Interaction Design in Everyday Computer-based Systems

—·—

Challenges, Design Strategies and Solutions

A THESIS SUBMITTED TO
THE UNIVERSITY OF KENT AT CANTERBURY
IN THE SUBJECT OF ELECTRONIC ENGINEERING
FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

By
Dimitrios-Stavros Grammenos

March 2005

ABSTRACT

This thesis aims to contribute towards addressing the challenges that interaction designers of "emerging everyday computer-based systems" (ECS) are facing. ECS can be characterised as the computer technologies and applications which are expected to become mainstream and have high impact on the lives and everyday activities of the citizens of the Information Society in the coming years. A key aspect of ECS is *diversity* in the underlying technology and (hardware and software) user interfaces, as well as in the target user groups, contexts of use and user goals.

More specifically, this thesis aims to create new knowledge and provide concrete design approaches and solutions related to four interaction domains, namely: (i) Virtual Environments (VEs); (ii) Accessible Computer Games; (iii) Software Agents; and (iv) Interactive Applications for Young Children.

In this context, the main outcomes of this thesis are:

- A novel intuitive interaction metaphor for Virtual Environments and the process followed towards defining and refining it.
- A fully-functional universally accessible chess game which is publicly available on the Web and was nominated for the final jury decision of the 2004 European Design for All Awards set by the European Commission.
- A generic model for simulating an artificial sensory system intended to be used for the creation of intelligent software agents.
- A design process and its results for the creation of a multi-perspective collaborative application for young children.

Beyond the specific contribution in each interaction domain, some more general conclusions are derived from the overall research work conducted, regarding interaction design for ECS. For each design challenge guidance is provided on how it can be faced, based on the experience gained from the research and development work conducted for the completion of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

For their encouragement and support, I am grateful to my supervisors Professor Michael C. Fairhurst and Professor Constantine Stephanidis.

For their help and advice I would like to thank my colleagues Dr. Anthony Savidis and Dr. Margherita Antona.

For their assistance I would like to thank Alexandros Mourouzis, Marialena Filou and Panagiotis Papadakos.

I would like to dedicate this thesis to

my parents Σπύρο & Καίτη,

my sister Εύη,

and my grandparents Δημήτρη & Ρηνούλα,

for all their love and support.

PUBLICATIONS RESULTING FROM THIS RESEARCH WORK & RELATED PROJECTS

The following publication stemmed from the work presented in this thesis:

Chapter III: NAVIGATION IN VIRTUAL ENVIRONMENTS

1. Grammenos, D., Mourouzis, A., and Stephanidis, C. (2005). Virtual Prints: Augmenting Virtual Environments with Interactive Personal Marks. To appear in the International Journal of Human-Computer Studies.
2. Grammenos, D., Mourouzis, A., and Stephanidis, C. (2004). Virtual Prints: A Novel Interaction Concept for Virtual Environments. In ERCIM News No. 57, April 2004. Also available on-line at: http://www.ercim.org/publication/Ercim_News/enw57/grammenos2.html
3. Mourouzis, A., Grammenos, D., Filou, M., Papadakos, P., & Stephanidis, C. (2004). Case Study: Sequential Evaluation of the Virtual Prints Concept and Pilot Implementation. In the Proceedings of the Workshop on "Virtual Reality Design and Evaluation", Nottingham, U.K., 22-23 January [The Proceedings are electronically available - CD-ROM ISBN 0 85358 123 1].
4. Mourouzis, A., Grammenos, D., Filou, M., Papadakos, P., & Stephanidis, C. (2003). Virtual Prints: an empowering tool for virtual environments. In D. Harris, V. Duffy, M. Smith, & C. Stephanidis (Eds.), Human - Centred Computing: Cognitive, Social and Ergonomic Aspects - Volume 3 of the Proceedings of the 10th International Conference on Human-Computer Interaction (HCI International 2003), Crete, Greece, 22-27 June (pp. 1426 - 1430). Mahwah, New Jersey: Lawrence Erlbaum Associates (ISBN: 0-8058-4932-7).
5. Grammenos, G., Filou, M., Papadakos, P., & Stephanidis, C. (2002). Virtual Prints: Leaving trails in Virtual Environments. In Proceedings of the Eighth Eurographics Workshop on Virtual Environments, Barcelona, Spain, 30-31 May.

Chapter IV: UNIVERSALLY ACCESSIBLE GAMES

6. Savidis, A., Grammenos, D., Stephanidis C., (2005). Developing Inclusive e-Learning Systems. To appear in a Special Issue of the international journal

Universal Access in the Information Society (UAIS) on Information Systems Accessibility (ISA).

7. Grammenos, D., Savidis, A., Stephanidis C., (2005). UA-Chess: A universally accessible board game, to appear in the Proceedings of the 3rd International Conference on Universal Access in Human-Computer Interaction (UAHCI 2005).
8. Grammenos, D., Savidis, A., Stephanidis C., (2004). An Accessible Two-player Multi-modal Board Game. In ERCIM News No. 57, April 2004. Also available on-line at: http://www.ercim.org/publication/Ercim_News/enw57/grammenos2.html

Chapter V: ENVIRONMENT SENSING FOR SOFTWARE AGENTS

9. Grammenos, D., Savidis, A., Stephanidis C., (2005). The development of a sensory system for intelligent game creatures. To appear in the Proceedings of the 11th International Conference on Human-Computer Interaction (HCII 2005).

