that the study underestimated the effect real-life constraints have on shaping the knowledge and skills of experienced investigators which pushes them into adopting shortcuts. With time these shortcuts become the norm. This was evident from the improvements reported in the photographing task for the experienced investigators (see section 6.5.3.2).

The above findings are important since they indicate the suitability of this type of technology for the personnel in the Dubai police force. This opens the door for expanding the investigation of its use into different fields. In fact a number of projects have been discussed since demonstrating SGTAI at InterSec 2006[12] (Daggash, 2006; Haidarh, 2006), such as using it for forensic science, search and rescue, hostage negotiation, and airport security. The findings also indicate that the three learning foundations selected (experiential learning principles, Aldrich's elements (Aldrich, 2005), and increasing the feedback loops) have managed to make learning an integral part of SGTAI.

Comments from participants who were trained with SGTAI indicated that it was effective. Comments from trainers indicated that SGTAI was effective at improving performance and at providing an environment that they could utilize in a classroom setting. Other studies such as Tactical Iraqi (Vilhjalmsson & Samtani, 2005) and Full Spectrum Command (FSC) (Beal & Christ, 2004) reported similar perception of learning by participants. For instance in Tactical Iraqi one participant commented that "I learned more in 1 day with this [TLTS] than I did in a whole tour in Iraq." In SGTAI, the perception of the participants' ability to learn is also clear from their comments. One participant commented that "In my opinion if everyone in the Dubai police force is trained on this [SGTAI] there is no need for lectures". Other comments showed increased interest in the subject being taught and a willingness to spend time on their own working on SGTAI. This is similar to the findings of a project that used a game to teach operations management, which found a substantial amount of increased interest in the subject (Chwif & Barretto, 2003).

However, some participants disliked the fact that they had to explain their actions via the reflection box (see Figure 6.6) and thought it was a waste of time. The inclusion of the reflection box was to allow trainers to get insights into the participant's thinking which trainers appreciated. What further aggravated the issue was the time constraints imposed on the training session. Dropping timing when using the

[12] http://www.intersecexpo.com/

reflection box can help reduce the frustration but since it is unclear how the interruptions affected the participants' sequence of thoughts, the recommendation is not to use them in the future. Instead, trainers should refer to the recordings when in doubt of participants' actions during the after action review session. Other participants struggled with the use of the technology at the start, especially those that were not used to 3D technology. The difficulties included navigation and controlling objects and characters. Similar issues were reported by subjects who were used to evaluate Ambush! (Diller et al., 2005). Fortunately, these initial shortcomings in SGTAI were soon overcome with the support of additional time spent on allowing participants to get used to the technology (see section 6.5.3.1 for practical measures taken to achieve that). Another factor that limited the learning in SGTAI was not to provide a mechanism for participants to access the course material. This is a missed learning opportunity that could have facilitated uniform feedback. Currently SGTAI provides learners with model answers of what should be accomplished and leaves it up to student to find out why such action is necessary from the trainer or by referring to other resources.

6.6.2 Fidelity

The comments made by participants on fidelity were mixed (see Figure 6.33). Some thought the environment felt like a real environment and others raised issues with regards to characters lacking realism and the environment needing traffic and crowds. The variability in the comments made supports the arguments made by Presnky about the difficulty in finding the right answer for fidelity (Prensky, 2004) as it is subjective and sometimes contradictory. The fidelity design for SGTAI mainly focused on functional fidelity and the findings from the performances for individual investigative tasks (e.g. photographing and measuring (see section 6.5.3.2)) give a positive indication of the ability of SGTAI to improve performances across these tasks. Additionally participants' comments mentioned a number of these tasks as adding to the fidelity of SGTAI which are positive indicators to achieving functional fidelity. However the effect of the abstraction (i.e. through the computer medium) on functional fidelity was not measured. For instance, would the investigators still be able to achieve similar performances without having the icons present in the GUI acting as a constant reminder of the tasks that need to be accomplished? Moreover, have some of the abstracted tasks prevented the investigator from learning (i.e. reflection boxes)?

The results from the presence questionnaire can be used as an indication of the level of psychological fidelity achieved in SGTAI. Both novice and experienced investigators reported a similar level of presence (about 66% using Slater's presence questionnaire (Slater, 1999)). The investigators' comments listed the factors that they felt increased or reduced the fidelity as a whole. Among the factors that increased fidelity were: the use of 3D technology, travelling to the accident scene, and communicating with the operation room. The factors that undermined fidelity included: lack of feeling from the

characters, moving people, navigation difficulty, and unrealistic accident. Physical fidelity[13] was preserved by two lessons learned from the preliminary experiment (BinSubaih et al., 2005a): cultural issues and familiar places. In the preliminary experiment a woman character was present at the accident scene and she was dressed in a short skirt. To the trainer's amazement this managed to deter one of the investigators from approaching the woman although she might have been a witness in the case. Here it can be seen that conservative cultural principles should be considered in a serious game. This suggests that it is probable that racial and religious issues also need to be carefully considered so as not to influence an investigator's performance. The course material used in traffic accident investigation training also warns of favouritism at the accident scene and demands that all parties should be treated equally. To minimise the possibility of these issues emerging the characters should be of similar creed and religious belief, which should be exhibited in the way they dress and speak[14].

Similar consideration also needs to be given as to whether or not the same game character should be used in different accidents. In real life, if we see somebody involved in more than one accident, we may suspect his driving skills. This could result in an investigator jumping to conclusions. SGTAI has used different game characters for the drivers for each scenario by changing their faces and textures and reusing the body mesh. It has also used different voices for the actors.

Another lesson learned was with regards to the location chosen for the accident. In the preliminary experiment the virtual street in the scenario was named after a known road. This caused problems. As the model of the street and its surroundings was not a replica of the real one any missed information was pointed out by the user. In presence terminology this means a break of presence (Brogni et al., 2003).

6.6.3 Expertise

One of the challenges and time consuming tasks of building SGTAI was collecting, verifying, and filtering expertise to find the suitable expertise that can be delivered through a serious game. This required bringing together the expertise from the fields of: traffic investigation, game design, instructional design, and game development.

The process used to collect traffic investigation expertise was indiscriminate and tried to gather as much information about the topic as possible through: reading material, discussions, interviews, role-playing sessions, elicitation sessions, travelling to accidents, attending interrogations, attending a training session, examining previous cases, and running preliminary experimentation. The reading material helped with understanding the traffic accident investigation field and acted as a reference manual to resolve issues about knowledge and procedures gathered from the field. The discussions held with police officers of different ranks were useful in identifying the complexities and difficulties facing investigators.

---

[13] The level to which the virtual environment is made to look like the real environment.

[14] We wanted to control these variables during this study but in future studies they can be varied and used to help detect favouritism and discrimination and identify how they affect the investigation process.

The discussions have also identified the issues facing current training methods employed by the Dubai police force. The interviews (unstructured and semi-structured) were used to clarify information collected, elicit rules, and discuss what a serious game can provide. The role-playing sessions proved useful in understanding what an investigation consists of and appreciate the different investigation patterns. The elicitation sessions focused on formatting the expertise into rules. Travelling to accidents helped in living the incident of what an investigator has to deal with, the variety of issues faced, and the constant interruptions from the drivers, crowd, motorists, and other police personnel. Attending interrogations showed the need for the investigator to have collected all the necessary information before confronting the drivers with his findings. Examining previous cases showed the similarities between accidents. Preliminary experimentation helped focus SGTAI (BinSubaih et al., 2005a).

Another consideration in a serious game is the use of subject matter experts (SMEs). During the collection stage two SMEs were selected to help filter the experiences and approve the final embedding of the expertise in SGTAI and the assessment used. Although the reliance on qualified SMEs is important (Beal, 2004) to identify learning objectives and instructional problems, having a first hand experience of the investigation topic is very beneficial. The time spent working in a police station and travelling to accidents revealed that there is a disconnection between what is being taught (and often what the SMEs breach) and what is actually being practiced. After further investigation and discussion it was identified that the disconnection resulted from the constraints imposed by reality which fostered the adoption of shortcuts. With time, these shortcuts became the norm and often ended up being passed on to new recruits during the on-the-job supervised training. Equipped with this knowledge SGTAI was focused to force the investigator to do all the tasks individually. This also makes SGTAI applicable when spaced appropriately over time to ensure that these shortcuts are identified and corrected. The other issue with SMEs is that they are linear experts (Aldrich, 2004) who speak about sequences and cases. In the development of Virtual Leader (Aldrich, 2004) it was found that trying to make experts think in a non-linear way is a very difficult task. With the two SMEs used in SGTAI this was apparent in the way they often cited previous cases. This is where the game design and instructional design expertise need to take linear information and convert it into dynamic simulation.

Technological limitations have also limited what can be included in SGTAI. For example the dialogue system used (see Figure 6.7) became a problem after changing the game type from being a multiplayer game in the preliminary experiment to a single player game. In the multiplayer game actors were used to play the roles of the characters. It was difficult and expensive to try to automate the dialogue system in a single player version while still maintaining the same level of dialogue freedom and fidelity. Similar issues were reported for a serious game developed to teach leadership skills (Iuppa & Borst, 2007). Therefore, for SGTAI, the decision was made not to assess investigators on this part of the investigation. In future, to include this assessment (i.e. interviewing drivers, communicating with the operation room, and collaborating with police personnel at the scene), a multiplayer version of the game should be used.

To help increase the fidelity of the dialogue system, the audio of the characters was synthesized but later during the testing was found to be unclear and the Arabic accent to be distracting (i.e. it used accents which sounded Algerian or Moroccan, which the players found amusing). Based on this the decision was made to replace the synthesized audio with actors' voices with local accents (i.e. UAE accent).

The other factor that helped identify what SGTAI should simulate is to identify what a serious game is going to add to training methods that are currently being used or could be used in the Dubai police force. If SGTAI is not going to add anything that other cheaper methods can achieve the whole purpose of it becomes questionable. This is where some of the initial time was spent. The development of an early prototype facilitated running the preliminary experiment which compared the use of a serious game against the use of a tabletop training (BinSubaih et al., 2005a). The serious game in the preliminary experiment used an open environment that required actors to participate in the session. This restricted its usage. It also did not provide the trainer with additional functionalities (such as assessment) to compete with the tabletop method. To address these shortcomings the type of serious game required was switched to a single player game and further functionalities were added to assist the trainer in evaluating students (e.g. self-evaluation, score sheet, navigational patterns, and storage of all interactions). Furthermore, it was helpful and important to look at SGTAI not only as a training tool but as part of a wider setup within the organization. Doing so revealed general issues (i.e. not only related to training) facing the Dubai police force which SGTAI can contribute to such as experience sharing. Addressing these in SGTAI should increase its appeal.

What helped during the development of SGTAI was the author being a member of the Dubai police force who could relate to the knowledge and training issues raised. For instance having gone through the training provided by the college the author was aware of the lack of practical training and managed to early on relate to the issues raised. Also being a member of the police force helped in getting unrestricted access to sources needed from accessing previous cases to interviewing and requesting officers for the experiment.

6.6.4 Development Using GSA

The decision to use GSA to develop SGTAI was made to evaluate the scalability of the architecture, although there is no reason why it could not have been developed using a typical game development approach. In fact using GSA added overhead which consumed time that could have been spent on improving the learning part of SGTAI (e.g. validating fidelity). The overhead was mainly in developing the communication between the game space and the game engine. This has also complicated the modding ability of SGTAI. Using a typical game development approach the modder needs only to work with a single language (TorqueScript) but when using GSA he has to learn two languages and needs to be able to create the adapter. However the creation of the adapter is a one time setup and after that

scenarios can be created as long as the adapter handles the necessary communication. This is demonstrated by the three traffic accident scenarios developed (see Figure C.3). Also the use of techniques such as dynamic object model has somewhat compensated by simplifying part of the development.

6.6.5 The Ability of SGTAI to Address Current Training Issues

Besides improving performance, there are other indications that suggest the potential suitability of SGTAI to address the problems with the two training methods – lectures and on-the-job training – employed by the Dubai police force. The issues facing the use of lectures are (see section 5.3): exam-focused teaching, lack of hands-on practice, class size and time constraint, and lack of motivation and engagement. The exam-focused teaching could be attributed to the fact that students are only tested using theoretical examinations which leads them to focus on the topics that are going to be in the exams. These exams often measure the students' ability to memorize facts, but the students' ability to apply the knowledge remains questionable. Serious games can provide a platform for students to put what they have learned into practice, which can help them to refocus on the whole investigation topic rather than what is going to be in the exam. Additionally a game often forces students to take an active role which provides hands-on practice.

The issues of the class size and the time constraints were raised during interviews conducted with officers of different ranks. These issues have limited the types of accidents students are exposed to and limited the feedback they receive during lectures. SGTAI can address these constraints since students can use the game in their own time. SGTAI is also capable of running different accident scenarios to suit the different kinds of accident types the trainers feel necessary to expose students to, but due to time constraints are unable to. Furthermore, SGTAI is well suited for providing the immediate feedback that is lacking from lectures and which is key to retention and understanding.

SGTAI also logs the participants' interactions, which a trainer can examine and use to provide further feedback. This logging ability can be used to analyse data in ways that is impractical to do in lectures or field training. For example, the navigational behaviours of participants and the way they prioritize tasks at a scene are easier to record and analyse in a serious game. As an example, the navigational pattern could reveal that an investigator had strayed into an unsafe area, e.g. into the opposite lane of a highway, thus putting himself at risk. Another potential use for the logging ability of SGTAI is to use it as a platform for sharing the experiences of an ageing workforce. The environment records users' missions for after-action review. This data can be used to share experience. The last issue regarding the use of lectures is that of motivation and engagement. The ability of SGTAI to motivate and engage investigators has been discussed in section 6.6.1.

The issues facing the use of on-the-job training by the Dubai police force are: impracticality, varying levels of exposure, and lack of uniform assessment. The impracticality issue was raised due to the lack of

repeatability and exploration. SGTAI allows students to practice as many times as they feel necessary to improve their skills. Since they can practice on their own they can explore different options without fear of failure or embarrassment. The issue of varying levels of exposure is addressed by the ability to create different accident scenarios. The issue of the lack of uniform assessment is addressed by using performance metrics, which provide a more systematic and fair assessment system.

6.6.6 Implications and Future Research Directions

This study has received positive feedback from students, educators, and policy makers. The comments described earlier show that both students and educators found SGTAI to be practical and effective. Policy makers found SGTAI to be innovative. Students, educators, and policy makers also pointed towards improvements required and other fields within the Dubai police force that could make use of this technology. This suggests that serious games have a potential in becoming one of the training methods utilized by the Dubai police force. It is important to point out that the openness to change (especially technology-driven change) is partly due to the current push in the Dubai government to become an electronic government. The Dubai eGovernment project began in 2001 with the aim of converting 90% of all services to electronic services by the end of 2007[15]. In November, 2006, Dubai police announced that it had managed to reach 88% and Dubai Municipality had managed to achieve 90%[16]. These are positive indicators towards technology tolerance.

The implications for policy makers concern the use of serious games for training and for sharing experiences. As the number of examples demonstrating the ability of serious games to deliver on their objectives increases, combined with digital natives demanding change, the police domain would find it difficult not to follow suit with other domains that have become "true believers" in the use of this technology. The use of serious games represent a viable option that not only appeals to the new generation of police recruits, but has shown its ability to address a number of issues facing current training methods at the Dubai police force (see section 6.6.5). During discussions the author held with police officers of different ranks, the issue novice investigators raised was the lack of practical training environments, and the issue experienced investigators raised was the lack of training provided to help them improve their skills and keep up-to-date with advances in the traffic investigation field. SGTAI can address both issues. It is practical and has been developed as a standalone environment. This means it can be used to provide experienced investigators with on-demand learning. Policy makers also know that these issues are not limited to the traffic investigation field but can be found across many other fields in the police domain. From this study, and judging by the requests received for such environments, it

---

[15] http://www.dubaipolice.gov.ae/dp/e_services.jsp?Page=A4&Id=857366261&ArticalType=1 (accessed 4/1/2007)
[16] http://www.ameinfo.com/102168.html (accessed 4/1/2007)

shows that forensic science investigation, search and rescue, hostage negotiation, and airport security are some of these fields.

In addition, serious games have a greater potential compared to the video-based simulations which currently dominate the domain of police training (Bennell & Jones, 2003) because serious games are easier to modify (or mod). Modding is a powerful tool for digital natives who thrive on social interaction (Herz & Macedonia, 2002) and many studies have shown it to be effective in the serious games domain (Fong, 2004). Furthermore, modding has a role to play in building an infrastructure for sharing experiences. It has been shown that one of the factors that pushes people to develop their skills is to get peer acknowledgement (Herz & Macedonia, 2002). This means that policy makers would have to provide an infrastructure capable of supporting such activities. They also need to ensure that educators are available to monitor such environments to verify the experiences shared and to ensure that the shortcuts that currently undermine on-the-job training are identified and corrected. Policy makers should also consider providing incentives for investigators to share their experiences. A similar scheme currently exists in the Dubai police force to encourage suggestions, the Suggestion Program, which began in 1998. The program has three objectives. The first objective is to unleash the talents and innovative powers of human resources. The second objective is to get acquainted with the views of the public. The third objective is to ensure the continuous improvement of performance. It works based on points and there are rewards for people with the most implemented suggestions. They are given titles such as "Knight of Suggestions" and "King of Suggestions". A similar system to encourage investigators to exchange experiences and also to become modders by developing accidents scenarios would help create a continuous learning environment.

The main implication for educators is that they must understand that the current on-the-job practical training environment is not delivering what is expected of it. The causes for that have been highlighted in section 6.2.1. The cost of not having an unconstrained practical training environment is evident from the relatively low results of the pre-test, which averaged 39% and 51% respectively for novices and experienced investigators. This requires educators from the on-the-job training and the ones at the Dubai Police Academy to come together to identify the responsibilities, the shortcomings of the current investigator training, and possible solutions to address them. A serious game can only achieve so much and can only deliver on the learning objectives set for it. Therefore it should be part of a larger solution, and should not be seen as the only solution for a lack of practice. The ideal role for it is to bridge the gap between lectures and on-the-job training by easing learners into an intense, unsafe, and unpredictable real-life situation. Educators also need to break a serious game into chunks that can be delivered in the period of a classroom. They should also ensure, when using serious games for on-the-job training, that it is spaced appropriately over time to prevent the issue of shortcuts becoming part of the investigation process. In addition, educators must be prepared to deal with students who are not video game players and understand the difficulty they are going to face, especially at the start with the navigation and control

issues. To do this it helps if educators themselves try to become gamers to better understand these issues.

Although the assumption made earlier (see section 6.4.1) was that learners enter SGTAI already having gone through the traffic investigation material, there is no reason why SGTAI cannot act as pre course training material. The benefits of this would be to give the learners understanding of the vocabulary used, tasks they have to do, the people they have to interact with, and the marking scheme. America's Army is a good example of a serious game that has been used to inform potential candidates about life in the army before joining and it has been shown to be effective as a training tool (Zyda, 2005). Another example is Microsoft Flight Simulator which has been described as the most successful use of commercial games for training - in the US Navy all student pilots and undergraduates receive a customized version of the software (Herz & Macedonia, 2002). A study conducted by the US Navy showed that students who used the game during early flight training received higher scores than those who did not.

The implication of this study for researchers concerns the use of instructional design when developing a serious game. The debate of whether or not there is a need to use instructional design is ongoing. From this study's perspective, instructional design helped in breaking SGTAI into manageable blocks, which helped focus the design process. At the start of the development of SGTAI, the vast number of instructional design models available made it difficult to know what to choose. This was, and still is, hampered by the lack of practical demonstrations of how effective or ineffective instructional design is when used alongside game design. As noted earlier, this has forced some researchers to try to reverse engineer serious games to identify what principles were used. The study in this chapter provide researchers with a practical demonstration of using instructional design to integrate the learning objectives in SGTAI, thus improving on the scant knowledge obtained by reverse engineering of existing serious games. Towards the latter stages of this thesis a number of instructional design methods targeted for serious games emerged (e.g. CRAFTE (Charsky, 2006)). However these are new and their abilities need to be further investigated. A possible future research direction is to verify if SGTAI includes the principles suggested by these instructional design methods.

7. Conclusions and Future Work

A new architecture, called game space architecture (GSA), has been developed to address the first aim of this thesis work. This architecture enables different game engines to be 'plugged in', thus separating the game from the game engine. The second aim of the thesis has also been fulfilled by using GSA to develop a serious game (SGTAI) that has been used to train traffic accident investigators in the Dubai police force. SGTAI addresses the issues facing the training methods currently employed by the Dubai police force, which consist mainly of lectures and on-the-job training. Using SGTAI, both novice and experienced personnel improved their ability to deal with traffic accidents.

Over the years, game development has evolved to support different aspects of portability (e.g. assets, components, and operating systems). Unfortunately, this support has not reached the stage where a game can be easily ported between game engines, hence the existence of "the RenderWare Problem" (Carless, 2007). The best support for G-factor portability so far is provided by AI architectures. These architectures emerged to deal with the increase in AI complexity in games and provide ways for setting the G-factor. The level of portability supported varies and so does the way the AI component is linked to the game engine. However these architectures exhibit similar issues to the ones exhibited by game engines such as using proprietary formats and making the G-factor dependent on the whole architecture. The next step needs to remove this dependency and also needs to provide a modularized approach that identifies the decisions that aid G-factor portability thus allowing for a progressive move towards portability (see section 2.5.3). The need for a solution to the G-factor portability problem is further highlighted by the latest announcement about Torque game engine, which was used in the development of SGTAI. In September 2007, InterActiveCorp (IAC) announced it had acquired a majority stake in GarageGames (Fritz, 2007), the developer of the Torque game engine. Despite assurances by both companies that this does not mean the end of the Torque engine, there are worries in the games community – similarities with Atmosphere, which was bought to be enhanced, only to be discontinued later[1], have been suggested.

This thesis has examined the causes of the G-factor portability problem through a review of game engines and projects that have used game engines (see chapter 2). The review found that the proprietary object model provided to create the classes (e.g. players and non-player characters (NPCs)) and the proprietary scripting languages provided to set the game logic are the main contributing factors to tying these two elements (object model and game logic) to a particular game engine. It was also found that some game engines (e.g. Half-Life 2, Jupiter EX, and 3Impact) still prefer the hard-coded practices which are inflexible compared to data-driven practices (see section 2.3.1). Moreover, the tendency to

[1] http://www.garagegames.com/mg/forums/result.thread.php?qt=67342 (6/10/2007).

specify the game state in a way that restricts access to it from external modules was an issue raised by projects that used game engines but needed to use complex AI behaviour (Fielding et al., 2004).

The solution provided by GSA used a client-server approach where the G-factor elements can exist independently of the game engine. This was achieved by hosting the G-factor elements in a self-contained service (i.e. the game space) and using an adapter to communicate with the game engine. In order to address the proprietary object model issue, GSA supports the creation of a dynamic object model. This allows the object model to exist in a database independently of the architecture and the game engine, which makes it easier to port compared to an object model that is hard-coded in the game engine (see section 2.3.2). To address the language problem an existing off-the-shelf language was used (Jython). Using an existing language is also recommended since it provides rich features and often better documentation (Tong, 2003; Bilas, 2000). Furthermore, the languages supported by the open-source community lessen the concern about their future security (e.g. discontinuation).

However the solution of separating the G-factor from the game engine, and using a dynamic object model and scripting languages, has an adverse effect on performance. The challenge is how to keep the game playable. Servicing the G-factor elements from the game space to the game engine at a real-time rate is network intensive and the decision made was that its effects on performance would make the game unplayable. The compromise reached was to replicate part of the G-factor elements that require real-time processing in the game engine. This was achieved by separating the game objects into two types. The first type were the game objects that have to have representations inside the game engine to provide visual representations such as the Player and the NPCs needed for the Moody NPCs game (see section 3.5). These required real-time processing in the game engine and it was impractical to communicate every frame from the game space to the game engine. Therefore these objects had to be created in the game engine as well as the game space and only updates were communicated. The second type of game objects were the ones that did not have representations inside the game engine such as the Action, Interaction, and Reaction objects for the Moody NPCs game. These objects could be created in the game space only. The object model creation was similarly split over the game engine and the game space. This replication of work adds implementation overhead and makes it harder to implement as the developer has to be familiar with languages provided by both the game space and the game engine. Nevertheless, the benefits of having a portable G-factor are worthwhile.

To assess GSA's success in achieving its objectives (i.e. portability, modifiability, and performance), both structured and unstructured evaluation processes were conducted (see Chapter 4). The findings from the unstructured evaluation process showed that GSA was capable of making the G-factor portable, but it added performance and implementation overheads. Despite these overheads, the unstructured evaluation process demonstrated that GSA was able to scale to real world applications. The fact that different G-factors (e.g. Moody NPCs, First-person shooter game, and SGTAI) could be

developed using GSA showed its modifiability. The structured evaluation used the architecture trade-off analysis method (ATAM). Using ATAM revealed that in the event that a single unique identifier cannot be set for game objects in the game space and game engine, GSA becomes very sensitive to any modification as it has to be added manually in the adapter. Furthermore, using on-the-fly scripting allowed for better modifiability but ran slower than pre-compiled code. Modifiability was also enhanced by the use of a variance of the model view controller (MVC) pattern which reduced the dependencies between the model and the view.

Contrasting ATAM's output to the unstructured evaluation results, which quite often answer the challenge with yes or no, or with some metrics such as network load or fps, highlights the strengths of ATAM. ATAM classified the decisions according to how they affect the architecture (i.e. support or undermine it). The ATAM process was helpful in understanding the architecture better. A similar finding was reported by (Lattanze, 2001). Of further benefit was that it should also act as a guide when there is a need to modify or evolve GSA. This guidance is based on the fact that it reveals the strengths and weaknesses of the architectural decisions. The ability of ATAM to provide directed guidance was also reported by (Nord, 2001). In future, the recommendation is to use ATAM alongside the development cycle. This is where ATAM is designed to be most effective by revealing issues at different stages of the development cycle when they are cheaper to address. Had the evaluation process started with ATAM, it would have saved the time and effort spent in creating a number of redundant challenges. For instance, the implementation of SGTAI used in the scalability challenge could have been used to test all the other challenges. Although structured evaluation cannot guarantee that redundancy would not occur, the guided approach has a better chance at identifying redundant challenges.

Despite making the G-factor elements for SGTAI portable via GSA there is still more work to be done. Currently, porting the G-factor would involve replicating some of the elements. For example, it would involve recreating the second type of game objects and their object model (i.e. the game objects that require visual representation) in the new engine. It would also involve recreating the communication between the game space and the new game engine. There is a need to investigate how this overhead can be reduced. There is also a need to understand how the serious game developers' community is going to react to the changes required to the typical game development approach and the upfront investment needed in terms of implementation overhead. The risks identified by ATAM also need to be examined (see section 4.3.5).

For the second aim, the development of SGTAI has illustrated the ability of GSA to support a virtual training environment that has shown its ability to address the current training issues facing the methods employed by the Dubai police force, which consist mainly of lectures and on-the-job training. The field study conducted in the summer of 2004 revealed a number of issues with these methods. It found that lectures lacked interaction and engagement, which are important in any learning environment (Sankaran

& Bui, 2001; Sachs, 2001). Furthermore the time allocated for the traffic investigation course was not sufficient to cover all the various accident types. The field study also found that the on-the-job training suffered from issues such as impracticality, varying levels of exposure, and lack of uniform assessment (see section 6.2.1). These are some of the learning objectives the design of SGTAI focused on addressing (see section 6.2.2 for the complete list). To ensure these learning objectives were integral to the gameplay the design process combined game design and instructional design. This was achieved by using experiential learning principles (see section 6.4.1), Aldrich's elements (see section 6.4.2), and increasing the feedback loops (see section 6.4.2).

To assess if the SGTAI design succeeded in enhancing learning and addressing the issues facing the training methods, an experiment was conducted in 2006 for 56 police officers from the Dubai police force (see chapter 6). The findings suggest that there is a statistically significant improvement in the performance of both novices and experienced investigators who were trained on SGTAI compared to those who were not. Comments from participants who were trained with SGTAI indicated that it is effective. SGTAI also received positive feedback from educators and policy makers. Educators liked its practicality and policy makers found it to be innovative and saw other opportunities for its use in the police domain. Despite this success, there are still issues to address. One issue that remains is whether or not transfer of learning takes place from the virtual world to the real world. Another issue is to better understand the effect of the abstraction (i.e. through the computer medium) on the learning experience. For example, the use of a graphical user interface might have acted as a constant reminder of the tasks that needed to be accomplished.

This work has implications for the use of serious games in the Dubai police force, the police domain in general, and the whole of the learning domain. For the Dubai police force, the success of a serious game at addressing issues that the current conventional training methods found difficult to deal with demonstrates the viability of this technology, and warrants the need for further investigation of its applicability to other fields. The implications for the police domain in general are concerning the dominance of video-based simulations (Bennell & Jones, 2003). Besides being effective, serious games are cheaper (Aldrich, 2004) and provide better modifiability (modding) which is a powerful feature for digital natives who thrive on social interaction (Herz & Macedonia, 2002). In the Dubai police force, modding also has the potential to be used for sharing experiences. Furthermore, the work contributes to the evidence of the effectiveness of serious games in the police domain in particular, which lacks empirical results (Bennell & Jones, 2003), and learning in general, which is also in need of further evidence. The final implication concerns bridging the gap between showing how existing games employ "best practice" in instructional design and turning that around to use "best practice" to develop good serious games (Becker, 2006b). The design of SGTAI contributes by demonstrating how effective the use of instructional design was alongside game design in the development process.

Appendix A. A Survey of Projects that have used Game Engines

The findings of the survey of projects listed in Table A.1 have been described in section 2.4. The table lists six items for each project. The first item (column three) specifies the game engine used. The second item (column four) specifies whether the project uses a hard-coded or a data-driven approach or a combination of both. To find out if the concept of having the game state (or part of it) outside the engine is acceptable, item three (column five) shows where the game state is at run-time (i.e. inside or outside or uses a combination of both). The game state holds the game objects. If these objects are living inside the engine only then are they labelled inside. If they are living outside the engine and have corresponding objects inside the engine then they are labelled outside. Finally if part of them is inside and the other part is outside then they are a combination of both.

Table A.1: A Survey of projects that have used game engines.

Seq	Project	Engine	Location		Game State (run-time)		Object Model		Game Logic Language		Approach
			Hard-coded	Data-driven	Inside	Outside	Specific	Independent	Specific	Independent	
1	Ambush! (Diller et al., 2005)	Operation Flashpoint		√	√		√		√		
2	Tactical Iraqi (TLTS) (Vilhjalmsson & Samtani, 2005)	Unreal Tournament 2003		√	√	√	√	√	UnrealScript	√ (C++, Python, database, and xml files)	Gamebots, MissionEngine
3	UnrealTriage (first version) (McGrath & Hill, 2004)	Unreal Tournament 2004		√	√		√		UnrealScript		
4	UnrealTriage (second version) (Ryan et al., 2005)	Unreal Tournament 2004		√	√	√	√	√	UnrealScript	√	Extended version of Gamebots
5	Urban search and rescue (Wang et al., 2003)	Unreal Tournament 2003		√	√	√	√	√	UnrealScript	√	Gamebots
6	VRND Notre Dame (DeLeon & Berry, 2000)	Unreal		√	√		√		UnrealScript		
7	Efficient and Dynamic Response to Fire (Darken et al., 2004)	Unreal		√	√		√		UnrealScript		
8	Sonocard[1]	Virtools		√	√		√		√ Graphical tools		
9	Le Redoutable[2] (Blackman, 2005)	Virtools		√	√		√		√ VSL		
10	3D Driving Academy (Traffic AI & Physics engine) (Blackman, 2005)	3D GameStudio (A6 engine)		√	√		√		C-Script		
11	Information and Decision-Making (Creel et al., 2006)	Neverwinter Nights Aurora Engine		√	√		√		NWScript		
12	Mimesis Virtual Aquarium (Young et al., 2004)	Unreal		√	√	√	√	√	UnrealScript	√	Mimesis
13	PSDoom (Chao, 2001)	Doom	√		√	√	√	√	√	√	
14	Visualisation Tools (software Visualization	Quake 3	√	√	√		√		Shader script		

[1] http://www.virtools.com/applications/simulation-enteccs.asp (accessed 1/3/2007)
[2] http://www.virtools.com/applications/simulation-redoutable.asp (accessed 1/3/2007)

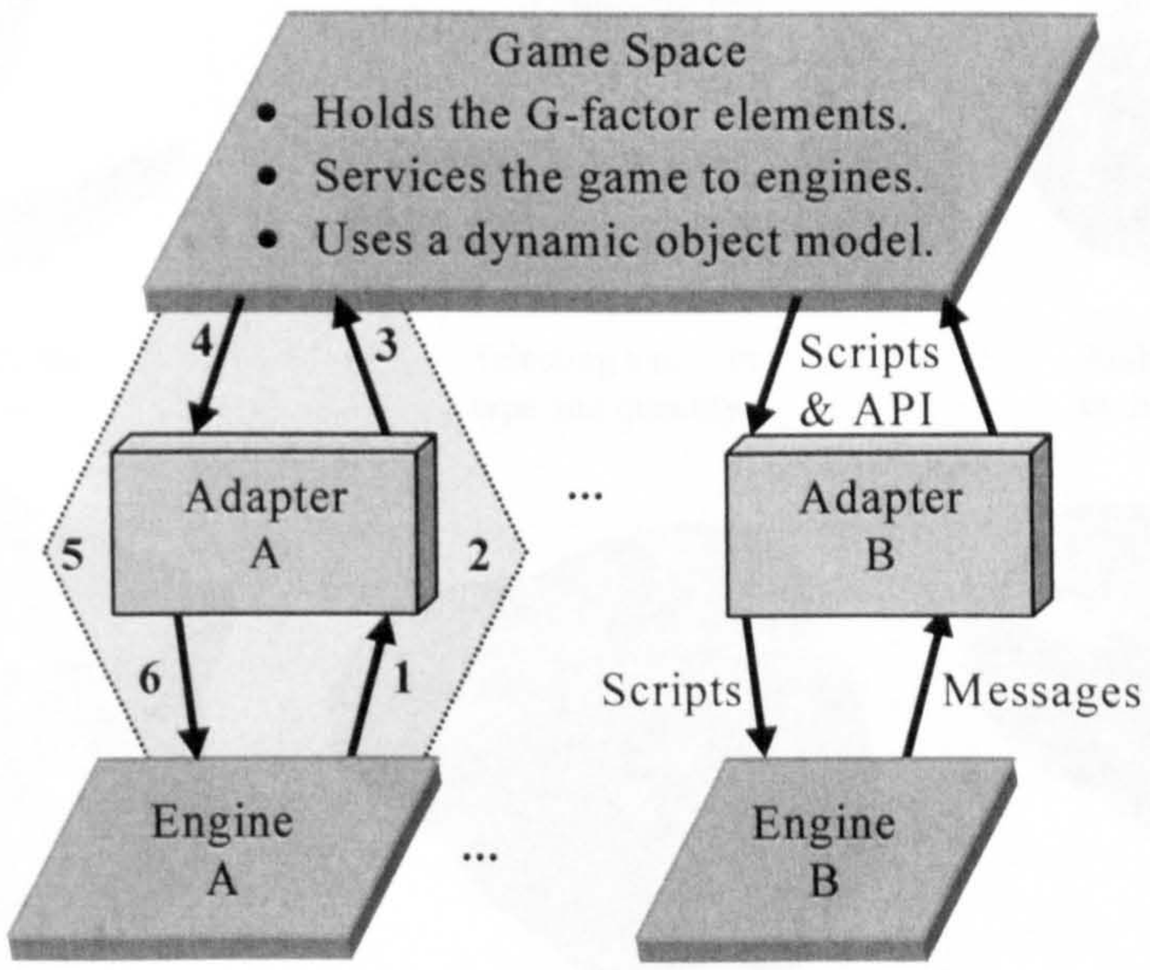
Seq	Project	Engine	Location		Game State (run-time)		Object Model		Game Logic Language		Approach
			Hard-coded	Data-driven	Inside	Outside	Specific	Independent	Specific	Independent	
	tool and a biomedical visualisation tool) (Wuensche et al., 2005)										
15	Flying Mutator (Ota, 2003)	Unreal		√	√		√		UnrealScript		
16	VU-Life 2 (Eliens & Bhikharie, 2006)	Half-Life	√		√		√			√	
17	Creating and Visualising an Intelligent NPC using Game Engines and AI Tools (Davies et al., 2005)	Unreal		√		√		√		√	Gamebots
18	Stratagus: An Open-Source Game Engine for Research in Real-Time Strategy Games (Ponsen et al., 2005)	Stratagus		√		√		√		√	TIELT
19	Neverwinter Nights Game AI (Spronck, 2005)	Neverwinter Nights Aurora Engine		√	√		√		NWScript		
20	Wargus Game AI (Spronck, 2005)	Wargus		√	√		√			Lua	
21	Flexible and Purposeful NPC Behaviors using Real-Time Genetic Control (Hussain & Vidaver, 2006)	Neverwinter Nights Aurora Engine		√		√		√		√	Shadow Door + ACTB-NWN bridge
22	Interacting with Virtual Characters in Interactive Storytelling (Cavazza et al., 2002)	Unreal		√		√	√	√	UnrealScript	√	
23	Qualitative Physics In Virtual Environments (Cavazza et al., 2004)	Unreal		√	√	√	√	√	UnrealScript	√	
24	Extending Game Participation with Embodied Reporting Agents (Fielding et al., 2004)	Unreal		√		√		√	UnrealScript	√	Gamebots
25	Ghostwriter (Robertson & Good, 2003)	Unreal		√	√		√		UnrealScript		
26	America's Army third-person perspective helicopter physics (Davis et al., 2004)	Unreal		√	√		√		UnrealScript		
27	NERO project (Stanley et al., 2005)	Torque		√	√		√		TorqueScript		
28	The Minority Game (Heckenberg et al., 2004)	Unreal Tournament 2003		√	√		√		UnrealScript		
29	Explanation for Hierarchical case-based planning (Muñoz-Avila & Aha, 2004)	Stratagus		√		√		√		√	TIELT
30	Hamlet (Hunicke & Chapman, 2004)	Half-life	√		√		√			√	

Appendix B. Adapter

This appendix describes how the adapter handles the communication between the game engine and the game space. The role of the adapter is to map the G-factor elements (i.e. game logic, object model, and game state) across the game engine and the game space. Section B.1 presents an overview of the adapter. Section B.2 describes the two adapters used for the serious game presented in chapter 6. Section B.3 presents the two adapters used for the portability challenge presented in chapter 4 to link a bespoke engine and Torque game engine. Finally section B.4 describes the adapter used for the performance challenge in chapter 4.

B.1 An Overview of the Adapter

Figure B.1 shows the communication between the game space and the game engine via the adapter. For



Communication between the game engine and the game space

1. Updates are received from the game engine.
2. The adapter uses the scripts mapping table to convert the message to a Jython script.
3. Game state is updated.
4. When a modification is done in the game state the adapter is notified.
5. If the object is of class interest then the adapter converts it to a game engine script.
6. Script is sent to the game engine.