Chapter VI: SOCIAL MUTLI-PERSPECTIVE INTERACTIVE TOOLS FOR YOUNG CHILDREN

10. Grammenos, D., & Stephanidis, C. (2002). "Interaction design of a collaborative application for children". In M.M. Bekker, P. Markopoulos, & M. Kersten-Tsikalkina (Eds.) Proceedings of the International Workshop "Interaction Design and Children", Eindhoven, The Netherlands, 28-29 August (pp. 11-28). The Netherlands: Shaker Publishing.
11. Grammenos, D., Paramythis, A., & Stephanidis, C. (2000). Designing the User Interface of an Interactive Learning Environment for Children. In C. Stephanidis (Ed.), Proceedings of the ERCIM WG UI4ALL one-day joint workshop with i3 Spring Days 2000 on "Interactive Learning Environments for Children", Athens, Greece, 3 March.

Part of the presented work was conducted in the context of the following research projects:

i) "VIEW of the Future" (IST-2000-26089) project, funded by the European Commission in the framework of the Information Society Technologies (IST) Programme.

ii) "Today's Stories" P29312 funded by the European Commission in the framework of the Intelligent Information Interfaces (i3), Experimental School Environment Programme.

UA-Chess, a universally accessible chess game developed in the context of this thesis (described in Chapter IV: UNIVERSALLY ACCESSIBLE GAMES), was nominated for the final jury decision of the European Design for All Awards[1] set by the European Commission, in the category "AT/Culture, Leisure and Sport". UA-Chess is also listed at the OneSwitch.org.uk web site, which provides resources for moderate to severely learning/physically disabled people, and described as a "*highly accessible Chess game*". The PCS Games web site[2] has a link to UA-Chess in a web page entitled "Fantastic Games and where to find them, accessible for the blind[3]". The Newsletter of Accessibility Interest Group by PurpleTop of March 2005[4] in the section entitled "Sites of Interest" has a link to UA-Chess and the following short description: "UA Chess is an accessible chess game with a variety of features including sole mouse, or keyboard (or any type of switches emulating keystrokes) or speech recognition input, including an inbuilt screen reader". Research related to UA-Chess resulted in the development of a tutorial entitled "Design and Implementation of Universally Accessible Computer Games", which will be delivered on the 24th of July, 2005 in the context of the 11th International Conference on Human-Computer Interaction (HCII 2005) in Las Vegas, , 22-27 July 2005.

Finally, the work conducted in Chapter V: ENVIRONMENT SENSING FOR SOFTWARE AGENTS was presented in two undergraduate seminars on "Artificial Intelligence, Software Agents and Sensing" at the Department of Computer Science of the University of Crete during the Fall semester 2004.

[1] http://www.dfa-at-awards.org
[2] http://www.pcsgames.net/
[3] http://www.pcsgames.net/game-co.htm
[4] http://www.purpletop.com.au/newsletter/current.php

EXECUTIVE SUMMARY

The advent of the Information Society is bringing about a paradigm shift, whereby computers are no longer conceived as mere business tools, but as integrated environments of use, available to anyone, anytime, anywhere. Thus, interaction parameters (i.e., the users, tasks and context of use) are turning from static, limited and precise, to highly dynamic, manifold and vague. A single person may interact with several different devices at the same time, a number of people may be sharing the same device, or many people may cooperate through different devices. New interaction technologies and platforms, for which limited - if any - interaction design knowledge is currently available, rapidly become mainstream and require the development of adequate user interfaces. Although the prevalent user interface paradigm is still that of windows-based graphical user interfaces, they now come in several different variations, depending on the type and characteristics of the device or platform (e.g., desktop, palmtop, mobile phone) through which they are instantiated. As a result, the challenges faced when designing for human-computer interaction are dramatically changing.

The notion of "emerging everyday computer-based systems" (ECS) is an all-encompassing term used for describing computer technologies and applications which are expected to become mainstream and have high impact on the lives and everyday activities of the citizens of the Information Society in the coming years. The key characteristic of ECS is *diversity* in the underlying technology and (hardware and software) user interfaces, as well as in the target user groups, contexts of use and user goals. In this context, a number of challenges faced by interaction designers are identified: (i) creating novel interaction metaphors and techniques; (ii) designing for non-typical interaction paradigms; (iii) designing for non-typical user groups; (iv) designing for accessibility; and (v) designing based on incomplete knowledge or, designing for openness and extensibility.

The main goal of this thesis is to contribute to the field of interaction design by creating new knowledge and providing concrete solutions to these challenges, as well as deriving more general conclusions concerning the characteristics and properties

that interaction design solutions for ECS should exhibit in order to address such challenges. As ECS constitute a vast research area, comprising an ever-increasing number of interaction domains, this thesis focuses on four representative interaction domains which were deemed of major importance: (i) Virtual Environments (VEs); (ii) Accessible Computer Games; (iii) Software Agents; and (iv) Interactive Applications for Young Children. For each of these domains a key research issue reflecting one or more of the aforementioned challenges is selected and addressed. In summary, the related challenges and research issues are:

Interaction Domain	**Design Challenges**	**Key Research Issue**
Virtual Environments	Creating novel, intuitive, interaction metaphors for a non-typical interaction domain.	Developing a new interaction concept for supporting navigation in Virtual Environments.
Accessible Computer Games	Designing for accessibility and Universal Access; designing based on incomplete knowledge.	Creating a universally accessible game, proactively designed to optimally fit and adapt to different individual gamer characteristics and (dis)abilities without the need of further adjustments or developments.
Software Agents	Extending the capabilities of a novel interaction paradigm providing higher design flexibility and more options to designers; designing for openness and extensibility.	Developing a generic approach to creating a synthetic sensory system, applicable to any type of embodied agent in any type of virtual world, independent of the nature or the intrinsic characteristics of the senses or simulated stimuli.
Interactive Applications for Young Children	Designing user interfaces for a non-typical application (including the creation of new metaphors) for non-typical user groups.	Designing an interaction environment, suitable for young children, empowering them to reflect on recorded events of their daily lives, presented through alternative perspectives and supporting collaborative viewing and annotation.