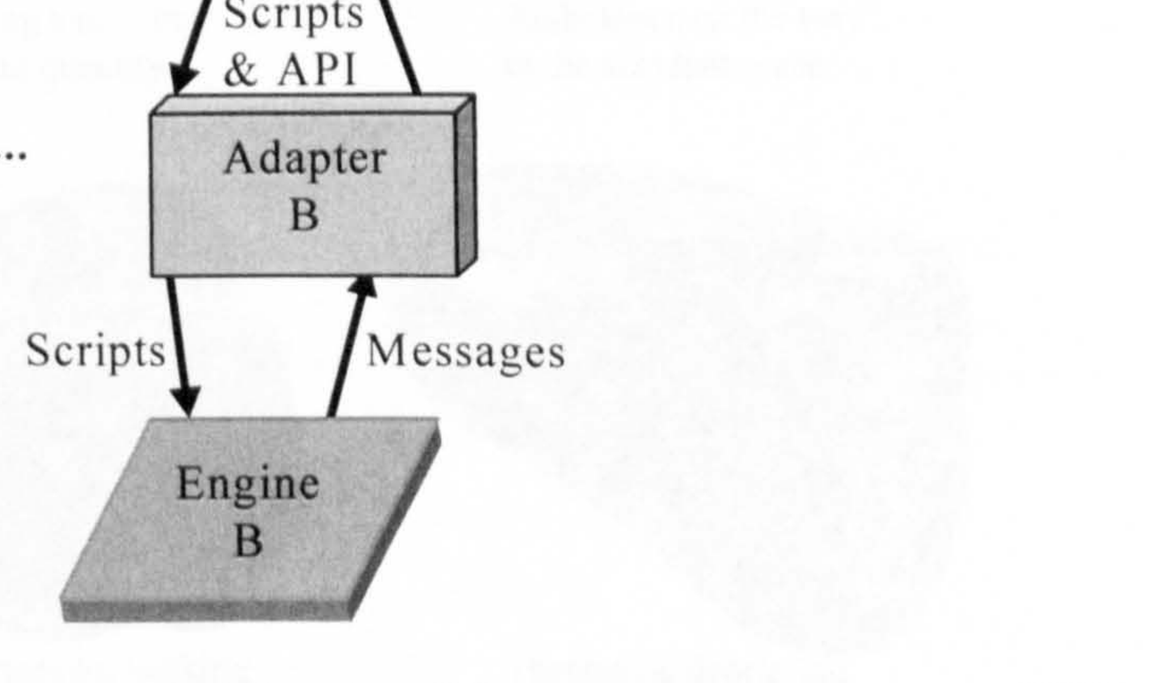


Figure B.1: An overview of the communication between the game space and the game engine via the adapter.

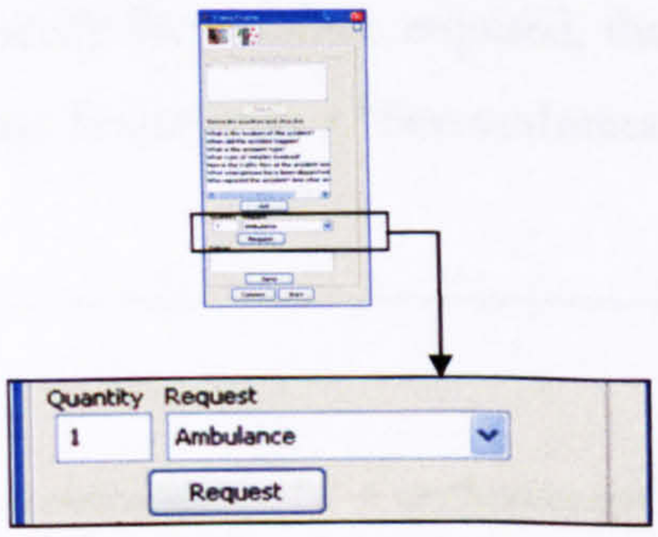
example, communication may begin with the game engine sending the updates to the adapter (step 1). The adapter converts them into scripts or direct API calls (step 2) which are then used to update the game space (step 3). When the game space needs to communicate with the game engine it notifies the adapter of the changes that need to be communicated (step 4). The adapter formats these into the engine's scripting language (step 5) and sends them to the engine to be executed (step 6).

B.2 The Adapter for the Serious Game of Chapter 6

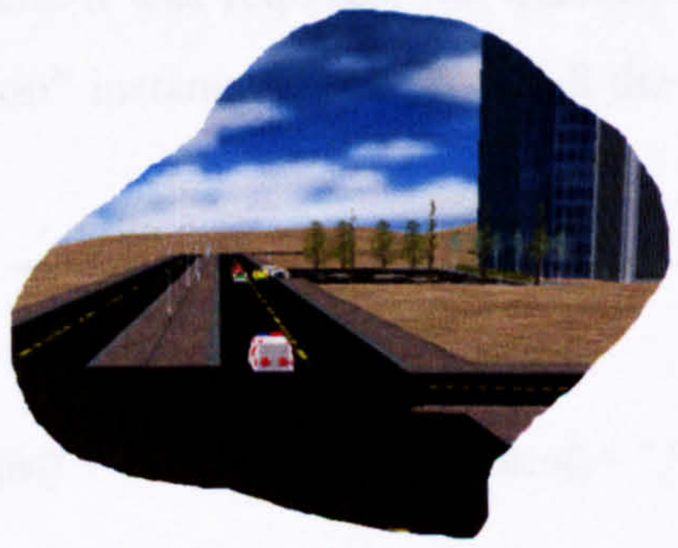
Figure B.2 presents a walkthrough of requesting an ambulance. The request is passed from the game interface to the game space via the interface adapter. The request is then picked by the Torque adapter and gets passed to the game engine.

When the investigator clicks on the radio button an operator interface appears. The investigator

Clicking on the radio button.



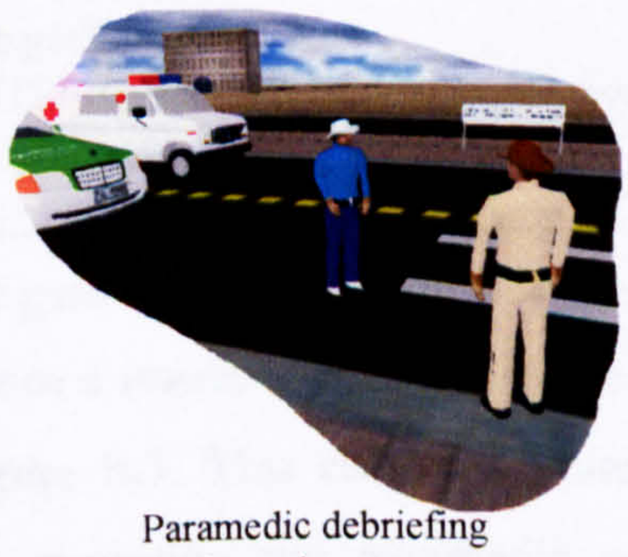
Selecting a resource type and quantity.



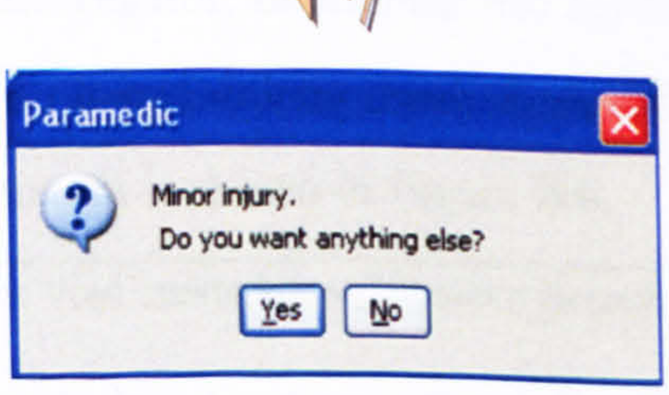
Ambulance on the way to the accident scene.

Ambulance arriving at the scene.

Paramedic walking towards the investigator.



Paramedic debriefing the investigator.



The debrief is in text and audio format.

Paramedic walking back to the ambulance.

Ambulance driving away.

Figure B.2: Requesting an ambulance.

selects the resource type (e.g. ambulance) and the quantity required and then clicks the request button. Figure B.3 shows the function called by the request button. This function converts the call into a predefined message which is then sent to the interface adapter over the network.

```
private void jBtnOprRequest_actionPerformed(ActionEvent e)
  {
   Holder h = (Holder)jComboOprRequest.getSelectedItem();
   System.out.println(h.name + " " + jtfOprQuantity.getText());
   if(jtfOprQuantity.getText().trim().length()>0)
   {
    String requestMsg ="opr_r@"  + h.name + "@" + jtfOprQuantity.getText();
    sendMessage(requestMsg);
   }
}
```

Figure B.3: The game interface request function.

Figure B.4 shows how the interface adapter converts the messages into function calls which add a "ResourceInteractions" instance to specify the resource required, the time it was required, the quantity required, and the reason for the request. It also adds a "SessionInteraction" instance which stores all the interactions.

```
else if(token.equalsIgnoreCase("opr_r"))
{
 ++ontServer.interactionID;
 String strPropertiesValues =ontServer.interactionID + "," + ontServer.getTime() + "," + tokens.nextToken()+ ","
                             +tokens.nextToken() +",,,";
 ontServer.getScenario().addInstance("ResourcesInteractions","ID,Time,ResourceType,Quantity,why,Response1,
                             Lng_Response1",strPropertiesValues,"");
 strPropertiesValues =ontServer.interactionID + "," + ++ontServer.interactionID +
                          ",Resource,"+ontServer.sessionID + "," + ontServer.getTime();
 ontServer.getScenario().addInstance("SessionInteraction","InteractionID,ID,InteractionType,SessionID,time",
 strPropertiesValues,"gameInterfaceAdapter");
}
```

Figure B.4: The interface adapter sample code to update the game state in the game space.

The game loop in the game space monitors the resources and once a resource is added to the game state the "createAgent" function is called which is shown in Figure B.5. This creates a finite state machine which controls the ambulance resource. This include mounting the paramedic on the ambulance, sending the ambulance to the accident location, unmounting the paramedic, walking him to the investigator, debriefing the investigator, walking the paramedic back to the ambulance, mounting him on it and driving away. A sample of the adapter code that is responsible for communicating these commands is shown in Figure B.6.

```
public void createAgent(Instance person,Instance resourceInstance)
{
 StateMachineAgent agent = new StateMachineAgent(ontServer,person,resourceInstance);
 vecAgents.addElement(agent);
 Timer timer = new Timer();
 timer.scheduleAtFixedRate(agent, 100,100);
}
```

Figure B.5: CreateAgent function to start the state machine which controls the paramedic and ambulance.

```
else if(propertyName.equalsIgnoreCase("Action"))
{
 String strAction = instance.getInstancePropertyFirstValue("Action");
 StringTokenizer tokens = new StringTokenizer(strAction,":");
 if(tokens.countTokens()>=2)
 {
  String strCommand=tokens.nextToken();
  String strObjID = tokens.nextToken();
  Instance instanceObj=(Instance)ontServer.getScenario().getInstanceByID(Integer.parseInt(strObjID));
  if(instanceObj!=null)
  {
   String strObjName=instanceObj.getInstancePropertyFirstValue("Name");
 if(strCommand.equalsIgnoreCase("WalkTo"))
 {
  if(strObjName.equalsIgnoreCase("Investigator"))
    vecPlayersWalkingToMovingTarget.addElement(instance.getInstancePropertyFirstValue("Name"));
  String strTorque=instance.getInstancePropertyFirstValue("Name") + ".disableCollision();";
  strTorque += instance.getInstancePropertyFirstValue("Name") + ".setMoveDestination(adjustMove(" +
                    strObjName + ".getTransform()));";
  sendToTorque(strTorque);
 }
 else if(strCommand.equalsIgnoreCase("GetIn"))
 {
  String strPlayer =instance.getInstancePropertyFirstValue("Name");
  String strTorque = strObjName + ".mountObject(" + strPlayer + ", 2);" + strPlayer
                    + ".setActionThread(\"sit\");";
  sendToTorque(strTorque);
 }
else if(strCommand.equalsIgnoreCase("GetOut"))
{
 String strTorque = strObjName + ".mountObject(" + instance.getInstancePropertyFirstValue("Name")
                    + ", 5);";
 sendToTorque(strTorque);
 strTorque = strObjName + ".unmountObject(" + instance.getInstancePropertyFirstValue("Name") + ");";
 sendToTorque(strTorque);
}
else if(strCommand.equalsIgnoreCase("DriveTo"))
{
 String strDestination=((Instance)ontServer.getScenario().getInstanceByID(
                        Integer.parseInt(tokens.nextToken()))).getInstancePropertyFirstValue("Name");
 String strTorque = strObjName+".setMoveDestination(" + strDestination +
                    ".getPosition());"+strObjName+".playAudio(2,NewSoundStart);onAddSiren("
                    + strObjName +");";
 sendToTorque(strTorque);
}
else if(strCommand.equalsIgnoreCase("AskInvestigator"))
{
 String strTorque = instance.getInstancePropertyFirstValue("Name") + ".setMoveDestination("
                    + strObjName +"); $BasicObject.say(\"what do you want\");";
 sendToTorque(strTorque);
}
}}}
```

Figure B.6: A sample of Torque adapter code controlling the resource interactions.

B.3 The Adapter for the Smart Terrain Example

This example was developed using an earlier version of GSA where a behaviour engine (i.e. Jess rules-engine) was embedded in the game space (BinSubaih et al., 2005b). In this example the environment allows the player to navigate and interact with non-player characters (NPCs) and objects. The player also receives hints about the zones. The engines send the player position to the game space as shown in Figure B.7a for the bespoke engine and Figure B.7b for Torque engine. The position updates are then

```
strcpy(szMessage,"");
strcpy(szMessage,"4@");
strcat(szMessage,IToCS(MyNetID));
strcat(szMessage,"@");
strcat(szMessage,IToCS(MyGameID));
strcat(szMessage,"@");
if((eye==0)&&(rightEyeIDIndex!=-1))
{
   strcat(szMessage,FToCS(x));//x
   strcat(szMessage,"@");
   strcat(szMessage,FToCS(y));//y
   strcat(szMessage,"@");
   strcat(szMessage,FToCS(z));//z
}
else
{
   strcat(szMessage,FToCS(m_pCamera->GetXPos()));//x
   strcat(szMessage,"@");
   strcat(szMessage,FToCS(y));//y
   strcat(szMessage,"@");
   strcat(szMessage,FToCS(m_pCamera->GetZPos()));//z
}
strcat(szMessage,"@");
strcat(szMessage,FToCS(m_pCamera->GetXRotation()));//xRot
strcat(szMessage,"@");
strcat(szMessage,FToCS(m_pCamera->GetYRotation()));//yRot
strcat(szMessage,"@");
strcat(szMessage,FToCS(m_pCamera->GetZRotation()));//zRot
m_pClient->SendTextMsg(szMessage);
```

(a) Converting the player movement to a predefined message and sending to the bespoke engine adapter.

```
function updatePositions()
{
   %count = ClientGroup.getCount();
   for(%cl= 0; %cl < %count; %cl++)
   {
      %client = ClientGroup.getObject(%cl);
      %position = %client.player.getPosition();    // do whatever you want with the position
      %positionFormatted =strreplace(%position," ", "@");
      %id =%client.player.getId();
      %msg="4" @ "@-1@" @ %id @ "@" @ %positionFormatted @ "0@0@0";
      $BasicObject.processMessage(%msg);
   }
}
```

(b) Converting the player movement to a predefined message and sending to Torque engine adapter.

Figure B.7: Sending predefined messages of the selection to (a) the bespoke engine adapter and (b) Torque engine adapter.

```
if(i==4)
 {
  int gameID=-1;
  int dpnidPlayer=-1;
  float x,y,z,xRot,yRot,zRot;
  x=0.0;y=0.0;z=0.0;
  sscanf(msg,"%i@%u@%i@%f@%f@%f@%f@%f@%f",&i,&dpnidPlayer,&gameID,&x,&y,&z,&xRot,&yRot,&zRot);
  cOntology *ont=getOntology(gameID);
  if(ont!=0)
  {
    CString strGameID="";
    strGameID.Format("%d",gameID);
    CString jessMessage="2@(modify (fact-id "+ getPropertyInstanceValue(strGameID,"Graphics_Attribute");
    jessMessage+= ")(";
    jessMessage+= getPropertyMap("x");
    jessMessage+= " ";
    jessMessage+= FToCS(x);
    jessMessage+=")(";
    jessMessage+= getPropertyMap("y");
    jessMessage+= " ";
    jessMessage+=FToCS(z);
    jessMessage+=")(";
    jessMessage+=getPropertyMap("z");
    jessMessage+= " ";
    jessMessage+=FToCS(y);
    jessMessage+="))";
    jessMessage+= "\n";
    jessMessage.ReleaseBuffer();
    client->Send(jessMessage,jessMessage.GetLength(),0);
  }
}
```

Figure B.8: Converting the predefined message into JessScript for the bespoke engine and Torque engine.

translated in the adapters to JessScript format (see Figure B.8). When the player enters a zone of interest a rule (see Figure 4.3) is fired by Jess engine which sends a message to the player (e.g. "You have entered the accident scene zone. You should secure the scene and search for injured people"). Figure B.9 and Figure B.10 illustrates this. Upon receiving the messages from the engines the adapter converts them to appropriate calls (i.e. Jython for bespoke engine and TorqueScript for Torque) and sends them to the engines to be executed (see Figure B.11).

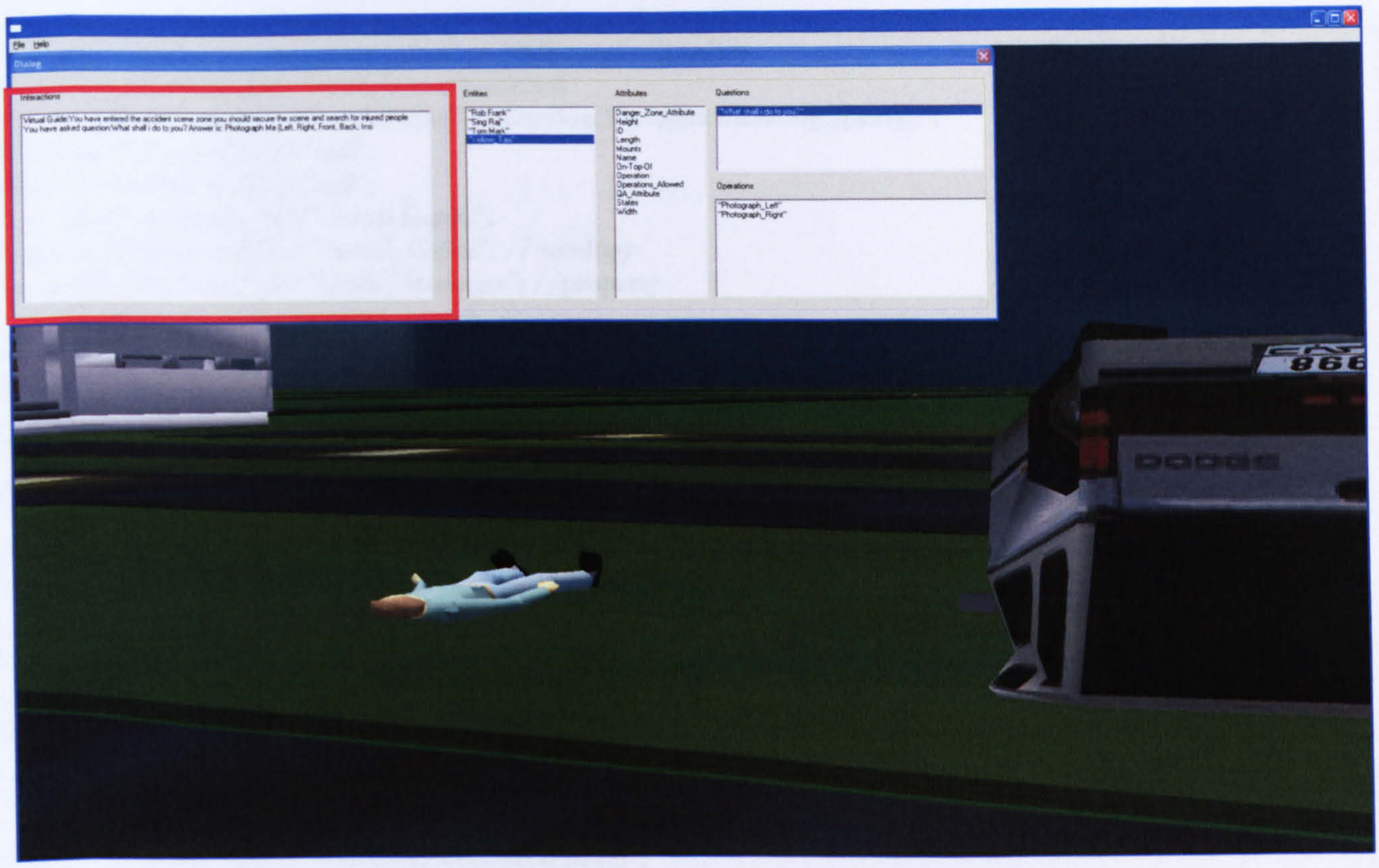


Figure B.9: Running smart terrain on the bespoke engine.

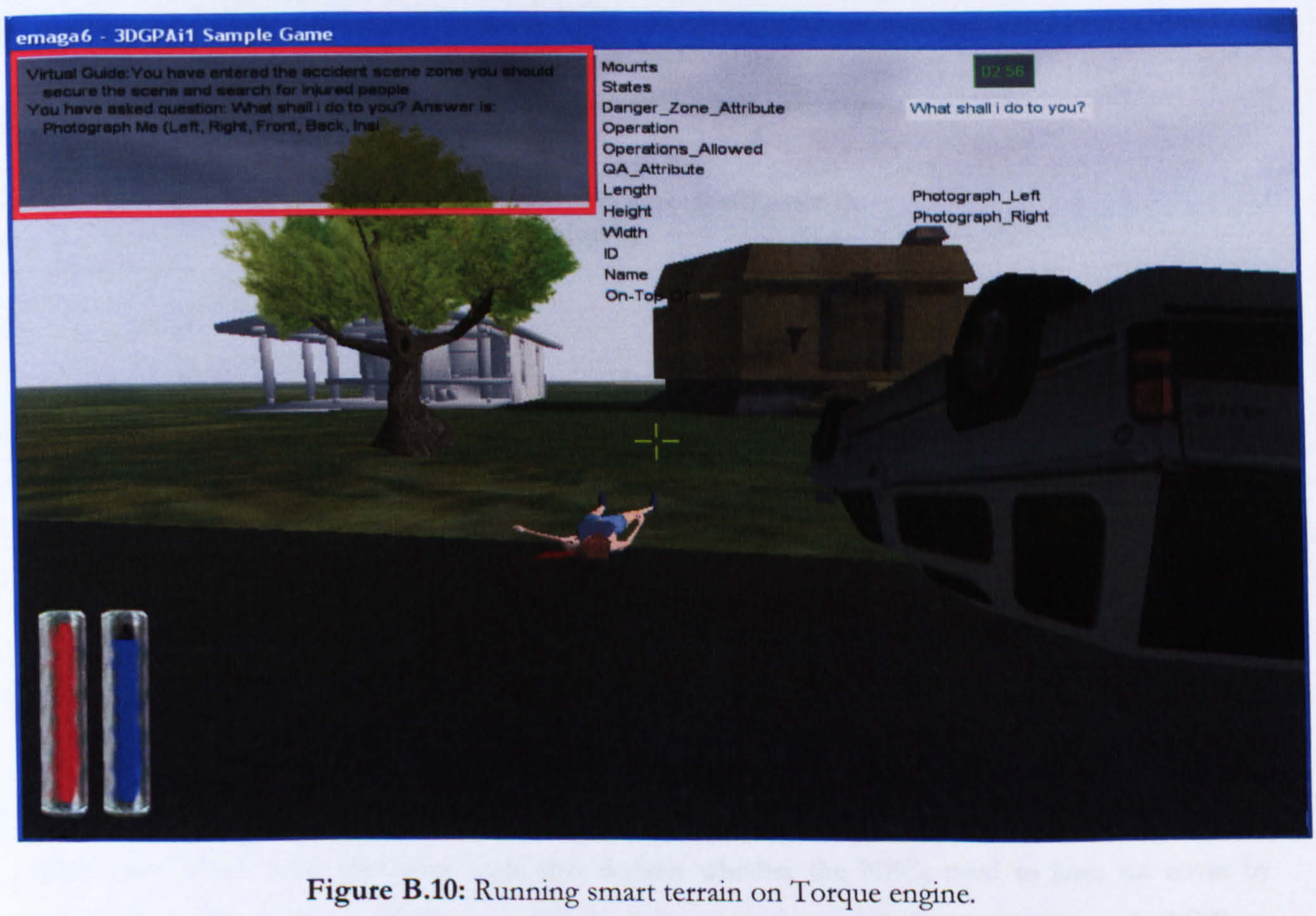


Figure B.10: Running smart terrain on Torque engine.

```
//Adding the warning message using Jython: p.AddStringToCListBox
arrPlayerMapToPython[2][0]="null";//source control
arrPlayerMapToPython[2][1]="p.AddStringToCListBox(1018,@itemData@,@str@)";
arrPlayerMapToPython[2][2]="null";
arrPlayerMapToPython[2][3]="null";
arrPlayerMapToPython[2][4]="Virtual Guide:";
arrPlayerMapToPython[2][5]="Virtual_Guide"; //ontology
arrPlayerMapToPython[2][6]="Guide_Messages"; //property

CString strPython="901@"+getPythonScriptByOntologyAndProperty(strOntology,strProperty);
strPython.Replace("@itemData@",IToCS(ont->factID));
CString strToBeInserted="";
strToBeInserted+=getUserMsgByOntologyAndProperty(strOntology,strProperty);
CString strTemp=ont->getPropertyValue("Guide_Messages");
strTemp.Remove("");
strToBeInserted+= strTemp;
strToBeInserted+="";
strPython.Replace("@str@",strToBeInserted);
//Replace the witespace by * before sending it to python
ReplaceWhiteSpace((char*)(const char*)strPython);
server->SendTextMsg(DVID_ALLPLAYERS,(char*)(const char*)strPython);
```

(a) Converting the warning to Jython script.

```
//Adding the warning message using TorqueScript: ChatHud.addLine
arrPlayerMapToPython[3][0]="null";//source control
arrPlayerMapToPython[3][1]="ChatHud.addLine(@str@)";
arrPlayerMapToPython[3][2]="null";
arrPlayerMapToPython[3][3]="null";
arrPlayerMapToPython[3][4]="Virtual Guide:";
arrPlayerMapToPython[3][5]="Virtual_Guide"; //ontology
arrPlayerMapToPython[3][6]="Guide_Messages"; //property

CString strTorque=getTorqueScriptByOntologyAndProperty(strOntology,strProperty);
strTorque.Replace("@itemData@",IToCS(ont->factID));
CString strToBeInserted="";
strToBeInserted+=getUserMsgByOntologyAndProperty(strOntology,strProperty);
CString strTemp=ont->getPropertyValue("Guide_Messages");
strTemp.Remove("");
strToBeInserted+= strTemp;
strToBeInserted+="";
strTorque.Replace("@str@",strToBeInserted);
strTorque+=';';
basicObject->executeScript(strTorque);
```

(b) Converting the warning to TorqueScript.

Figure B.11: Converting the warning message into (a) Jython script for the bespoke engine and (b) TorqueScript for Torque engine.

B.4 The Adapter for the First-Person Shooter (FPS) Game

In the sample game described in section 4.2.3, non-player characters (NPCs) run for cover from a player. The role of the adapter is to link Torque game engine. which receives the player movements, to the game space, which holds the game logic that decides whether the NPCs need to look for cover by moving to another waypoint. If that is needed the game space then sends the instruction to the NPCs to move. This is passed back to the game engine via the adapter. Figure B.12 shows the code which

constructs a predefined message and sends it to Torque adapter. The adapter converts the messages to function calls to the game space to update its state. as shown in Figure B.13.

```
if (%obj.getClassName() $= "AIPlayer")
{
  %updateMsg="3a@" @ %obj.getName()@"@"@%obj.getPosition()@"@"@ %obj.humansInSight @ "@"
          @ %obj.waypointsInSight@ "@" @ getReached(%this,%obj);
  if(%obj.lastMsg!$=%updateMsg)
  {
    chatConnection.sendMessage(%updateMsg);
    %obj.lastMsg=%updateMsg;
  }
}
```

Figure B.12: The message sent from the game engine to Torque adapter.

```
Instance instance =ontServer.getScenario().getInstance(torqueName);
if(instance!=null)
{
 //set HumansInRange
 scenario.setPropertyValue(instance,"HumansInRange",tokens.nextToken(),"torqueAdapter");
 //set NPCsInRange
 scenario.setPropertyValue(instance,"NPCsInRange",tokens.nextToken(),"torqueAdapter");
 //set HumansInSight
 scenario.setPropertyValue(instance,"HumansInSight",tokens.nextToken(),"torqueAdapter");
 //set NPCsInSight
 scenario.setPropertyValue(instance,"NPCsInSight",tokens.nextToken(),"torqueAdapter");
 //set WaypointsInRange
 scenario.setPropertyValue(instance,"WaypointsInRange",tokens.nextToken(),"torqueAdapter");
 //set WaypointsInSight
 scenario.setPropertyValue(instance,"WaypointsInSight",tokens.nextToken(),"torqueAdapter");
 scenario.setPropertyValue(instance,"onReach",tokens.nextToken(),"torqueAdapter");
}
```

Figure B.13: Torque adapter convert the message to function calls.

The game loop in the game space monitors the player movement (see Figure B.14) and once it gets close to NPCs an instance of type "MoveTo" is added to the game state. The adapter monitors this class (i.e. "MoveTo") when an instance is added the code in Figure B.15 is executed which sends a torque script to the game engine (e.g. "NPC.setMoveDestination(Waypoint.getPosition());NPC.reache=0;").

```
public void runGameLoop()
{
  ...
  scenario.addInstance("MoveTo","ID,Name,NodeObject,NodeDestination",strValues,"");
  instance.getInstanceProperty("OnReach").setPropertyValues("0");
  ...
}
```

Figure B.14: The game loop in the game space.

```
//Get the class name
String className = ontServer.getOnt().getClass(instance.idon_classes).getName();
if(className.equalsIgnoreCase("MoveTo"))
{
 //Get NodeDestination,NodeObject
 InstanceProperty nodeDestination = instance.getInstanceProperty("NodeDestination");
 InstanceProperty nodeObject = instance.getInstanceProperty("NodeObject");
 if((nodeDestination!=null)&&(nodeObject!=null))
 {
  String moveMsg= instance.getInstancePropertyFirstValue("NodeObject") + ".setMoveDestination(" +
                  instance.getInstancePropertyFirstValue("NodeDestination")+
                  ".getPosition());"+instance.getInstancePropertyFirstValue("NodeObject")+".reached=0;";
  sendToTorque(moveMsg);
 }
}
```

Figure B.15: The TorqueScript code sent from Torque adapter to Torque engine.

Appendix C. Building SGTAI

The serious game for traffic accident investigators (SGTAI) described in chapter 6 was developed using GSA and the Torque game engine. This appendix presents the development using GSA's five steps (see section 3.5): create the level data (section C.1), create the GUI (section C.2), create the object model (section C.3), create the game logic (section C.4), and create the adapter (section C.5).

C.1 Level Data

The development pipeline of the assets needed for the level data is shown in Figure C.1. The roads were created using 3D Studio Max and the buildings and vehicles are from freely-available online resources (3D Cafe[1] and TurboSquid[2]), which were modified to suit the requirements of SGTAI, e.g. the police car was customised and damage was added to other vehicles. The characters were also from online resources and were modified to fit in with Arabic culture. The faces for the models were generated using FaceGen[3], and then exported to the Torque format.

GSA did not introduce major changes to the typical game development approach when creating the level data. The engine's world builder was used to create the decorative objects and game objects. The only additional work was to separate the game objects into two types: ones that have visual representation in the game engine such as characters, markers, and cones, and ones that do not have visual representation such as user interactions (e.g. Moving, Navigation, Photographing, and Measuring). The first object types are created in the game engine (using world builder and TorqueScript) as well as in the game space (using Jython). These objects are given unique IDs to be able to identify them across the game engine and the game space. The second object types are created only in the game space using Jython. Figure C.3 shows the three scenarios created. The first scenario was used in the experiment described in chapter 6. The second is between a car and a bicycle and the third is between two cars at a junction. The second and third scenarios were developed to highlight GSA's modifiability. The same object model (see section C.3), dialogue system, logic to control the resources (see section C.4), and adapter (see section C.5) were used across the three scenarios. This demonstrates that a huge part of the development is done once and reused for different levels.

[1] http://www.3dcafe.com (accessed 2/8/2006).
[2] http://www.turbosquid.com (accessed 2/8/2006).
[3] http://www.facegen.com (accessed 2/8/2006).

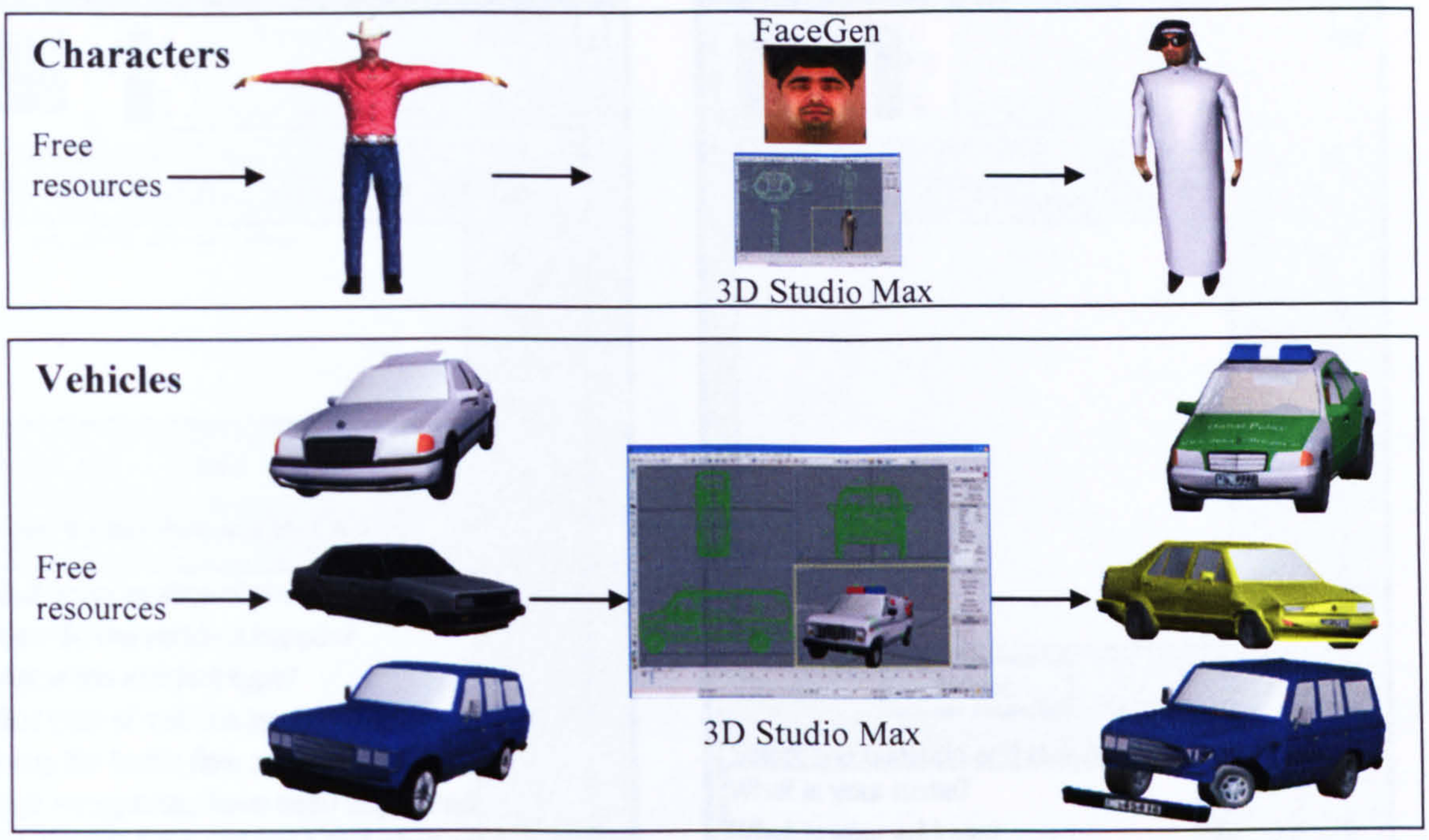
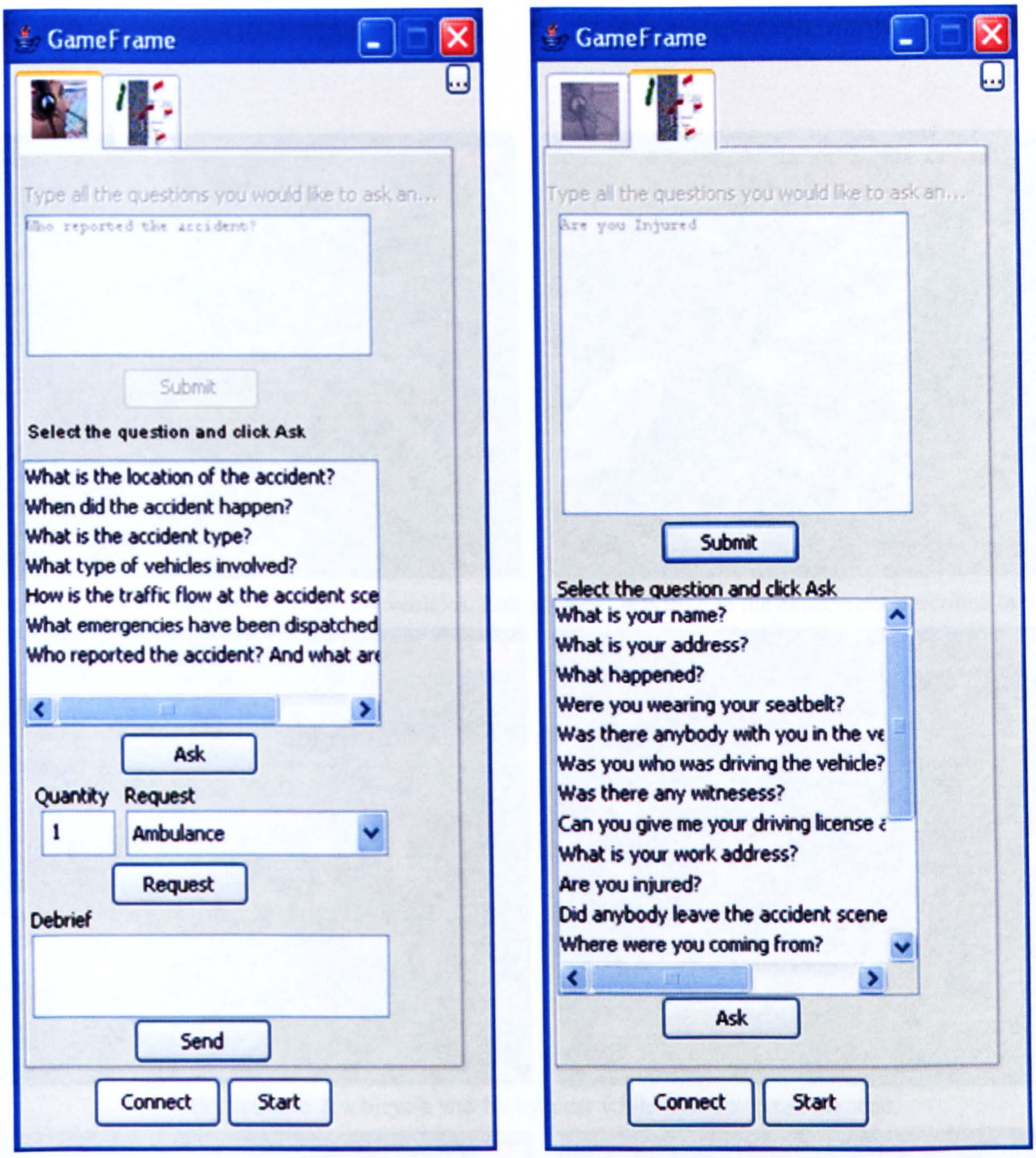


Figure C.1: 3D assets.