Towards addressing the above design challenges and key research issues in the selected domains, the main contributions of the thesis are:

- A novel intuitive interaction metaphor (Virtual Prints) for supporting navigation, orientation, way-finding, as well as a number of additional functions in Virtual Environments, along with the process followed towards defining and refining it.
- A fully-functional universally accessible Web-based board game, designed on the basis of Design for All principles, and a detailed account of how accessibility is supported for different user categories through the game's interface, its adaptation capabilities and the available alternative input and output modalities. The game is publicly available on the Web and was nominated for the final jury decision of the European Design for All Awards set by the European Commission, in the category "AT/Culture, Leisure and Sport".
- A generic model for simulating an artificial sensory system intended to be used for the creation of intelligent software agents which exhibits openness, scalability, reusability, extensibility, simplicity and computational efficiency.
- A design process for the creation of a multi-perspective collaborative application for young children, as well as the resulting design concepts, metaphors and solutions, along with their design rationale.

Furthermore, some more general conclusions stemming from the overall research work conducted are presented, regarding interaction design for ECS in order to address the related challenges.

The outcomes of the presented work have appeared in several publications and also led to two undergraduate seminars on "Artificial Intelligence, Software Agents and Sensing" at the Department of Computer Science of the University of Crete during the Fall semester 2004 and to a tutorial entitled "Design and Implementation of Universally Accessible Computer Games", to be delivered in the context of the 11th International Conference on Human-Computer Interaction (HCII 2005) in Las Vegas, 22-27 July 2005.

Chapter I

INTRODUCTION

1.1. INTERACTING WITH COMPUTERS IN THE INFORMATION SOCIETY

The technological progress of the last few decades in the domain of information technology and telecommunications has led to the penetration of computers in almost all walks of life (work, entertainment, education, etc.) and has generated a new industrial revolution, that, in contrast to those of the past, is not based on material, but on information. This information revolution is changing dramatically the way people work, communicate and live. The result of these social and organisational changes has lead to the creation of what is today called the *Information Society*.

In this emerging Information Society, people become more and more dependent on computer technology and, in contrast to previous generations of computing, computer users are not only the experts, but practically anybody, thus addressing a user population with diverse abilities, skills, requirements and preferences. In order to maximise the benefits and minimise the risks of the Information Society, "*a great deal of effort must be put into securing widespread public acceptance and actual use of the new technology*" (Bangemann, 1994). To achieve this goal, "*people need to be convinced that the new technologies will give them more control over their lives, not less, that they are the masters of the technology, not its slaves*" (ISPO, 1995).

Undoubtedly, security, privacy and data integrity are some of the basic pre-requisites for gaining the public acceptance. However, in order to make technology truly acceptable, interaction has to become more usable and friendly.

In this context, the discipline of Human-Computer Interaction (HCI) is "concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them" (Hewett, 1992). HCI is not a "stand-alone" discipline (in the sense of physics or mathematics) but rather an amalgam resulting from the combination of computer science with human-related disciplines such as psychology, sociology, anthropology, linguistics, design, ergonomics and human factors. A basic concept of HCI is the "user interface",

which can be described as a communication means through which interaction can be accomplished between the human/user and an interactive application computer. A very general definition of the interface of a system, according to (Moran, 1980) is: "... *the part of the system with which the user comes into contact physically, perceptually and cognitively.*" Referring to computer systems, the term 'user interface' (see Figure 1) encompasses the hardware and software which enable end-users to access and interact with computer applications or services, facilitate use of the functions provided by the application / service, and empower accomplishment of the intended tasks.