C.2 Interface

Figure C.4 shows the game interface. This was created using the Torque GUI editor. The menu has the following items starting from the top: timer, compass[4], camera, measuring wheel, traffic cones, markers, investigator folder, and radio. The player can navigate using the keyboard arrows, and the mouse wheel is used to look up and down. Figure C.2 shows the communication interfaces, developed in Java, to communicate with the operator in the operation room (see Figure C.2a) and the characters at the scene (see Figure C.2b). The reason for not using Torque GUI editor to build this was because Torque 1.3 did not support Unicode at the time of development of the system (which is needed to display Arabic text). A more recent version of Torque (version 1.4) has added support for Unicode.

[4] Adopted from Garage Games Modding Community.

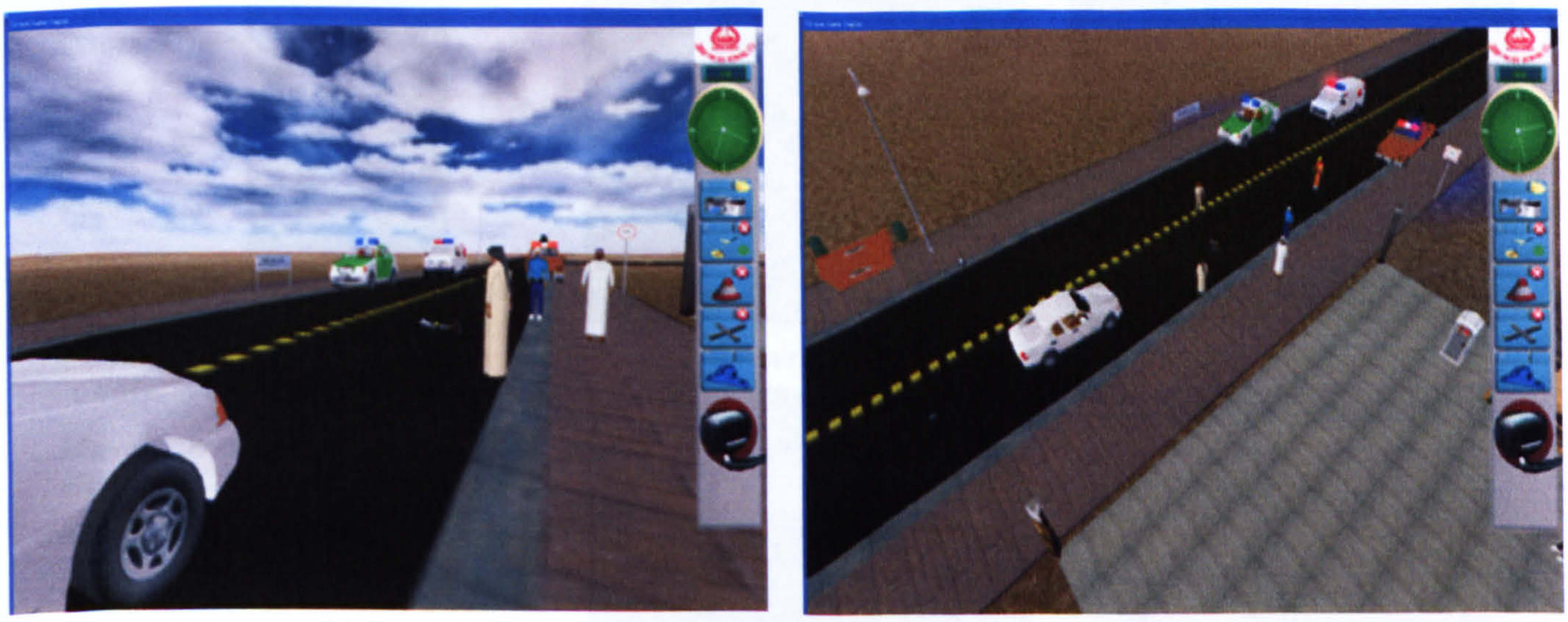


(a) Communicating with operation room.

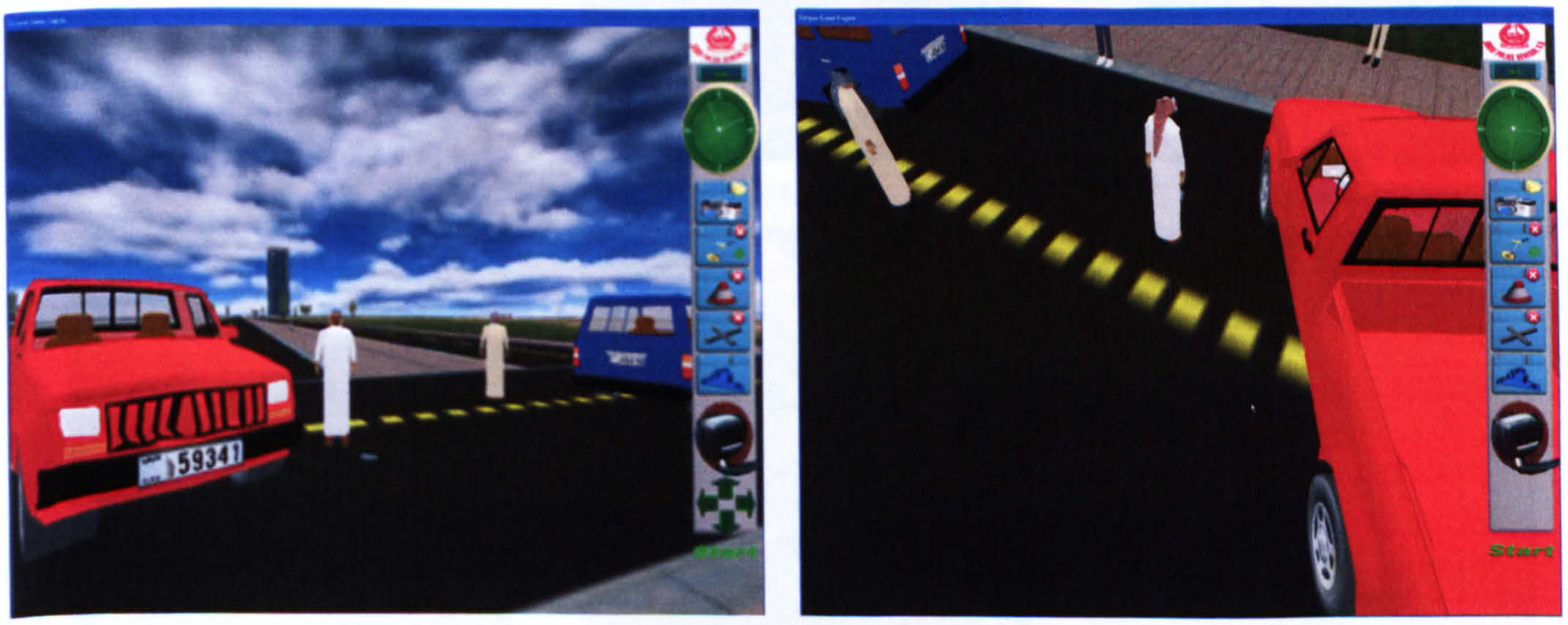
(b) Communicating with characters.

Figure C.2: The communication interfaces.

(a) Scenario 1: is an accident between two vehicles. This accident was used in the experiment described in chapter 6.

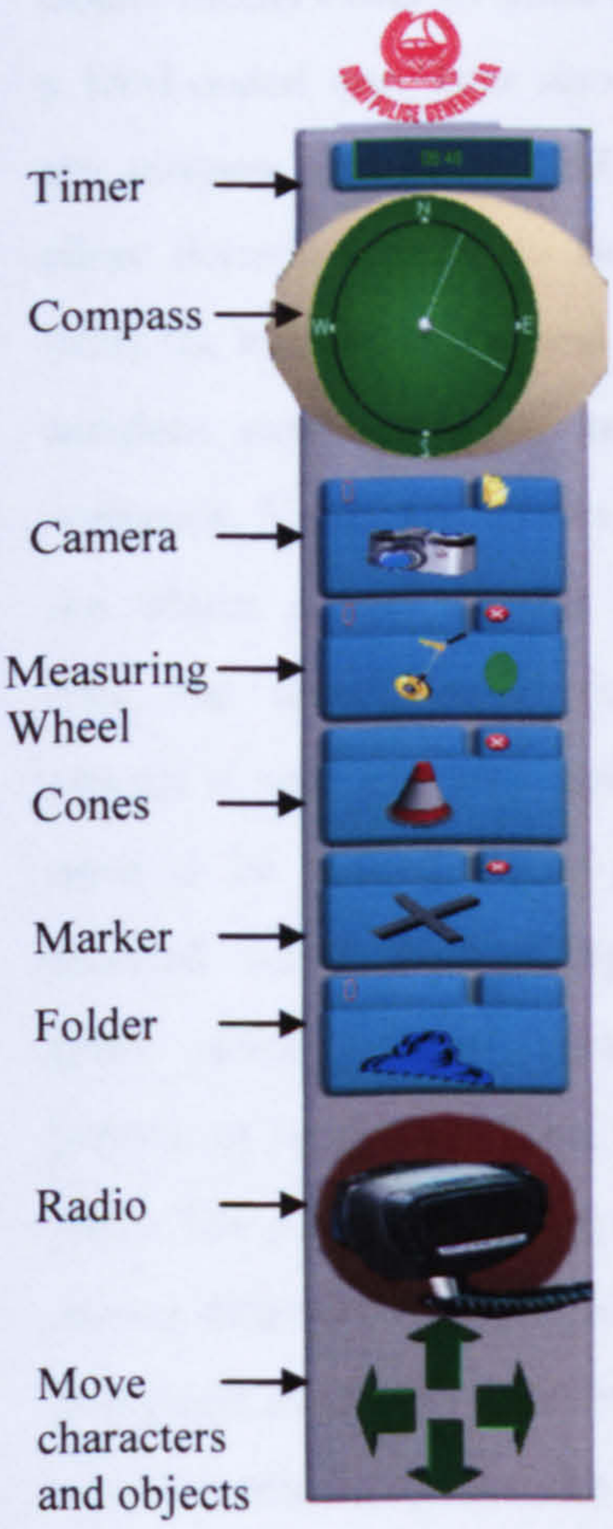


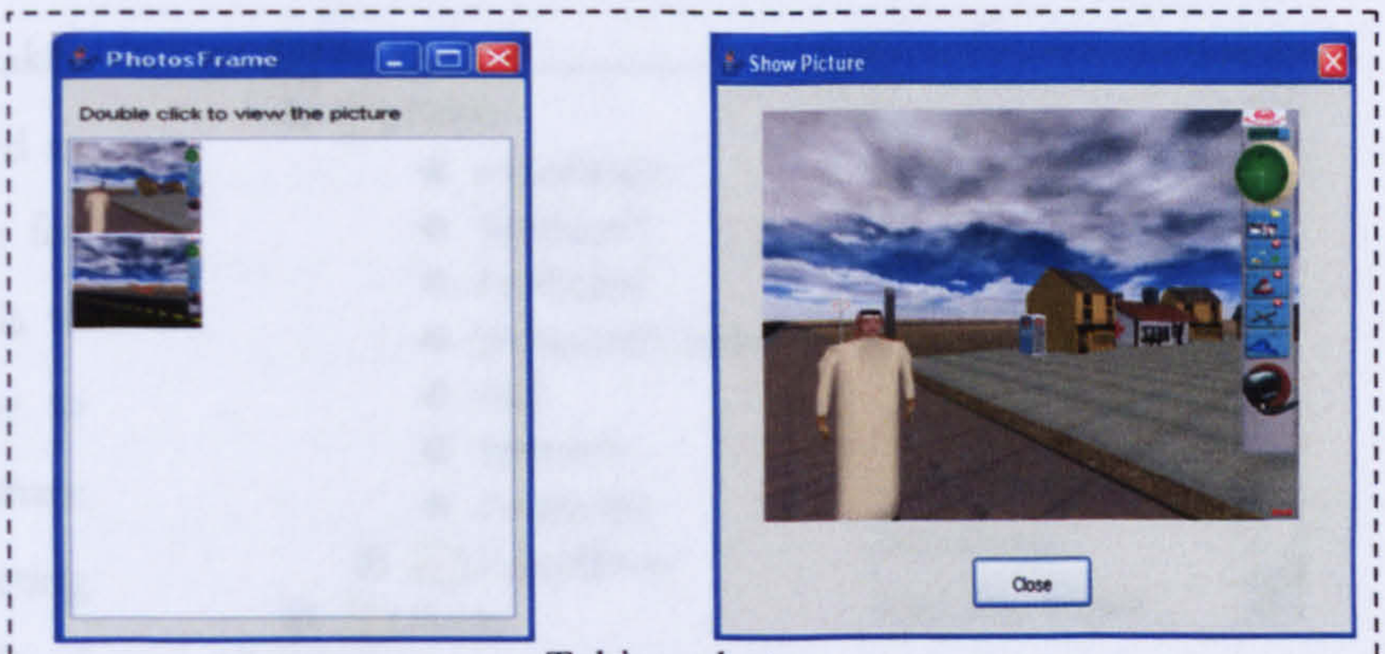
(b) Scenario 2: a bicycle was hit by a car while trying to cross the road.



(c) Scenario 3: the red car entered the road without checking to see if it was clear and was hit by the blue car.

Figure C.3: The three accident scenarios created.





Taking photos

Measuring wheel

Cone

Marker

Driving license

Moving and interacting with a character

Moving the police car

Reflection box

Figure C.4: The interface showing the menu and how the real investigation experience looks in the virtual environment (e.g. taking photographs, driving licences, taking measurements).

C.3 Object Model

The architecture uses a dynamic object model (see section 2.3.2). This makes object model easier to build compared to a hard-coded one. It is also suitable for the ultimate aim of SGTAI which is to allow domain experts to build these to unify the terminologies used and to share accident experiences by reconstructing scenarios. Figure C.5 shows a sample of the object model. Similar to the level data, the object model for the game objects is split into two types: ones that need to be created in the game engine (created using TorqueScript) and the game space (created using OntRAT, Jython, or Java), and others that are only needed in the game space (created using OntRAT, Jython, or Java). The practice of using a database to set the object model made it easier to reuse the object model across different accident scenarios. It also speeded up the development process when using GSA compared to the development using hard-coded classes. The availability of OntRAT to create the object model further simplified the development.

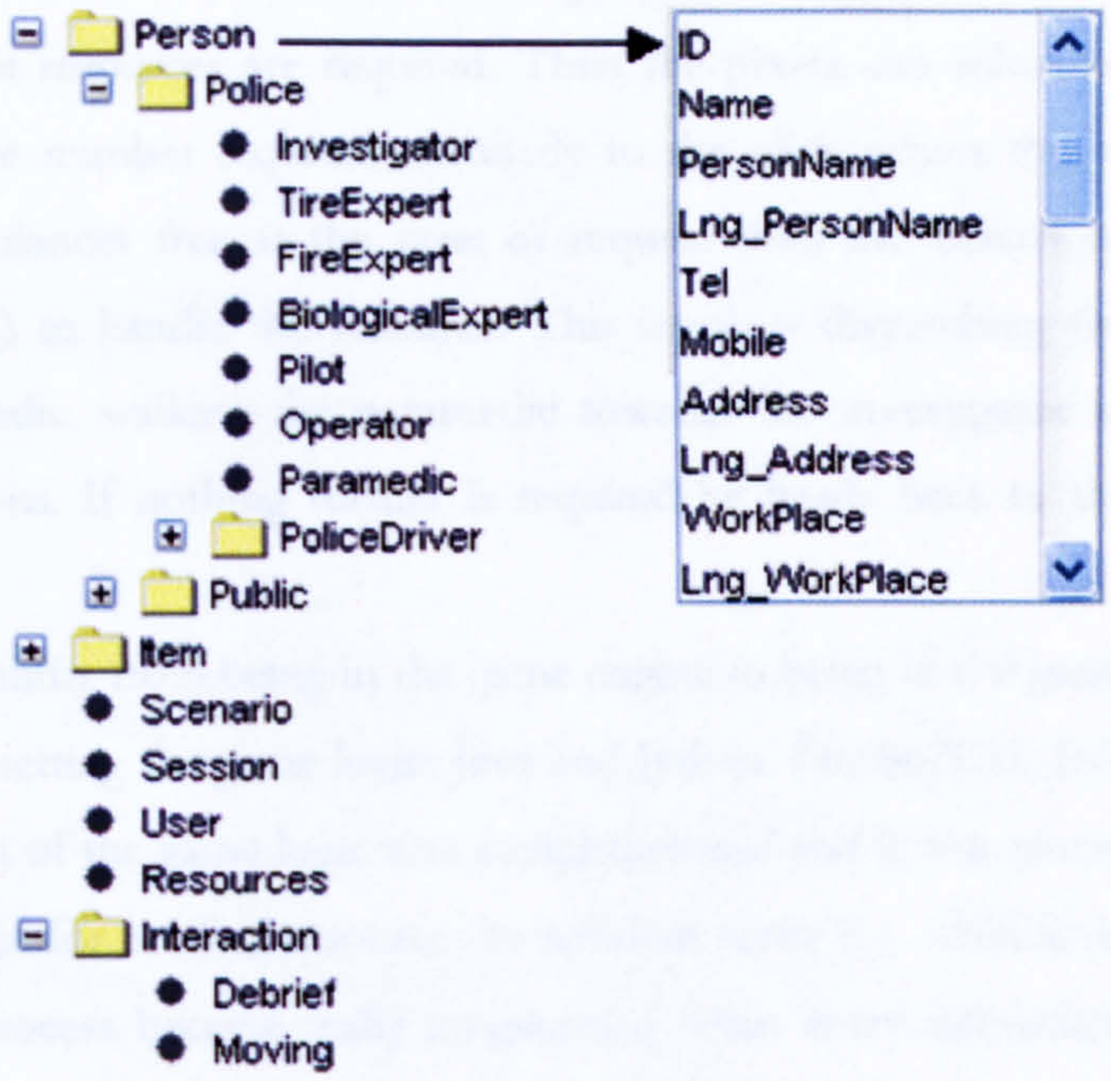


Figure C.5: Sample of the object model.

C.4 Game Logic

The game logic of a typical SGTAI session starts with the investigator standing beside his patrol vehicle waiting for an incident call. Upon receiving and accepting the deployment, the investigator is put into a car and gets driven to the accident scene – the investigator does not drive the vehicle as the training is aimed at the officer in charge. During the travel his role is to communicate with the operation room to find out more details about the incident (such as who reported it, seriousness, number of vehicles involved, etc). After arriving at the accident scene, the investigator is placed outside the vehicle and he can start attending to the accident. His first role is to secure the accident scene by clicking on the patrol vehicle and moving it to an appropriate spot. Then he can search for injured people and request additional resources (i.e. an ambulance) from the operation room. After that he can carry out other tasks such as asking questions, examining the scene, placing markers, taking photographs, taking measurements, etc.

The behaviour is controlled from the game space which receives updates and sends actions to the game engine. Consider for example the action of requesting an ambulance. When the investigator clicks

on the radio icon, the click is sent to the game space which updates its game state. This state is monitored by a behaviour controller, which, in this instance, sends an action to the game engine to display the operator interface shown in Figure C.2a and synthesize a message acknowledging the action and requesting the investigator to specify what resources are required. Then the player can select an ambulance from the resources and specify the number required. Similarly to the click action this is updated to the game state. If there are ambulances free at the time of request then the behaviour controller creates a finite state machine (FSM) to handle the resource. This involves dispatching the ambulance to the scene, dropping the paramedic, walking the paramedic towards the investigator to debrief him, and waiting for further instructions. If nothing further is required he heads back to the ambulance and drives away.

In GSA the development of the game logic shifts from being in the game engine to being in the game space. The game space provided two ways for setting the game logic: Java and Jython. For SGTAI, Java was used to create the game logic. The creation of the game logic was straightforward and it was reused across different scenarios such as the game logic for sending resources to accident scene (i.e. ambulance and tow truck). However, the development process became really longwinded when every interaction had to be communicated from the game engine to the game space and back again if necessary. It also required the developer to be disciplined in doing that instead of the using the shortcut by developing it in the game engine. However sometimes the incentive of being able to port that part of the logic, especially when it was very small, was not as strong as the urge to speed the compile and run cycle. Figure C.6 shows an example of this issue where part of the logic was left in the game engine. What should have been done is to notify the game space at point A of the measurement icon being clicked. The game space will then send the block of code B or C depending on the current state of the wheel (on or off). If the state of the wheel is off when it is clicked, the code in block B is executed which places the measuring wheel in front of the investigator and changes the icon colour. If the state of the wheel is on when clicked, the code in block C sends the distance to the game space. Therefore what is required for the GSA approach is to add another tidying step to ensure that these steps are rectified.

```
function onMeasure(%this)                                          <-- A
{
  if(imgMeasure.state==0)
  {
    addWheel();
    getUserName().mountObject($wheel,2);
    $distance=0;
    distanceWheel.setText("\c4 " @$distance);                      } B
    getUserName().lastMeasureMsg=getUserName().getPosition();
    imgMeasure.setBitmap("starter.fps/client/ui/images/measure_r.png");
    imgMeasure.state=1;
    onMeasuring();
  }
  else
  {
    $wheel.delete();
    imgMeasure.setBitmap("starter.fps/client/ui/images/measure_g.png");
    imgMeasure.state=0;
    cancel($uMeasure);
    if($connected==1)
    {
       %measureMsg="measuring@"@getUserName()@"@"@ $distance@"@"@ clock.getTime();
        chatConnection.sendMessage(%measureMsg);
     }
    }
}
```

C

Figure C.6: Sample code highlighting the issue with small game logic that was not made portable.

C.5 Adapter

Two adapters were developed for SGTAI (see Figure C.7): Torque adapter, and interface adapter. The Torque adapter is used to communicate between the game space and the game engine. The interface adapter is used to communicate between the game space and the interfaces used for communicating with the operation room and characters at the scene. Although the interface adapter is not a requirement when following the GSA approach, the opportunity rose because of the lack of Arabic text support with earlier versions of Torque. This was used to make the interface portable. The implementation of the adapters follows a similar approach to the one used for the Moody NPCs game (see section 3.5).

The major difference between GSA and the typical game development was apparent in creating the adapter. The development required sending every interaction that was relevant to the game logic from the game engine to the game space and vice versa. The adapter's development also required creating a mapping between object models, and languages across the game engine and the game space. The positive point about the mapping was that it was created once and utilized thereafter for applications of similar requirements (i.e. the three traffic accident scenarios). However, there were two concerns with the use of the adapter: sensitivity and performance overhead.

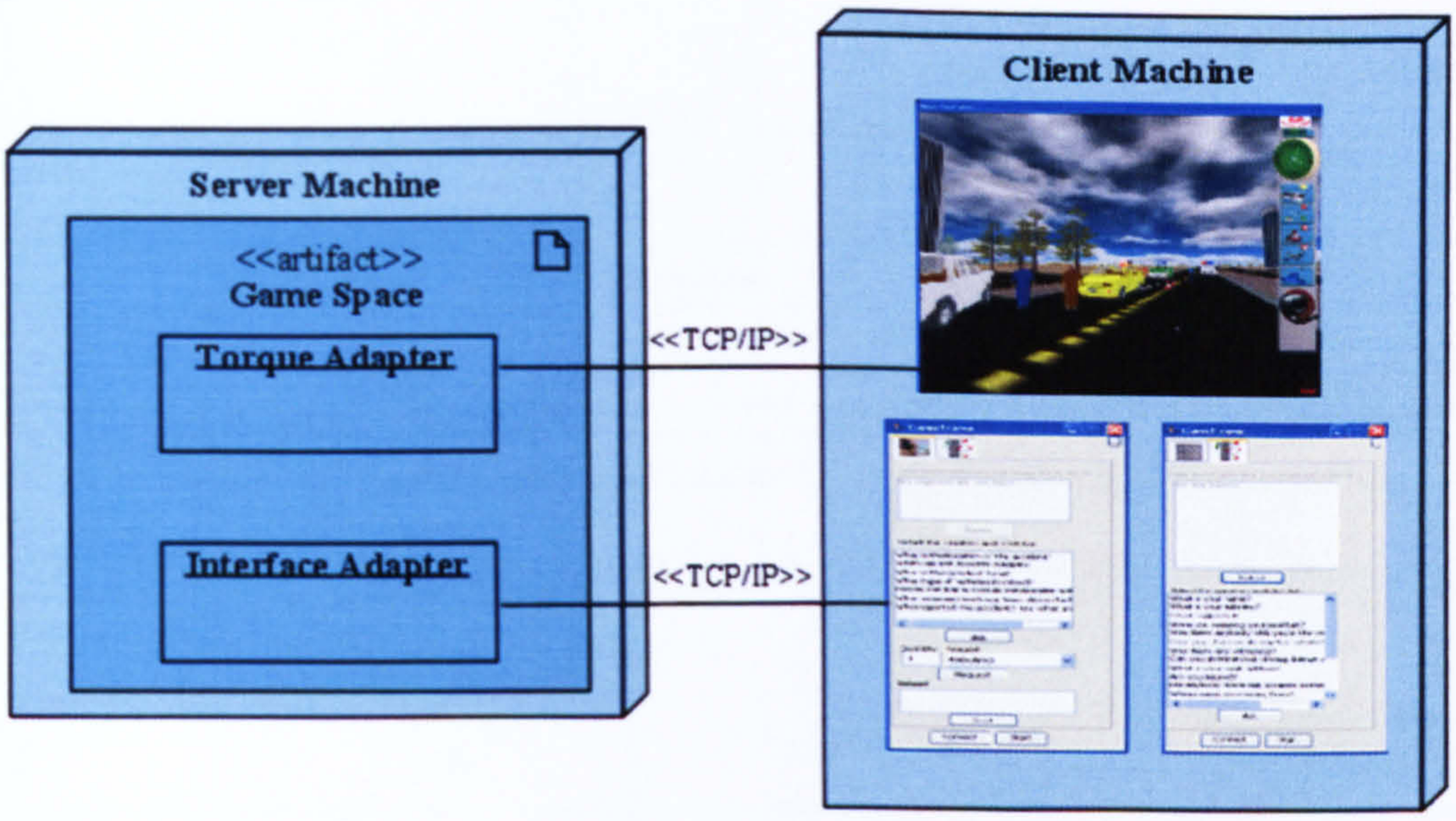


Figure C.7: Two adapters were created: one to communicate with the game engine and the other to communicate with the communications interfaces (see Figure C.2).

The adapter was sensitive to change which undermined the benefits of the dynamic object model. This is due to the manual remapping required in the adapter every time the dynamic object model is changed and that change affects the communication with the game engine. For instance the Moving class contains PersonID and ItemID properties and any simple change such as renaming these properties to Person and Item respectively would require the adapter to be manually updated as shown in Figure C.8. Since the object model is database driven the adapter can be made to use the names in the database. However the adapter's sensitivity is harder to solve automatically for other changes such as moving properties between classes.

```
ontServer.getScenario().addInstance("Moving","ID,PersonID,ItemID,FromPosition,ToPosition,tim
e",strPropertiesValues,"torqueAdapter");
```

↓

```
ontServer.getScenario().addInstance("Moving","ID,Person,Item,FromPosition,ToPosition,time",str
PropertiesValues,"torqueAdapter");
```

Figure C.8: Sample code showing the adapter's sensitivity to object model changes.

GSA's performance managed to scale quite well to SGTAI without noticeable negative effects on game playability. One of the scalability concerns was with regards to the use of interpreted scripting (T3 in Figure 4.12) for the adapter which favoured modifiability at the expense of performance. The concern was that GSA would not scale well to the increase in the number of interpreted scripts being executed. However the overhead was not noticeable in the game. The second concern was with the fourth trade-

off made (T4 in Figure 4.12) of making the object model modifiable at the expense of performance. This concern again was not noticeable.

Appendix D. Pre- and Post-Tests

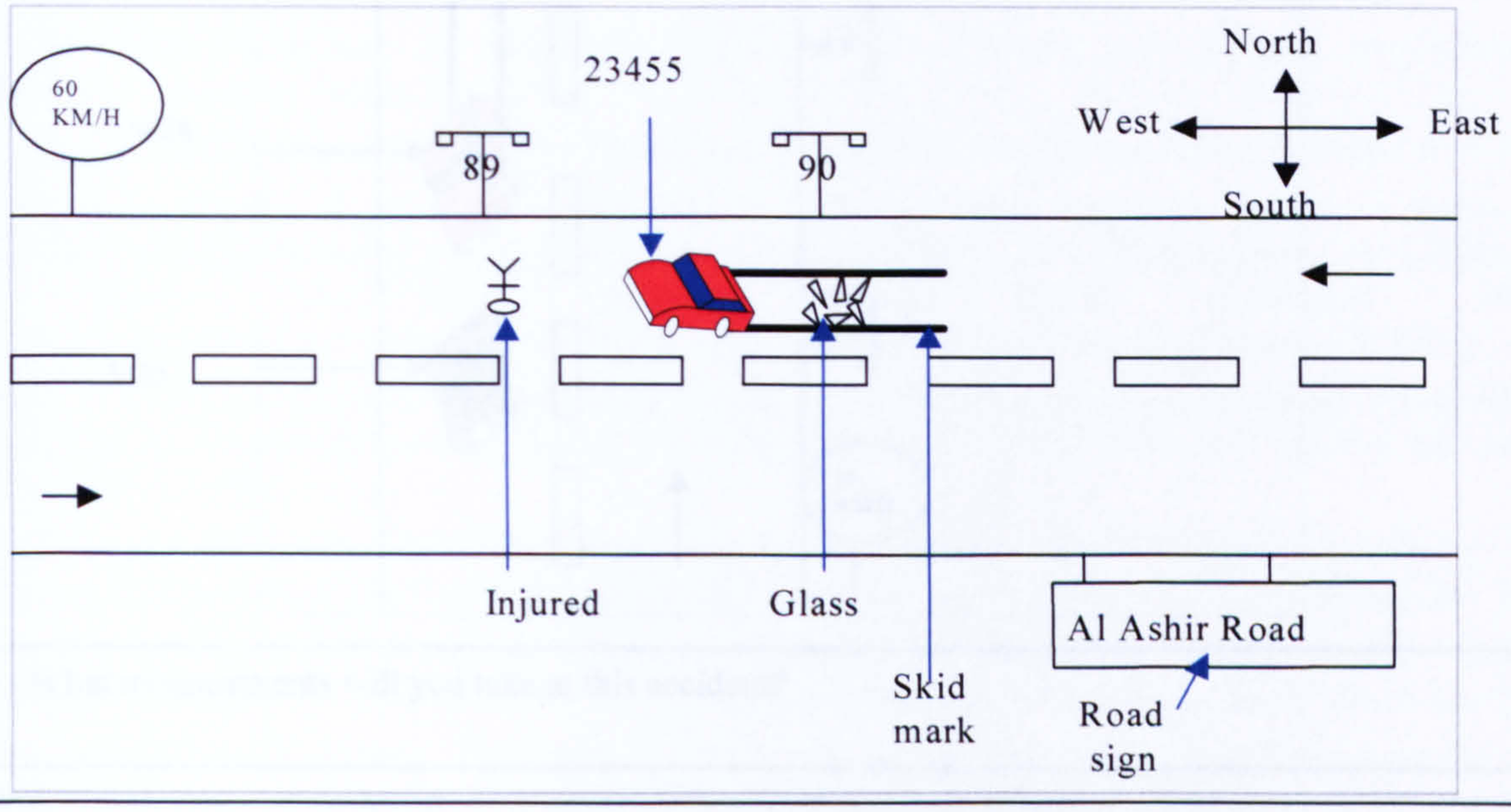
This appendix lists the pre- and post-tests used in the experiment described in chapter 6.

Pre-Test

Date:

Name:

Assume you're the first at the following accident scene. Examine the accident drawing and answer the questions below.



1- What measurements will you take at this accident?

2- Place markers at the important points on the scene using the X sign?

3- What pictures will you take at this accident?

4- Mark on the scene where you'll park your vehicle and why?

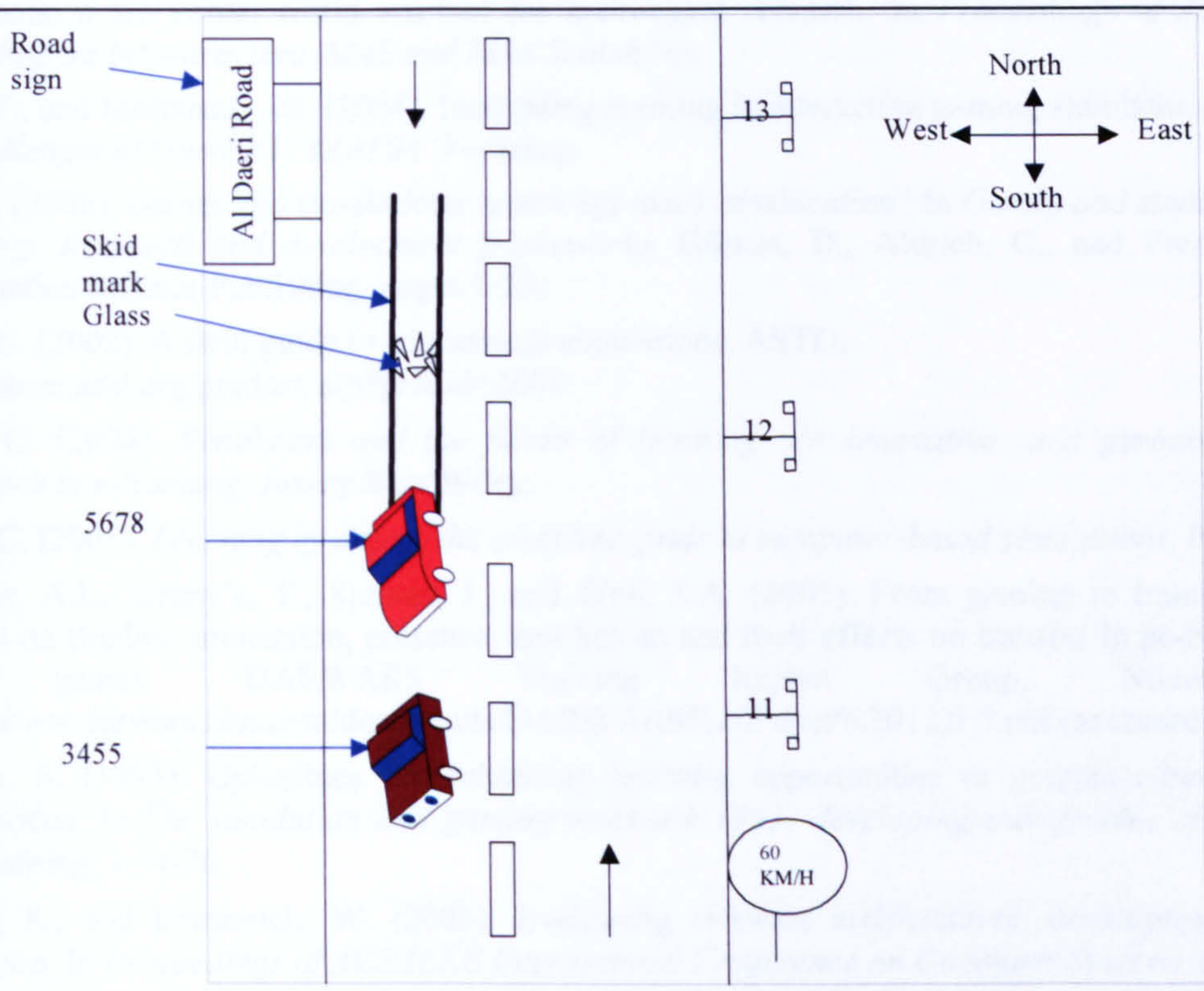
5- Place two circles on the scene to indicate where you'll place the cones?

6- Once the ambulance completes its job at the accident, the ambulance driver steps to you and asks your permission to leave the scene, what information do you require from him? (**not marked**)

Figure D.1: Pre-test.

Post-Test	
Date:	Name:

Assume you're the first at the following accident scene. Examine the accident drawing and answer the questions below.



1- What measurements will you take at this accident?

2- Place markers at the important points on the scene using the X sign?

3- What pictures will you take at this accident?

4- Mark on the scene where you'll park your vehicle and why?

5- Place two circles on the scene to indicate where you'll place the cones?

6- Once the ambulance completes its job at the accident, the ambulance driver steps to you and asks your permission to leave the scene, what information do you require from him? (**not marked**)

Figure D.2: Post-test.

References

Abt, C. C. (1970). *Serious games*. New York: Viking Press.

Adobbati, R., Marshall, A.N., Scholer, A., Tejada, S., Kaminka, G., Schaffer, S., and Sollitto, C. (2001). Gamebots: a 3D virtual world test-bed for multi-agent research. In *Proceedings of 2nd International Workshop on Infrastructure, MAS and MAS Scalability*.

Aha, D.W., and Molineaux, M. (2004). Integrating learning in interactive gaming simulators. In *Proceedings of Challenges of Game AI: AAAI'04 Workshop*.

Akilli, G. (2006). Games and simulations: a new approach in education? In *Games and simulations in online learning: Research and development frameworks*, Gibson, D., Aldrich, C., and Prensky, M. (Eds.), Information Science Publishing, pages 1-20.

Aldrich, C. (2002). A field guide to educational simulations. ASTD, http://store.astd.org/product.asp?prodid=2000

Aldrich, C. (2004). *Simulators and the future of learning: an innovative (and perhaps revolutionary) approach to e-learning*. Jossey Bass Wiley.

Aldrich, C. (2005). *Learning by doing: the complete guide to computer-based simulations*. Pfeiffer Wiley.

Alexander, A.L., Bruny'e, T., Sidman, J., and Weil, S.A. (2005). From gaming to training: a review of studies on fidelity, immersion, presence, and buy-in and their effects on transfer in pc-based simulations and games. DARWARS Training Impact Group, November 2005. http://www.darwars.bbn.com/downloads/DARWARS%20Paper%2012205.pdf (accessed 14/8/2007).

Armitage, S. (1993). Guidelines for enhancing learning opportunities in computer-based management simulations. In *The simulation and gaming yearbook 1993: developing transferable skills in education and training*. 1:14-26.

Bahsoon, R., and Emmerich, W. (2003). Evaluating software architectures: development, stability and evolution. In *Proceedings of ACS/IEEE International Conference on Computer Systems and Applications*.

Bahsoon, R., and Emmerich, W. (2006). Architectural Stability and Middleware: An Architecture Centric Evolution Perspective. In *Proceedings of ECOOP 2006 workshop on Architecture-Centric Evolution*.

Baker, J., and Fricke, L. (1986). *The traffic-accident investigation manual. At-scene investigation and technical follow-up*. Northwestern University Traffic Institute.

Bass, L., Clements, P., and Kazman, R. (1998). *Software architecture in practice*. Addison Wesley.

Beal, S. (2004). Using games for training dismounted light infantry leaders: emergent questions and lessons learned. Technical Report 1841, United States Army Research Institute for the Behavioral and Social Sciences.

Beal, S., and Christ, R. (2004). Training effectiveness evaluation of the Full Spectrum Command game. Technical Report 1140, United States Army Research Institute for the Behavioral and Social Sciences.

Becker, K. (2005). *How are learning objectives woven into the design of a serious game? Instructional design for serious games*. PhD thesis proposal, http://minkhollow.ca/KB/PhD/Thesis-Proposal-NTI.pdf (accessed 14/8/2007).

Becker, K. (2006a). Classifying learning objectives in commercial games, Proof of Concept. In *Proceedings of Canadian Games Study Association Symposium*.

Becker, K. (2006b). Design paradox: instructional games future play. In *Proceedings of International Conference on the Future of Game Design and Technology*.