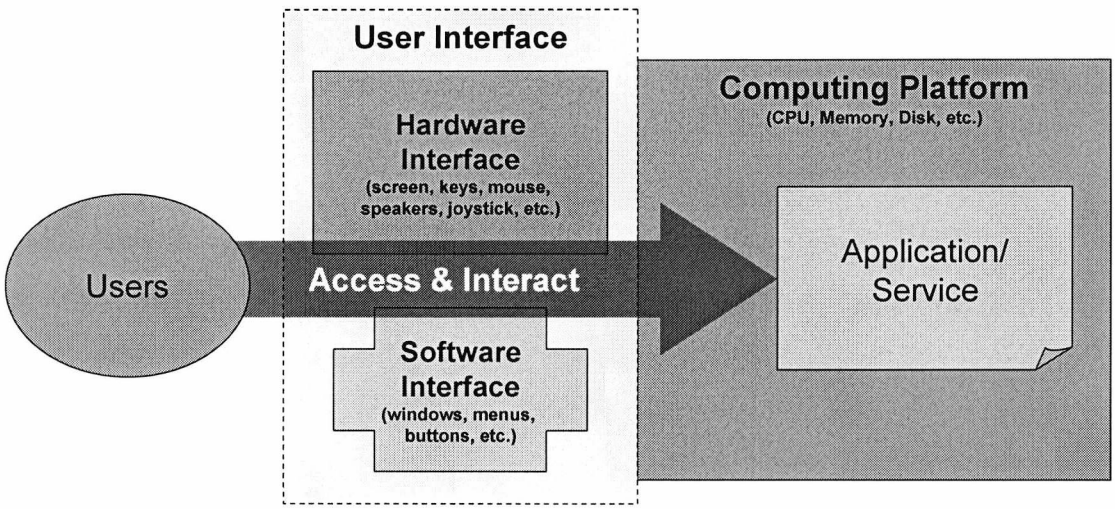


Figure 1: Abstract depiction of the user interface

Interaction design is the domain of HCI concerned with the design of user interfaces, both in terms of the process followed (methods, steps, participants, etc.) and of the characteristics of the resulting artifacts (metaphors, interaction objects, layout, etc.). Several approaches have been suggested for the process that should be followed, most of which are to some extent based on the principles of user-centred design (ISO, 1999). Regarding the properties of a "good" user interface, related literature comes in abundance, ranging from international standards (e.g., ISO, 1998; ISO, 2002), to commercial companies' guidelines (e.g., Apple, 2004; Microsoft, 2004) and books providing related guidance and examples (e.g., Galitz, 1997; Howlett, 1996; Shneiderman, 1998; Tognazzini, 1996; Wood, 1998).

1.2. OBJECTIVES AND RATIONALE

Until recently, a typical user interface design case would be described as: single user, using one computer with a windows-based graphical user interface (GUI), to accomplish well-defined (mostly work-related) tasks, in a specific, static, environment. The most "extreme" design scenarios might include someone using a bank ATM against bright sunlight wearing gloves, or a ticket vending machine in a noisy underground station. Thus, interaction design was quite straightforward and mainly limited to mapping system functions to adequate interaction objects, successfully grouping menu items, and properly adjusting the visual layout in order to maximize user effectiveness and efficiency and minimize errors and mental workload.

The advent of the Information Society is bringing about a paradigm shift, whereby computers are no longer conceived as mere business tools, but as integrated environments of use, available to anyone, anytime, anywhere. In contrast to the "standard" desktop PC of the past, comprising a screen, a bulky rectangular CPU unit, a keyboard, and a mouse, computers now come in all shapes and sizes, and even reside in most everyday electronic appliances. Computer application domains and services range from work and information access to leisure and independent living and are targeted to people with different cultural, educational, training and employment background, novice and experienced computer users, the very young and the elderly, as well as people with different types of disabilities. A single person may interact with several different devices at the same time, a number of people may be sharing the same device, or many people may cooperate through different devices.

Thus, interaction parameters (i.e., the users, tasks and context of use) are turning from static, limited and precise, to highly dynamic, manifold and vague. For example, when designing an Internet-based chess game, it is rather difficult (if not impossible) to exhaustively account (or predict) and analyze the characteristics of all the potential players or contexts of use. New interaction technologies and platforms, to which limited if any previous interaction design knowledge applies, rapidly become mainstream and require the development of adequate interfaces. Although the prevalent user interface paradigm is still that of windows-based graphical user interfaces, they now come in several different variations, depending on the type and

characteristics of the device or platform (e.g., desktop, palmtop, mobile phone) through which they are instantiated. As a result, the challenges faced when designing for human-computer interaction are dramatically changing.

In this context, the notion of "emerging everyday computer-based systems" (ECS) is an all-encompassing term used for describing computer technologies and applications which are expected to become mainstream and have high impact on the lives and everyday activities of the citizens of the Information Society in the coming years. Some indicative examples include Virtual & Augmented Environments, ubiquitous computing and ambient intelligence, intelligent agents, wearable and mobile computers, and universally accessible systems.

The key characteristic of ECS is *diversity* (Figure 2), which is present in the underlying *technology* and (hardware and software) *user interfaces*, as well as in target user groups (who), context of use (where) and user goals (what for).

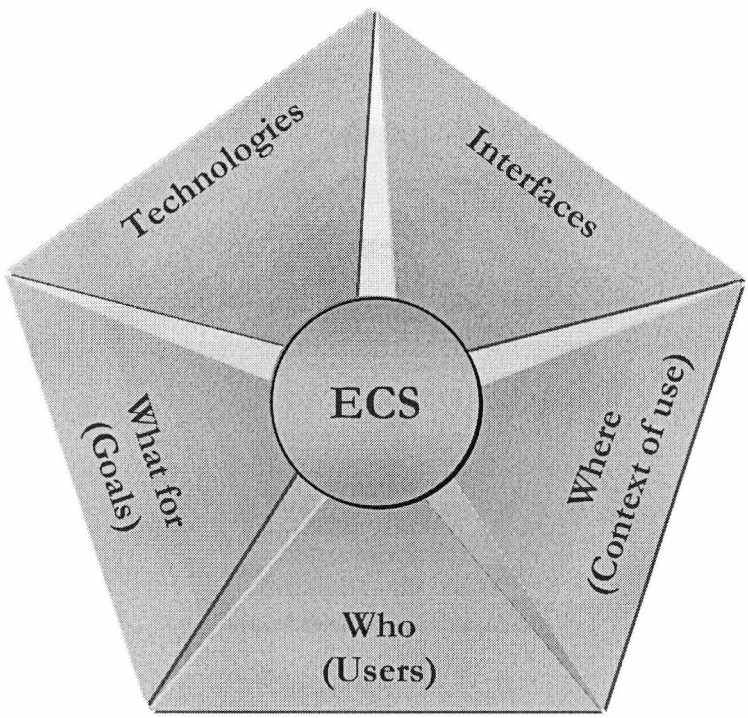


Figure 2: Dimensions of diversity of emerging everyday computer-based systems

In the light of the above, interaction designers of ECS have to face an increasing number of interweaved challenges which can be summarized as follows:

[C1] Creating novel interaction metaphors and techniques that better suit the characteristics, needs and requirements of any given combination of technological platforms, interfaces, user groups, contexts of use and applications.

[C2] Designing for non-typical interaction paradigms for which little or no established design wisdom, guidance, examples and guidelines exist, e.g., Virtual and Augmented Environments, ubiquitous and pervasive computing, Artificial Intelligence and software agents, etc.

[C3] Designing for non-typical user groups, which may have nothing in common with the designer, e.g., young children, people with no computer experience, the elderly.

[C4] Designing for accessibility, which is associated with the previous challenge, but also a requirement that is gradually becoming a legal obligation.

[C5] Designing based on incomplete knowledge, since it is not longer possible to be aware or keep track of all the parameters of the design space. This challenge can also be formulated as designing for openness and extensibility, so that additional requirements can be accommodated later on, for example through the extension of the supported interaction methods and devices.

These challenges are not isolated, since it is quite common that a combination of several of them may appear in a design case. For instance, in the example of the Internet-based chess game previously mentioned, challenges C3, C4 and C5 need to be concurrently addressed. C2 may also be relevant if the game is to be played using new devices, such as palmtops or mobile phones. Furthermore, for each of the challenges, all the dimensions of diversity illustrated in Figure 2 may apply.

In this context, the main goal of this thesis is to contribute to the field of interaction design by providing concrete knowledge and solutions to the aforementioned challenges faced by interaction designers, as well as deriving more general conclusions concerning the characteristics and properties that interaction design solutions for FEEECS should exhibit in order to address such challenges.

As ECS constitute a vast research area, comprising an ever-increasing number of interaction domains, it is not possible for a single research effort to cover all the possible research issues in all domains. Thus, in the context of this thesis, four representative interaction domains are addressed which were deemed of major importance. For each of them a key research issue is selected, and a respective solution is proposed.

The selected interaction domains and the reasons for which they were selected are briefly introduced below.

i) Virtual Environments (VEs)

Virtual Environments (Loeffler & Anderson, 1994) is a term used to describe Virtual Reality (VR) applications that constitute 3D representations of real or imaginary worlds, in which users are fully or partially immersed, and with which they interact using traditional (e.g., mouse, joystick, keyboard) or 3D (e.g., trackers, datagloves, 3D wands) devices and metaphors. As VR hardware technologies mature and become more stable and affordable, the respective market is growing rapidly (Arrington & Staples, 2000; CyberEdge Information Services, 2001), and VE applications start to migrate from the research laboratories to real use (Delaney, 1999). But as relevant research (e.g., Crosier at al., 2000; Bowman, Gabbard & Hix, 2001) shows, the way people interact with them is still quite problematic and, with a few exceptions, far from intuitive. Thus, there is a need for the creation of novel concepts and metaphors that support intuitive interaction in such environments.

ii) Accessible Computer Games

In the past few years, the accessibility of electronic applications and services by elderly and disabled users has become a topic of paramount importance at an international level. Based on the fundamental right of all people to equal access to

information and services, and equal opportunities for employment and independent living, several governments and political bodies (e.g., Access Board, 2000; European Commission, 2002) have adopted legislative and policy measures towards ensuring (or enforcing) software and Web accessibility. Beyond working and independent living, another basic need of most people is that of leisure and entertainment, and computer games constitute indisputably one of the major types of application in this filed. The related industry world-wide is vast and comparable to that of books and music retail, the cinema box office, and VHS/DVD rental (Screendigest, 2004). However, so far little attention has been paid to the development of computer games that can be played and shared by all players, independently of their personal characteristics, requirements or (dis) abilities. Thus, there is a real need for developing games that on the one hand prove that the creation of accessible entertainment applications is a feasible and manageable task, and, on the other hand, show how this can be achieved (Grammenos, Savidis & Stephanidis, 2005b).

iii) Software Agents

Currently, the interaction paradigm followed by the user interface of most computer applications and services is that of *explicit instruction*; i.e., the interface provides a number of functions (through a set of interaction objects) that users have to trigger using direct manipulation in order to complete their tasks. This paradigm, although offering maximum user control, is not suitable for users that do not know how to translate their goals to computer commands (i.e., non-experts), nor in situations where the information the users should act upon is of great size, or not readily available. To contribute overcoming such problems, a novel paradigm of human-computer interaction has been proposed in the form of *software (or intelligent) agents* (Franklin & Graesser, 1997; Wooldridge & Jennings, 1995). Agents are software entities that are based on a set of given goals and can act on behalf of the user, performing part (or all) of the user's tasks. A class of agents that can be of particular interest for future computing systems is that of *embodied* agents, i.e., agents that have a visual representation (*a virtual body*). Such agents can have a variety of uses ranging from friendly assistant characters for applications and services, to lifelike game creatures and believable virtual actors. A basic requirement for creating any type of agent is to provide it with the ability to sense the virtual environment it is situated in. In the case

of embodied agents, this task can be particularly challenging and currently no related generic solutions are available (Grammenos, Savidis & Stephanidis, 2005a).

iv) Interactive Applications for Young Children

In the emerging Information Society, computing systems and applications are being developed for user groups that have little or no previous computer experience, such as the very young, the elderly and the so called "computer illiterate". All these groups have their own particular requirements and preferences, and the software they are most likely to use shares little - if no - common ground with currently available business- and office-oriented software applications. Thus, there is an emerging need of designing "non-traditional" user interfaces for "non-traditional" users. In this context, designing for young children constitutes a representative and very challenging case, for several reasons:

(a) Today's children, since the day they are born, are immersed and grow up in a highly technological environment. Thus, their relation with and understanding of technology is totally different from that of adults, who have to reactively adapt their established way of thinking to "intruding" technologies radically affecting the way they live and work.