Becker, K. (2006c). Pedagogy in commercial video games. In *Games and simulations in online learning: Research and development frameworks*, Gibson, D., Aldrich, C., and Prensky, M. (Eds.), Information Science Publishing, pages 21-47.

Beedle, J.B., and Wright, V.H. (2006). Perspective from multiplayer video gamers. In *Games and simulations in online learning: Research and development frameworks*, Gibson, D., Aldrich, C., and Prensky, M. (Eds.), Information Science Publishing, pages 150-174.

Belanich, J., Sibley, D.E., and Orvis, K.L. (2004). Instructional characteristics and motivational features of a PC-based game. Research Report 1822, U. S. Army Research Institute for the Behavioral and Social Sciences.

Bennell, C., and Jones, N. (2003). The effectiveness of use of force simulation training. Carleton University, Department of Psychology. http://www.cprc.org/tr/tr-2005-01.pdf (accessed 14/8/2007).

Berndt, C., Watson, I., and Guesgen, H. (2005). OASIS: an open AI standard interface specification to support reasoning, representation and learning in computer games. In *Proceedings of Workshop for International Joint Conference on Artificial Intelligence (IJCAI-05), Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 19-24.

Bilas, S. (2000). Postmortem: Sierra Studios' Gabriel Knight 3: Blood of the Sacred, Blood of the Damned. *Game Developer*, June.

Bilas, S. (2002). A data-driven game object system. *In Proceedings of Game Developers Conference*. http://www.drizzle.com/~scottb/gdc/game-objects.ppt (accessed 5/5/2007).

Bilas, S. (2003). The Continuous World of Dungeon Siege. Gas Powered Games. www.drizzle.com/~scottb/gdc/continuous-world.htm (accessed 14/8/2007).

BinSubaih, A., Maddock, S., and Romano, D. (2004a). A collaborative virtual training architecture for investigating the aftermath of vehicle accidents. *In Proceedings of Middle Eastern Simulation and Modelling Conference*, pages 170-178.

BinSubaih, A., Maddock, S., and Romano, D. (2004b). A domain-independent multiplayer architecture for training. In *Proceedings of International Workshop in Computer Game Design and Technology*, pages 144-151.

BinSubaih, A., Maddock, S., and Romano, D. (2004c). An architecture for domain-independent collaborative virtual environments. In *Proceedings of 5th Annual European GAME-ON Conference*, pages 84-88.

BinSubaih, A., Maddock, S., and Romano, D. (2005a). Comparing the use of a tabletop and a collaborative desktop virtual environment for training police officers to deal with traffic accidents: Case study. In *Proceedings of International Conference on Engineering Education*, 2:94-100.

BinSubaih, A., Maddock, S., and Romano, D. (2005b). Game logic portability. In *Proceedings of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology ACE 2005, Computer Games Technology session*, pages 458-461.

BinSubaih, A., Maddock, S., and Romano, D. (2005c). OntRAT: ontology-based rules acquisition tool. In *Proceedings of m-ICTE2005, 3rd International Conference on multimedia and Information & Communication Technologies in Education*, pages 978-983.

BinSubaih, A., and Maddock, S. (2006). Using ATAM to evaluate a game-based architecture. *Workshop on Architecture-Centric Evolution (ACE 2006), hosted at the 20th European Conference on Object-Oriented Programming ECOOP 2006.*

BinSubaih, A., Maddock, S., and Romano, D. (2006a). An architecture for portable serious games. *Doctoral Symposium, hosted at the 20th European Conference on Object-Oriented Programming ECOOP 2006.*

BinSubaih, A., Maddock, S., and Romano, D. (2006b). A serious game for traffic accident investigators. Special Issue of International Journal of Interactive Technology and Smart Education on "Computer Game-based Learning.", 3(4):329-346.

BinSubaih, A., and Maddock, S. (2007). G-factor portability in game development using game engines. In *Proceedings of 3rd International Conference on Games Research and Development 2007* (CyberGames 2007), pages 163-170.

BinSubaih, A., Maddock, S., and Romano, D. (2007). A survey of 'game' portability. Department of Computer Science Technical Report CS-07-05, 2007, University of Sheffield.

BinSubaih, A., Maddock, S., and Romano, D. (2008). Developing a serious game for police training. In *Handbook of Research on Effective Electronic Gaming in Education*, Ferdig, R.E. (Ed.) Information Science Reference (In press).

Blackman, S. (2005). Serious games..and less!. *SIGGRAPH Computer Graphics*, 39(1):12-16.

Bourg, D. (2006). *Excel Scientific and Engineering Cookbook.* O'Reilly Media, Inc.

Brandon, B. (2005). Insights: e-learning gurus, challenges, and solutions. *The eLearning Developers' Journal*, March 28, 2005, pages 1-9.

Brogni, A., Slater, M., and Steed, A. (2003). More breaks less presence. In *Proceedings of 6th Annual International Workshop on Presence.*

Buch, T., and Egenfeldt-Nielsen, S. (2006). The learning effect of Global Conflicts: Palestine. In *Proceedings of Media@Terra conference.*

Burbeck, E.L. (1987). *The validity of the selection interview for recruits to the Metropolitan police force.* PhD thesis. London: University College.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *Pattern-oriented software architecture: a system of patterns.* Volume 1, John Wiley and Sons.

Carey, R. (2007). Serious game engine shootout: a comparative analysis of technology for serious game development. Serious Games Source, 21/2/2007, http://seriousgamessource.com/features/feature_022107_shootout_1.php (accessed 14/8/2007).

Carless, S. (2007). Rise of the game engine. *Game Developer*, April, 2007, pages 2-2.

Casanueva, J., and Blake, E. (2001). The effects of avatars on co-presence in a collaborative virtual environment. In *Proceedings of Annual Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT2001).*

Cavazza, M., Charles, F., and Mead, S.J. (2002). Interacting with virtual characters in interactive storytelling. In *Proceedings of 1st International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS '02), pages 318-325.

Cavazza, M., Hartley, S., Lugrin, J., and Le Bras, M. (2004). Qualitative physics in virtual environments. In *Proceedings of 9th International Conference on Intelligent User interfaces (IUI '04),* pages 54-61.

Chan, K., Spagnolo, S., Stevens, S., Hagger, N., Chau, D., and Carlton, G. (2003). Postmortem: Blue Tongue Software's Jurassic Park: Operation Genesis. Gamasutra, 17/3/2003, http://www.gamasutra.com/features/20030317/chan_01.shtml (accessed 14/8/2007).

Chandrasekaran, B., Josephson, J., and Benjamins, V. (1999). What are ontologies and why do we need them? *IEEE Intelligent Systems*, 14(1):20-26.

Chao, D. (2001). Doom as an interface for process management. In *Proceedings of SIGCHI Conference on Human Factors in Computing Systems (CHI '01)*, pages 152-157.

Chathman, R. (2005). A tale of training superiority, games, and people stuff. In *Proceedings of DARPATech 2005*, pages 51-55.

Charsky, D. (2006). Utilizing game elements for learning. http://idt.memphis.edu/podcasts/ppt/GameElements.ppt (accessed 14/8/2007).

Chee, Y.S. (2002). Refocusing learning on pedagogy in a connected world. *On the Horizon*, 10(4):7-13.

Chen, S., and Michael, D. (2005). Proof of learning: assessement in serious games. Gamasutra, 19/10/2005. http://www.gamasutra.com/features/20051019/chen_01.shtml (accessed 14/7/2007).

Chi, M., Glaser, R., and Farr, M. (1988). *The nature of expertise.* Lawrence Erlbaum.

Chwif, L., and Barretto, M.R.P. (2003). Simulation models as an aid for the teaching and learning process in operations management. In *Proceedings of Winter Simulation Conference,* 2:1994- 2000.

Clements, P. (2000). Active reviews for intermediate designs. Technical Report CMU/SEI-2000-TN-009, Software Engineering Institute.

Clements, P., Kazman, R., and Klein, M. (2001). *Evaluating software architectures: methods and case studies*, Addison Wesley.

Coffield, F., Moseley, D., Hall, E., and Ecclestone, K. (2004). Learning styles and pedagogy in post-16 learning: a systematic and critical review. Learning and Skills Research Centre. http://www.lsda.org.uk/files/PDF/1543.pdf (accessed 14/8/2007).

Cohen, J. (1988). Statistical power analysis for the behavioral sciences. Lawrence Erlbaum Associates.

Corti, K. (2006). Gamesbased learning. A serious business application PIXELearning Ltd, Feb, 2006.

Coyle, D., and Matthews, M. (2004). Personal Investigator: a therapeutic 3D game for teenagers. In *Proceedings of Social Learning Through Gaming Workshop, CHI 2004 Conference on Human Factors in Computing Systems.*

Crawford, C. (1984). *The art of computer game design.* Mcgraw-Hill Osborne Media.

Creel, J., Maslov, A., Mikeal, A., and Speight, C. (2006). Information and Decision-making in Immersive Digital Environments. Computer Science Department, Texas A&M University. http://adammikeal.org/courses/cnm/files/Project%20Final%20Report.pdf (accessed 14/8/2007).

Daggash, B. (2006). 3D Training for Traffic Cops. The Emirates Evening Post, Jan 30, pages 6-6

Dark, M., and Winstead, J. (2005). Using educational theory and moral psychology to inform the teaching of ethics in computing. In *Proceedings of 2nd Annual Conference on Information Security Curriculum Development*, pages 27-31.

Darken, C., Morgan, J., and Paull, G. (2004). Efficient and dynamic response to fire. In *Proceedings of AAAI 04 Challenges in Game AI workshop*, pages 44-48.

Davies, N.P., Mehdi, Q.H., and Gough, N. (2005). Creating and visualising an intelligent NPC using game engines and AI tools. In *Proceedings of 19th European Conference on Modelling and Simulation.*

Davis, M., Shilling, R., Mayberry, A., Bossant, P., McCree, J., Dossett, S., Buhl, C., Chang, C., Champlin, E., Wiglesworth, T., and Zyda, M. (2004). Making America's Army. In America's Army PC Game Vision and Realization, Davis, M., US Army and the Moves Institute, pages 9-15.

De Lisi, R., and Wolford, J.L. (2002). Improving children's mental rotation accuracy with computer game playing. *Journal of Genetic Psychology*, 163:272-282.

De Villiers, M. R. (2005). Three approaches as pillars for interpretive information systems research: development research, action research and grounded theory. In *Proceedings of the 2005 Annual Research Conference of the South African institute of Computer Scientists and information Technologists on IT Research in Developing Countries*, Vol 150, pages 142-151.

DeLeon, V., and Berry, R. (2000). Bringing VR to the desktop: are you game? *Multimedia, IEEE* 7(2):68-72.

Dempsey, J.V., Rasmussen, K., and Lucassen, B. (1994). Instructional gaming: implications for instructional technology. In *Proceedings of Annual Meeting of the Association for Educational Communications and Technology.*

Denis, G., and Jouvelot, P. (2005). Motivation-driven educational game design: applying best practices to music education. In *Proceedings of 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE '05)*, 265:462-465.

Dieleman, H., and Huisingh, D. (2006). The potentials of games in learning and teaching about sustainable development. *Journal of Cleaner Production, Special Issue on Education for Sustainable Development*, 14: 9-11.

Diller, D., Roberts, B., and Willmuth, T. (2005). DARWARS Ambush! A case study in the adoption and evolution of a game-based convoy trainer by the U.S. Army. http://ms.ie.org/SIW_LOG/05F/05F-SIW-052.pdf (accessed 14/8/2007).

Dobby, J., Anscombe, J., and Tuffin, R. (2004). Police Leadership: Expectations and Impact. Home Office Online Report RDS OLR 20/04, Home Office, London. http://www.homeoffice.gov.uk/rds/pdfs04/rdsolr2004.pdf (accessed 14/8/2007).

Dobnik, V., (2004). Surgeons may err less by playing video games. Associated Press, 7/4/2004, http://www.msnbc.msn.com/id/4685909/ (accessed 14/8/2007).

Doolittle, J.H. (1995). Using riddles and interactive computer games to teach problem-solving skills. *Teaching of Psychology*, 22(1):33-36.

Dorval, M., and Pepin, M. (1986). Effect of playing a video game on a measure of spatial visualization. Perception and Motor Skills, 62, 159-162.

Dounis, E. (2006). The great debate: gameplay vs. graphics. GamersMark.com, 7/9/2006, http://www.gamersmark.com/articles/205/ (accessed 31/1/2007).

Duran, A. (GDC 2003). Building object systems. In *Proceedings of Game Developers Conference 2003.*

Egenfeldt-Nielsen, S., (2005). *Beyond edutainment: exploring the educational potential of computer games.* Phd thesis, IT-University Copenhagen.

Eliens, A., and Bhikharie, S.V. (2006). Game VU developing a masterclass for high-school students using the Half-life 2 SDK. In *Proceedings of GAME'ON-NA'2006.*

Elliott, J., and Bruckman, A. (2002). Design of a 3D interactive math learning environment. In *Proceedings of Conference on Designing interactive Systems: Processes, Practices, Methods, and Technique*, pages 64-74.

ESA (2006) Essential facts about the computer and video game industry. http://www.theesa.com/archives/files/Essential%20Facts%202006.pdf (accessed 14/8/2007).

Feinstein, A.H., Mann, S., and Corsun, D.L. (2002). Charting the experiential territory: clarifying definitions and uses of computer simulations, games, and role play. *Journal of Management Development*, 21(10):732-744.

Fermier, R. (2002). Creating a data driven engine: case study: The Age Of Mythology. In *Proceedings of Game Developers Conference 2002.*

Fielding, D., Fraser, M., Logan, B. and Benford, S. (2004). Extending game participation with embodied reporting agents. In *Proceedings of 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, 74:100-108.

Fong, G. (2004). Adapting cots games for military simulation. In *Proceedings of 2004 ACM SIGGRAPH international Conference on Virtual Reality Continuum and Its Applications in industry (VRCAI '04)*, pages 269-272.

Foreman, J. (2003). Next-generation: educational technology versus the lecture. Educause Review, pages 12-22.

Fristrom, J. (2000). Postmortem: Treyarch's Draconus. Gamasutra, 14/8/2000, http://www.gamasutra.com/features/20000814/fristrom_01.htm (accessed 14/8/2007).

Fritz, E. (2007). IAC Acquires Majority Share of GarageGames, Announces Web-Based Gaming Platform InstantAction.com, http://www.garagegames.com/news/13587 (accessed 6/10/2007).

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software.* Addison-Wesley.

Garces, D. (2006). Scripting language survey. In *Game Programming Gems 6*, Charles River Media, pages 323-340.

Garneau, P. (2001). Fourteen forms of fun. Gamasutra, 12/10/2001, http://www.gamasutra.com/features/20011012/garneau_01.htm (accessed 14/8/2007).

Gaudiosi, J. (2005). Next generation game concepts. BusinessWeek, 1/12/2005, http://www.businessweek.com/innovate/content/dec2005/id20051201_193273.htm (accessed 14/8/2007).

Gee, J. (2003). *What video games have to teach us about learning and literacy.* New York: Palgrave/Macmillan.

Gee, J., and Prensky, M. (2006). *Don't bother me mom -- I'm learning!* Paragon House Publishers.

Goslin, M. (2004) Postmortem: Disney Online's Toontown. Gamasutra, 28/1/2004, http://www.gamasutra.com/features/20040128/goslin_01.shtml (accessed 14/8/2007).

Goslin, M., and Mine, M.R. (2004). The Panda3D graphics engine. *Computer* 37(10):112-114.

Green, C.S., and Bavelier, D. (2003). Action video game modifies visual selective attention. *Nature*, 423: 534-537.

Greiner, B. (2005). Game industry finds serious outlet for creative energies. The Washington Post, 31/10/2005,http://www.washingtonpost.com/wp-dyn/content/article/2005/10/30/AR2005103000578_pf.html (accessed 14/8/2007).

Griffith, J.L., Voloschin, P., Gibb, G.D., and Bailey, J.R. (1983). Differences in eye-hand motor coordination of videogame users and non-users. *Perceptual and Motor Skills*, 57:155-158.

Gunter, G., Kenny, R., and Vick, E. (2006). A case for formal design paradigm for serious games. In *Proceedings of International Digital Media and Arts Association.*

Habgood, M.P.J., Ainsworth, S.E., and Benford, S. (2005). Endogenous fantasy and learning in digital games. Simulation & Gaming, 36:483-498.

Haidarh, A. (2006). Ahmed BinSubaih Trains Traffic Investigators in Virtual Streets (Translated from Arabic), Al-Amn Magazine, April 2006, pages 42-45.

Harz, C. (2006). Games for learning: serious entertainment. Animation World Magazine, 26/4/2006, http://mag.awn.com/index.php?ltype=pageone&article_no=2863 (accessed 14/8/2007).

Heckenberg, S.G., Herbert, R.D., and Webber, R. (2004). Visualisation of the minority game using a mod. In *Proceedings of 2004 Australasian Symposium on Information Visualisation,* 35:157-163.

Herselman, M.E. (1999). South African resource-deprived learners benefit from CALL through the medium of computer games. *Computer Assisted Language Learning*, 12(3):197–218.

Herz, J.C., and Macedonia, M. (2002). Computer games and the military: two views. *Defense Horizons*, 11:1-8, especially page 7.

Hoffman, E. (2006). Games for Health 2006: Pulse!! first-person healthcare system simulation. Serious Games Source, 25/10/2006, http://seriousgamessource.com/features/feature_102506_pulse_2.php (accessed 14/8/2007).

Huang, W., Huang, W., Diefes-Dux, H., and Imbrie, P.K. (2006). A preliminary validation of attention, relevance, confidence and satisfaction model-based instructional material motivational survey in a computer-based tutorial setting. *British Journal of Educational Technology*, 37(2): 243-259.

Hunicke, R., and Chapman, V. (2004). AI for Dynamic Difficulty Adjustment in Games. In *Proceedings of Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence (AAAI '04).*

Hussain, T.S., and Vidaver, G. (2006). Flexible and purposeful NPC behaviors using real-time genetic control. In *Proceedings of 2006 World Congress on Computational Intelligence*, pages 785-792.

Issa, W. (2006). Dubai traffic will be smooth within three years. Gulf News, 20/10/2006. http://archive.gulfnews.com/indepth/trafficwatch/Traffic_initiatives/10076079.html (accessed 14/8/2007).

Iuppa, N., and Borst, T. (2007). *Story and Simulations for Serious Games: Tales from the Trenches*. Focal Press.

Jacobs, S., Ferrein, A., and Lakemeyer, G. (2005). Unreal Golog Bots. In *Proceedings of Workshop for International Joint Conference on Artificial Intelligence (IJCAI-05), Workshop on Reasoning, Representation, and Learning in Computer Games.*

Jankovic, L. (2000). Games Development in VRML. *Virtual Reality*, 5:195-203.

Jenkins, H., Klopfer, E., Squire, K., and Tan, P. (2003). Entering the education arcade. *Computers in Entertainment (CIE)*, 1(1):17-17.

Jerz, D.G. (2005). A debate between Jan Cannon-Bowers and Marc Prensky (Jerz's Literacy Weblog) Serious Games Summit DC 2005, Day 2. http://jerz.setonhill.edu/weblog/permalink.jsp?id=3835 (accessed 14/8/2007).

Johnson, W.L., Marsella, S. and Vilhjalmsson, H. (2004). The DARWARS tactical language training system. In *Proceedings of Interservice/Industry Training, Simulation and Education Conference (I/ITSEC).*

Joyce, L. (2005). Military apps drive simulation. *R&D Magazine*. Vol 47.

Kapolka, A. (2003). The extensible run-time infrastructure (XRTI): an emerging middleware platform for interoperable networked virtual environments. In *Proceedings of Lake Tahoe Workshop on Collaborative Virtual Reality and Visualization.*

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., and Carriere, J. (1998). The architecture tradeoff analysis method. In *Proceedings of 4th IEEE International Conference on Engineering of Complex Computer Systems*, pages 68-78.

Keith, C. (2003). From the ground up: creating a core technology group. Gamasutra, 1/8/2003, http://www.gamasutra.com/features/20030801/keith_01.shtml (accessed 14/8/2007).

Keller-McNulty, S., Bellman, K., Carley, K., Davis, P., Ivanetich, R., Laskey, K., Loftin, R., Maddox, D., McBride, D., McGinnis, M., Pollock, S., Pratt, D., Robinson, S., vonWinterfeldt, D., Zyda, M., Weidman,

S., Glassman, N., & Wright, B. (2006). *Report defense modeling, simulation, and analysis: meeting the challenge committee on modeling and simulation for defense transformation.* National Research Council.

Khoo, A., Dunham, G., Trienens, N., and Sood, S. (2002). Efficient, realistic NPC control systems using behavior-based techniques. In *Proceedings of AAAI 2002 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment.*

Kirriemuir, J. (2002). The relevance of video games and gaming consoles to the higher and further education learning experience. Techwatch Report TSW 02.01, Joint Information Systems Committee (JISC).

Kirriemuir, J., and McFarlane, A. (2004). Literature review in games and learning. Bristol: NESTA Futurelab.

Ko, S. (2002). An empirical analysis of children's thinking and learning using a computer game context. *Educational Psychology*, 22(2):219–233.

Laarni, J., Ravaja, N., and Saari, T. (2005) Presence experience in mobile gaming. In *Proceedings of DIGRA 2005.*

Laird, J. (2001). It knows what you're going to do: adding anticipation to a Quakebot. In *Proceedings of 5th International Conference on Autonomous Agents*, pages 385-392.

Laird, J.E., Assanie, M., Bachelor, B., Benninghoff, N., Enam, S., Jones, B., Kerfoot, A., Lauver, C., Magerko, B., Sheiman, J., Stokes, D., and Wallace, S. (2002). A testbed for developing intelligent synthetic characters. In *Artificial Intelligence and Interactive Entertainment: Papers from the 2002 AAAI Spring Symposium,* pages 52-56.

Lambie, C. (2006). Troops to train on video games. Canoe Inc, 2/6/2006, http://wham.canoe.ca/news/2006/06/02/1611922.html (accessed 14/8/2007).

LaMothe, A. (2002). Letter from the series editor. In Varanese, A. *Game Scripting Mastery*, Premier Press.

Lattanze, T. (2001). My experiences with the ATAM. In Clements, P., Kazman, R., and Klein, M. (2001). *Evaluating software architectures: methods and case studies*, Addison Wesley, pages 104-105.

Le Hy, R., Arrigoni, A., Bessi ere, P., and Lebeltel, O. (2004). Teaching Bayesian behaviors to video game characters. *Robotics and Autonomous Systems*, 47:177-185.

LeGrand, S., and Freedman, G. (1988). Software engineering and Ada training: an implementation model for NASA. In *Proceedings of 5th Washington Ada Symposium on Ada (WADAS '88)*, pages 123-132.

Lenoir, T. (2003). Taming a disruptive technology. In *Proceedings of Symposium on the Coevolution of Technology-Business Innovation.*

Leonard, T. (1999). Postmortem: Looking Glass's Thief: The Dark Project. *Game Developer*, July.

Lester, C. (2000). *Motivational change among police constables: A case study of the Metropolitan police service.* PhD thesis, London School of Economics, 2000.

Lewis, M., and Jacobson, J. (2002) Game engines in scientific research. *Communications of the Association for Computing Machinery (CACM)*, 45(1):27-31.

MacNamee, B., Rooney, P., Lindstrom, P., Ritchie, A., Boylan, F., Burke, G. (2006). Serious Gordon: Using Serious Games to Teach Food Safety in the Kitchen. In *Proceedings of 9th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games (CGAMES06).*

Magennis, S., and Farrell, A. (2005). Teaching and learning activities: expanding the repertoire to support student learning. In *Emerging Issues in the Practice of University Learning and Teaching*, O'Neill, G., Moore, S., McMullin, B. (Eds.), AISHE Readings.

Malenfant, D. (2000). Postmortem: Shiny's Wild 9. Gamasutra, 7/1/2000, http://www.gamasutra.com/features/20000107/wild9_01.htm (accessed 14/8/2007).

Malone, T.W., and Lepper, M.R. (1987). Making Learning Fun: A Taxonomy of Intrinsic Motivations for Learning. In *Aptitude, Learning and Instruction: III. Conative and affective process analyses*, Snow, R.E., and Farr, M.J. (Eds.), Hilsdale, NJ: Erlbaum, pages 223-253.

Mantovani, F. (2001). VR learning: potential and challenges for the use of 3D environments in education and training. In *Towards CyberPsychology: Mind, Cognitions and Society in the Internet Age*, Riva, G., and Galimberti, C. (Eds.), Amsterdam, IOS Press.

Mayer, R.E. (2001). Multimedia Learning. Cambridge University Press.

McFarlane, A., Sparrowhawk, A., and Heald, Y. (2002). Report on the educational use of games: An exploration by TEEM on the contribution which games can make to the educational process. Cambridge: TEEM.

McGrath, D., and Hill, D. (2004). UnrealTriage: a game-based simulation for emergency response. In *Proceedings of Huntsville Simulation Conference.*

Mergel, B. (1998). Instructional design and learning theory. Educational Communications and Technology, University of Saskatchewan.

Michael, D., and Chen, S. (2005). *Serious games: games that educate, train, and inform.* Course Technology.

Mitchell, A., and Savill-Smith, C. (2004). The use of computer and video games for learning: a review of the literature. London, UK: Learning and Skills Development Agency. http://www.lsda.org.uk/files/PDF/1529.pdf (accessed 14/8/2007).

Molenda, M. (2003). In search of the elusive ADDIE model. *Performance improvement*, 42(5):34-34.

Morrison, G., Ross, S., and Kemp, J. (2003). *Designing effective instruction.* John Wiley & Sons.

Muñoz-Avila, H., and Aha, D. (2004). On the Role of Explanation for Hierarchical Case-Based Planning in Real-Time Strategy Games. In *Proceedings of ECCBR-04 Workshop on Explanations in CBR.*

Narayanasamy, V., Wong, K.W., Fung, C.C., and Rai, S. (2006). Distinguishing games and simulation games from simulators. *Computers in Entertainment (CIE)*, 4:2-2.

Nareyek, A., Combs, N., Karlsson, B., Mesdaghi, S., and Wilson, I. (2005). The 2005 report of the IGDA's artificial intelligence interface standards committee. International Game Developers Association, http://www.igda.org/ai/report-2005/report-2005.html (accessed 14/8/2007).

Nichani, M. (2001). Exclusive interview with Marc Prensky. March 15, 2001. http://www.elearningpost.com/articles/archives/exclusive_interview_with_marc_prensky/ (accessed 14/8/2007).

Nielsen-Englyst, L. (2003). Game Design for Imaginative Conceptualisation. In *Proceedings of 7th International Workshop on Experimantal Interactive Learning in Industrial Management*, pages 149-164.

Nord, R. (2001). Making the most of evaluator's efforts. In Clements, P., Kazman, R., and Klein, M. (2001). *Evaluating software architectures: methods and case studies*, Addison Wesley, pages 97-98.

Ogle, T. (2002). The *effects of virtual environments on recall in participants of differing levels of field dependence.* PhD thesis, Virginia Polytechnic and State University.

Oliveira, M., Crowcroft, J., and Slater, M. (2003). An innovative design approach to build virtual environment systems. In *Proceedings of Workshop on Virtual Environments 2003, ACM International Conference Proceeding Series*, 39:143-151.

Ota, M. (2003). Extending the AI in a Commercial Game Engine. Bachelor thesis, School of Information Technology and Electrical Engineering, The University of Queensland.

Pair, J., Allen, B., Dautricourt, M., Treskunov, A., Liewer, M., Graap, K., Reger, G. and Rizzo, A. (2006). A virtual reality exposure therapy application for Iraq war post. In *Proceedings of IEEE VR2006 Conference*, pages 64-71.

Papert, S. (1991). Situating constructionism. In *Constructionism*, Norwood, Harel, I., and Papert, S. (Eds.), NJ: Ablex Publishing, pages 1-12.

Parnas, D.L., and Weiss, D. (1987). Active design reviews: principles and practices. *Journal of Systems and Software*, 7:259-265.

Patrick, E., Cosgrove, D., Slavkovic, A., Rode, J., Verratti,T., and Chiselko, G. (2000). Using a large projection screen as an alternative to head-mounted displays for virtual environments. In *Proceedings of SIGCHI Conference on Human Factors in Computing Systems*, pages 478-485.

Pillay, H., Brownlee, J., and Wilss, L. (1999). Cognition and recreational computer games: implications for educational technology. *Journal of Research on Computing in Education*, 32(1):203–216.

Ponsen, M., Lee-Urban, S., Mu˙oz-Avila, H., Aha, D., and Molineaux, M. (2005). Stratagus: an open-source game engine for research in real-time strategy games. In *Proceedings of Workshop for International Joint Conference on Artificial Intelligence (IJCAI-05).*

Prensky, M. (2001). *Digital game-based learning*. New York: McGraw-Hill.

Prensky, M. (2004). Interactive pretending: An Overview of Simulation. http://www.marcprensky.com/writing/Prensky-Interactive_Pretending.pdf (accessed 14/8/2007).

Psotka, J., Black, B., and Hom, D. (2004). Symposium on PC-based simulations and gaming for military training. ARI Research Product 2005-01, U.S. Army Research Institute for the Behavioral and Social Sciences.

Qiu, X. (2005). *Message-based MVC architecture for distributed and aesktop applications.* PhD thesis. Syracuse University.

Re-Mission™. (2006). Outcomes study: a research trial of a video game shows improvement in health-related outcomes for young people with cancer. http://www.hopelab.org/remission.html (accessed 14/8/2007).

Ricci, K.E. (1994). The use of computer-based videogames in knowledge acquisition and retention. *Journal of Interactive Instruction Development*, 7(1): 17-22.

Riehle, D., Tilman, M., and Johnson, R. (2005). Dynamic object model. In *Pattern Languages of Program Design 5*. Manolescu, D., Völter, M., and Noble, J. (Eds.). Addison-Wesley.

Ritchie, D., and Dodge, B. (1992). Integrating technology usage across the curriculum. In *Proceedings of Annual Conference on Technology and Teacher Education.*

Roberts, B., Diller, D., and Schmitt, D. (2006). Factors affecting the adoption of a training game. In *Proceedings of 2006 Interservice/Industry Training, Simulation, and Education Conference.*

Robertson, J., and Good, J. (2003). Ghostwriter: a narrative virtual environment for children. In *Proceedings of 2003 Conference on Interaction design and children*, pages 85-91.

Rosser, J.C., Lynch, P.J., Cuddihy, L., Gentile, D.A., Klonsky, J., and Merrell, R. (2007). The impact of video games on training surgeons in the 21st century. *Archives of Surgery*, 142(2):181-186.

Rouse, R. (1999). Leaping Lizard's Centipede 3D. Gamasutra, 10/9/1999, http://www.gamasutra.com/features/19990910/centipede_01.htm (accessed 14/8/2007).

Roussou, M. (2004). Learning by doing and learning through play: an exploration of interactivity in virtual environments for children. *Computers in Entertainment (CIE)*, 2(1):10-10.

RTA, (2005). Traffic accident facts in Dubai – Year 2005, Roads & Transport Authority, Traffic Department, Dubai.

Russell, C.K. (2005). A taxonomy of serious games for education in the healthcare professions. In *Proceedings of NMC Online Conference on Educational Gaming.*

Ryan, M., Hill, D., and McGrath, D. (2005). Simulation interoperability with a commercial game engine. In *Proceedings of European Simulation Interoperability Workshop 2005.*

Sachs, J. (2001). A path model for adult learner feedback. *Educational Psychology*, 21:267-275.

Sandford, R., and Williamson, B. (2005). Games and learning. A handbook from Futurelab 2005.

Sankaran, S.R., and Bui, T. (2001). Impact of learning strategies and motivation on performance: a study in web-based instruction. *Journal of Instructional Psychology*, 28:191-198.

Schell, J., and Shochet, J. (2001). Designing interactive theme park rides: lessons from Disney's Battle for the Buccaneer Gold. Gamasutra, 6/6/2001, http://www.gamasutra.com/features/20010706/schell_01.htm (accessed 14/8/2007).

Schertenleib, S. (2006). Designing a multilayer, pluggable AI engine. In *Game Programming Gems 6*, Charles River Media, pages 291-305.

Seymour, G.O., Stahl, J.M., Ingram, J.L., and Smith, R.F. (1994). Modifying law enforcement training simulators for us in basic research. *Behavior, Research Methods, Instruments, & Computers*, 26(2):266-268.

Shumaker, S. (2002). Techniques and strategies for data-driven design in game development. http://ai.eecs.umich.edu/soar/Classes/494/talks/Schumaker.pdf (accessed 14/8/2007).

Simon, S. (2005). Real experiences in a virtual world. *Law Enforcement Technology*, June, 2005 http://www.forterrainc.com/news/news_let.html (accessed 14/8/2007).

Sinclair, S.D. (2000). *Importance of work relations and personality to morale, work performance and wellbeing among police officers*. PhD thesis, University of Aberdeen.

Singh, I., Stearns, B., Johnson, M., and the Enterprise Team. (2002). *Designing enterprise applications with the J2EETM platform*. Addison-Wesley.

Slater, M. (1999). Measuring presence: a response to the Witmer and Singer questionnaire. *Presence: Teleoperators and Virtual Environments*, 8(5):560-566.

Smith, M.K. (2001). David A. Kolb on experiential learning. The encyclopedia of informal education, http://www.infed.org/biblio/b-explrn.htm (accessed 14/8/2007).

Smith, R. (1998). Essential techniques for military modeling and simulation. In *Proceedings of 30th Conference on Winter Simulation*, pages 805-812.

Smith, R.D. (2000). Strategic directions for distributed simulation. *Simulation 2000 Series*, 2:1-9.

Spronck, P. (2005). *Adaptive game AI*. PhD thesis, Maastricht University Press, Maastricht, The Netherlands.

Stang, B. (2003). Game engines: features and possibilities. Institute of Information and Mathematical Modeling, The Technical University of Denmark (IMM DTU), http://www.garagegames.com/uploaded/GameEngines2.pdf (accessed 14/8/2007).

Stanley, K., Bryant, B., and Miikkulainen, R. (2005). Real-time Neuroevolution in the NERO Video Game. *IEEE Transactions on Evolutionary Computation*, 9(6):653-668.

Stapleton, A. (2005). Game issue report. Korea IT Industry Promotion Agency, Serial No. 25.

Stokes, B. (2005). Videogames have changed: time to consider serious games? *The Development Education Journal*, 11(2).

Stone, B. (2005). Serious gaming. *Defence Management Journal*, Issue 31.

Stone, R.J., and Gleave, P. (1998). Crime Conquest: Societal VR-Based Training for Schoolchildren. In *Proceedings of Virtual Reality in Education & Training (VRET '98)*.

Susi, T., Johannesson, M., and Backlund, P. (2007). *Serious games: an overview*. Technical Report HS- IKI -TR-07-001, University of Skövde.

Svinicki, M.D. (1998). A theoretical foundation for discovery learning. *Advances in Physiology Education*, 20(1):S4-S7.

Tapper, P. (2003). Personality parameters: flexibly and extensibly providing a variety of AI opponents' behaviors. Gamasutra, 3/12/2003, http://www.gamasutra.com/features/20031203/tapper_01.shtml (accessed 14/8/2007).

Terdiman, D. (2006). What's wrong with serious games? At Game Developers Conference, serious-games proponents talk about what's wrong with the genre. CNET News.com. http://news.com.com/Whats+wrong+with+serious+games/2100-1043_3-6052346.html (accessed 14/8/2007).

Thalheimer, W. (2004). Research that supports simulations and simulation-like questions. A Work-Learning Research Publication.

Thiagarajan, R., and Thiagarajan, S. (1997). Interactive experiential training: eight breakthrough strategies. ISPI 1997 Session 37, http://www.thiagi.com/article-ispi97-w37.html (accessed 14/8/2007).

Thiagarajan, S. (2001). More misconceptions about simulation. Play for Performance. http://www.thiagi.com/pfp/IE4H/october2001.html#Commentary (accessed 14/8/2007).

Tong, T. (2003). Scripting in C using co-routines fully scriptable game logic. http://www.gamedev.net/reference/articles/article1974.asp (accessed 14/8/2007).

Trowbridge, D., Roxburgh, U., Hohpe, G., Manolescu, D., and Nadhan, E. (2004). *Integration patterns: patterns & practices*. Microsoft Press.

Tyndiuk, F., Lespinet-Najib, V., Thomas, G., and Schlick, C. (2004). Impact of large displays on virtual reality task performance. In *Proceedings of 3rd International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa (Afrigraph 2004)*, pages 61- 65.

Usoh, M., Catena, E., Arman, S., and Slater, M. (2000). Using presence questionnaires in reality. *Presence-Teleoperators And Virtual Environments*, 9(5):497-503.

Vilhjalmsson, H., and Samtani, P. (2005). MissionEngine: multi-system integration using Python in the Tactical Language Project. In *Proceedings of PyCon 2005*.

Vinayagamoorthy, V., Brogni, A., Gillies, M., Slater, M., and Steed, A.J. (2004). An investigation of presence response across variations in visual realism. In *Proceedings of 7th International Conference on Presence*, pages 148-155.

Vuorenmaa, M. (2000). *Automatic Presentation of Model Data in MVC++ Applications*. Master thesis. University of Tampere.

Wang, J., Lewis, M., and Gennari, J. (2003). Emerging areas: urban operations and UCAVs: a game engine based simulation of the NIST urban search and rescue arenas. In *Proceedings of 35th Winter Simulation Conference*, pages 1039-1045.

Watters, C., Oore, S., Shepherd, M., Abouzied, A., Cox, A., Kellar, M., Kharrazi, H., Liu, F., and Otley, A. 2006. Extending the use of games in health care. In *Proceedings of 39th Annual Hawaii international Conference on System Sciences*, (5):88b-88b.

Wexler, S., Adlrich, C., Johannigman, J., Oehlert, M., Quinn, C., and van Barneveld, A. (2007). Immersive Learning Simulations. February, 2007, The eLearning Guild Research, 360 Report.

Wilson, K. (2003). The GDC 2003 game object structure roundtable. http://gamearchitect.net/Articles/GameObjectRoundtable.html (accessed 14/8/2007).

Witmer, B.G., and Singer, M.J. (1998). Measuring presence in virtual environments: A presence questionnaire. *Presence: Teleoperators and Virtual Environments*, 7(3): 225-24.

Wong, W.L., Shen, C., Nocera, L., Carriazo, E., Tang, F., Bugga, S., Narayanan, H., Wang, H., and Ritterfeld, U. (2007). Serious video game effectiveness. In *Proceedings of Advances in Computer Entertainment Technology*, pages 49-55.

Wuensche, B., Kot, B., Gits, A., Amor, R., Hosking, J., and Grundy, J. (2005). A framework for game engine based visualisations. In *Proceedings of Image and Vision Computing (IVCNZ '05)*, pages 465-470.

Young, R.M., Riedl, M.O., Branly, M., Jhala, A., Martin, R.J., and Saretto, C.J. (2004). An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development*, 1(1):51-70.

Youngblut, C., and Huie, O. (2003). The relationship between presence and performance in virtual environments: results of a VERTS study. In *Proceedings of IEEE Virtual Reality Conference 2003 (VR'03)*, pages 277-278.

Yuji, H. (1996). Computer games and information-processing skills. *Perceptual and Motor Skills*, 83:643-647.

Zhou, J., and Reed, L. (2005). Chinese government documents on teacher education since the 1980s. *Journal of Education for Teaching*, 31(3): 201-213.

Zyda, M., and Sheehan, J. (Eds.) (1997) *Modeling and simulation: linking entertainment and defense*. National Academy Press.

Zyda, M. (2005). From visual simulation to virtual reality to games. *Computer*, 28(9):25-32.