(b) The main goal of children when interacting with technology (even when "learning" or "working") is to have fun. This design target is much more abstract and difficult to achieve than the traditional HCI goals of usability, effectiveness and efficiency, and little related documented background knowledge is available.

(c) Using the desktop metaphor is not appropriate, since children start to use computers far before they come into contact with an office environment. Consequently, new adequate metaphors and world analogies must be conceived.

(d) Children exhibit an interesting form of collaboration, where multiple users share a common computer and work / play together as partners, a case that is quite typical both in the school and the home environments.

1.3. STRUCTURE AND CONTENT OF THE THESIS

The rest of this thesis is structured as follows:

Chapter II: SELECTED TARGET INTERACTION DOMAINS

This chapter introduces the four major interaction domains addressed by this thesis. For each domain, a key research issue that was selected to be addressed is highlighted and related work is presented.

Chapter III: NAVIGATION IN VIRTUAL ENVIRONMENTS

This chapter is concerned with the challenge of creating of a novel, intuitive, interaction metaphor [C1] for a non-typical interaction domain (Virtual Environments) [C2], targeted to supporting navigation, orientation, way-finding, as well as a number of additional functions. A substantial part of the chapter is dedicated to describing the process followed towards defining and refining the metaphor (named Virtual Prints) and instantiating it in the form of an interactive prototype. A detailed account of how users can interact with Virtual Prints is provided and related issues and challenges are discussed along with techniques and methods for overcoming them. The chapter also includes detailed results from expert and user-based evaluation sessions.

Chapter IV: UNIVERSALLY ACCESSIBLE GAMES

This chapter is related to the challenge of designing for accessibility [C4] (and more specifically for Universal Access), as well as of designing based on incomplete knowledge [C5], in an important and substantial interaction domain (computer games) for which very few related efforts exist (and none addresses Universal Access). More specifically, the chapter presents the development of a universally accessible board game, named UA-Chess, based on the principles of Design for All (or Universal Design). This game can be concurrently played by people with different abilities and preferences, including people with disabilities (e.g., low-vision, blind and hand-motor impaired) remotely over the Internet, or locally on the same computer. The interaction capabilities of UA-Chess allow it to provide access to people with combinations of

disabilities, such as, for example, blind hand-motor impaired persons. The chapter discusses in detail how accessibility is supported in UA-Chess for different user categories through the game's interface, its adaptation capabilities and the available alternative input and output modalities. The chapter also provides the results of a preliminary evaluation with representative users.

Chapter V: ENVIRONMENT SENSING FOR SOFTWARE AGENTS

This chapter is about extending the capabilities of a novel interaction paradigm (embodied software agents) [C1] providing higher design flexibility and more options to designers and also designing for openness and extensibility [C5]. More specifically, the chapter describes the design and implementation of a generic model for recreating an artificial sensory system intended to be used for the creation of intelligent software agents. The main design goals were openness, scalability, reusability, extensibility, simplicity and computational efficiency. The chapter introduces the basic concepts and algorithms used by the suggested system, as well as a number of attributes and characteristics identified and adopted, on the one hand for supporting several alternative design options, and on the other hand for improving system performance. Subsequently, the chapter describes how the suggested sensory system can be integrated into an intelligent creature, and indicative uses and design cases are discussed.

Chapter VI: SOCIAL MUTLI-PERSPECTIVE INTERACTIVE TOOLS FOR YOUNG CHILDREN

This chapter is about designing the user interface of a non-typical application (multi-perspective collaborative application) [C2] (including the creation of new metaphors [C1]) for a non-typical user group (young children) [C3] based on several constraints and requirements stemming from the end-users (children), the domain experts (educators) and the technology used (hardware and software limitations). The chapter illustrates the design process followed, as well as the resulting design concepts and solutions that were created against these constraints, along with their design rationale and the related issues and problems that had to be tackled. The results of the evaluation with children are also presented, along with the consequent modifications of the user interface.

Chapter VII: SUMMARY, CONCLUSIONS AND FUTURE WORK

This chapter concludes the thesis providing a summary of the conducted research and its main contributions, and suggesting directions for related future work.

APPENDIX A

This appendix contains the questionnaire used for evaluating the usability of the Virtual Prints mechanism interactive prototype (presented in Section 3.6.3).

APPENDIX B

This appendix contains the questionnaires used for the evaluation of UA-Chess (presented in Section 4.4).

Chapter II

SELECTED TARGET INTERACTION DOMAINS

2.1. INTRODUCTION

This chapter presents the four interaction domains on which this thesis focuses. For each domain, a brief introduction is first provided. Then, an associated key research issue that the thesis addresses is stated, followed by an overview of related work in the particular area. The interaction domains addressed are:

i) Virtual Environments

ii) Accessible Computer Games

iii) Software Agents

iv) Interactive Applications for Young Children

2.2. VIRTUAL ENVIRONMENTS

The term Virtual Reality (VR) was introduced in 1989 by Jaron Lanier, founder of VPL Research (a company involved is NASA projects), as part of the marketing campaign for VPL's products (a head-mounted display and a wired glove). VR was defined as "*a computer generated, interactive, three-dimensional environment in which a person is immersed*". In the last few years, the term *Virtual Environments* (VEs) (Loeffler & Anderson, 1994), has prevailed over *Virtual Reality* (VR). A *Virtual Environment* can be described as a 3D representation of a real or imaginary environment, generated using computer technology, which a human can perceive and interact with through adequate input and output devices. Usually, both terms are used interchangeably, but in some cases *Virtual Environment* is used to describe just the content, while *Virtual Reality* refers to both the technology and the content. There are also several other terms related to the same concept, such as Artificial Reality, Synthetic Experience, interactive 3D (i3D), Synthetic Environments, Virtual Worlds and Artificial Worlds. The most prominent characteristic of VEs is their ability to provide a superior sense of involvement, creating a feeling "presence", i.e., of actually "being there".

Virtual Environments are instantiated using a VE (or VR) system. A VE system typically comprises:

(a) A computer with one or more graphics cards (usually with 3D acceleration) that recreates the virtual world in real-time.

(b) Software that is responsible for modelling the virtual environment, rendering different views of it according to the user's position, and orchestrating the interaction.

(c) A visual display for projecting the image of the VE, which may range from a standard computer monitor, to a head-mounted-display or multiple projection surfaces.

(d) Input devices for supporting user navigation in the VE and interaction with the objects it contains.

In addition to the above, sometimes alternative output modalities maybe supported, such as, for example, aural, haptic or even olfactory, in order to provide more realistic and immersive experiences. In the past few years, a number of industrial VE applications have been developed and put to practical use. The VR market is growing rapidly (Arrington & Staples, 2000; CYBEREDGE, 2001) as VEs have been adopted as a useful and productive tool for a variety of applications (Delaney, 1999).

2.2.1. Categories of VE systems

Depending on the technologies employed and the interaction capabilities they provide, VE systems can be classified into the following categories:

2.2.1.1. Desktop systems

The virtual environment is displayed using a desktop computer monitor through which the user sees only a small part of the virtual world. This is why this type is also called "Fishtank" VR, or Window on World (WoW). Typically, viewpoint control, navigation and interaction are achieved through standard PC input devices (e.g., keyboard, mouse, joystick), while more rarely 3D devices are used, as for example 3D mice or multi-axis joysticks. Although desktop systems are capable of rendering stereoscopic images (e.g., using polarized or shutter glasses, or stereo monitors) they often are not considered as VR because they cannot present objects in their real size or block out the real world, thus not creating a feeling of immersion. Desktop systems

have very low cost, and can be easily shared by several participants, but navigation and interaction through such a limited view using 2D devices can be really painstaking and problematic.

2.2.1.2. Immersive systems

In an immersive system, the user has no visual contact with the physical world. The VE is seen through a head-mounted display (HMD) or projected on multiple large surfaces, while interaction and navigation are mainly achieved through 3D input devices, such as 3D mice, wands, position trackers and data gloves.

Head-mounted display (HMD) systems

A head-mounted display (HMD) is a helmet equipped with two small screens that are placed in front of the user's eyes. The combination of the images projected on each screen produces an illusion of stereoscopic viewing. Usually, HMDs are coupled with stereo headphones, as well as with a head tracker that captures the user's position and head movement, so that the corresponding view of the environment can be projected on the screens. HMD systems were the first type of systems used for creating VEs and are usually single-user systems, since engaging more users would require replicating the whole (costly) hardware configuration. The main advantages of HDM systems are the high-level of immersion they can provide, since they totally isolate the user from the real world, and the tight coupling between the head movement and change of viewpoint. On the other hand, their major drawback concerns ergonomics and hygiene. Wearing a bulky device, tightly fastened on the head, can become very discomforting after a just quarter of an hour, while having several users sharing the same equipment may result in sanitary problems. Some additional problems, which are gradually being solved as technology progresses, are the limited available field of view and display resolution.

Projection systems

In these systems, the virtual world is projected on one or more surfaces using video projectors. In order to display stereo images, two techniques are used: (a) stereo projection and (b) passive projection. In stereo projection, for each surface a single projector is used that displays sequentially the image for the left and the right eye,

several times per second. To achieve 3D viewing, participants wear special (i.e., "shutter") glasses which, in synchronization with the projector, alternatively block the view of the "other" eye. In passive projection, the image for both eyes is concurrently displayed using two separate projectors. Each projector uses a different polarization filter, while participants wear correspondingly polarized glasses, which are lightweight and very cheap. Typical projection systems are single or multiple wall systems and workbenches, where the image is projected on a horizontal or inclined table-size surface. The main advantage of projection systems is that the virtual environment can be shared among several participants (at the minor cost of the special glasses) and with much greater comfort than HMD systems. Furthermore, objects can be represented at their real-life size. Their basic drawback is that the image is adjusted according to the viewpoint of one participant (who is being tracked), and the accuracy of the display for everybody else depends on the specific position in relation to that participant. Furthermore, usually the tracked participant is the only one who can interact with the system.

2.2.1.3. Augmented reality systems

Augmented Reality systems mix a scene of the real world with computer-generated 3D objects that appear to be integrated in it. This is achieved through the use of see-through HMDs and glasses. Ideally, the user should not be able to tell apart the real from the virtual objects. In certain cases, Augmented Reality systems maybe used to reduce the complexity of the perceived environment (e.g., outline points of interest, remove objects). This is why the terms Mediated or Diminished Reality are also used. Augmented reality systems have numerous potential uses, since they can augment human perception by timely providing context-sensitive information about any task and situation, but they are still at a preliminary experimental stage.

2.2.1.4. Telepresence systems

In telepresence systems, any of the aforementioned VE system types are used to control remote devices (e.g., robots, mechanical arms, vehicles) situated in environments that are hostile, inconvenient or inaccessible for human beings (e.g., the surface of the moon, a nuclear reactor, a water pipe). Environmental feedback is

collected through different types of sensors, such as cameras, distance detectors, X-ray scans, etc., and provided to the user through the virtual environment. Furthermore, telepresence may be used for supporting collaboration amongst geographically dispersed groups of people.

2.2.2. Interacting with VEs

Since their conception, Virtual Environments carried the promise of more intuitive and powerful interaction, based on the mechanisms and methods humans use to perceive and interact with the real world. The main idea was that there would no longer be the need for an explicit interface, since the user would be able to directly interact with the contents of the virtual environment. However, practice has shown that this is difficult to achieve.

Although today, using state-of-the-art computer graphics hardware and software, it is possible to build highly realistic and detailed virtual environments in which the user can freely move in real-time, the way people interact with them is still quite problematic and, with few exceptions, far from intuitive. This can be attributed to several reasons:

(a) *The precise manipulation of virtual objects is hard* (Mine et al., 1997). With a few (mainly research-based) exceptions, there is no *haptic feedback* or physical constraints to restrict movement and notify the user about collision with objects. Then, there is *limited input information*, as the user is equipped with a single means and / or modality for providing information to the system. Furthermore, there is *limited precision*, as the devices used to track position and orientation do not always provide accurate data, and are subject to different types of interferences.

(b) *Virtual environments lack a unifying framework for interaction* (Mine et al., 1997). Unlike 2D windows-based user interfaces, where the desktop metaphor is a *de facto* standard, there are no commonly established interaction techniques and metaphors, and consequently no relevant guidelines exist for the selection, development and use of such interaction techniques. This means that the same device or interaction technique may be treated in a radically different way by

different applications, leading to user confusion, frequent errors and substantially increased learning time and effort.

(c) *Poor hardware ergonomics*. Interacting in 3D space involves intensive physical movement, as well as standing up for long periods of time. In addition to that, the user is required to wear heavy gear on the head or different parts of the body and may also be attached with several wires. As a result, the whole experience can quickly become very tiring.

(d) *Lack of correspondence with real-life activities*. VEs, being digital, immaterial, worlds can allow for interactions and tasks which are not possible or even imaginable in reality. Thus, new techniques, for which there are no physical counterparts and consequently for which there is no prior user experience to take advantage of, must be first invented and then learned "from scratch".

To study in detail and eventually tackle the aforementioned problems, there have been several efforts towards analyzing, modelling and evaluating 3D interaction tasks and techniques (e.g., Poupyrev, et al., 1997, Bowman & Hodges, 1999, Masrh et al., 2001).

Interaction in VEs can be classified in two basic categories: *travel*, and *object selection / manipulation.*

Travel (or *Viewpoint Manipulation*, Hand, 1997) is the minimum interaction capability offered by any VE, and involves the ability of the user to control the position (i.e., to move) and orientation (i.e., where he is looking at) of the corresponding virtual body. This can be done either directly, e.g., by tracking the user's head, or indirectly, through a 2D or 3D input device, e.g., a mouse or a joystick. Several alternative related techniques have been proposed, while the most widely used (mainly due to its implementation and input requirements simplicity) is that of *flying*, where the user is able to move unconstrained anywhere in the VE, typically using a head-tracker or a mouse to select movement direction and one or more buttons to adjust movement velocity. Travel techniques involving physical locomotion (e.g., walking, cycling) are also employed but they have to deal one hand with limitations in the available physical space and the fact the user is probably wired to the system, and

on the other hand with user fatigue. A taxonomy of travel techniques, suggested by Bowman & Hodges (1999), is illustrated in Figure 3. *Navigation* is a combination of *travel* and *wayfinding*, which roughly means knowing where you are and how you can move to a desired destination. Navigation will be discussed in more detail in section 2.2.4 "Related Work".

In order for the user to be able to act in the VE and affect the state of its contents, *object selection and manipulation* needs to be supported. Object selection is required for specifying the item that the user wishes to interact with, while manipulation permits him to alter the selected object's characteristics (e.g., position, size, activation / deactivation). The same techniques are also used for system control, i.e., setting application parameters. Some of the required actions may have direct analogies in the real world (e.g., grabbing and moving an object, pressing a button) and thus designing appropriate interaction techniques can be quite straightforward, but others may not (e.g., deleting, resizing, copying and pasting), consequently requiring much experimentation and assessment. An indicative taxonomy of object selection and manipulation techniques, suggested by Bowman & Hodges (1999), is illustrated in Figure 4.

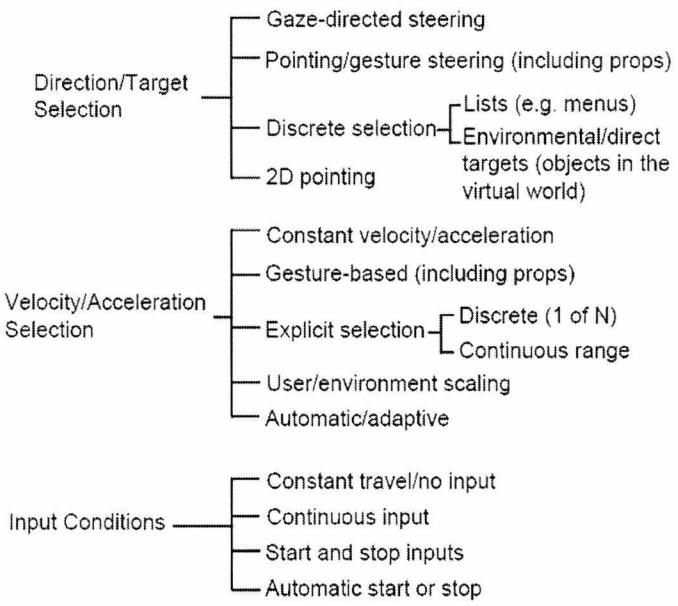


Figure 3: A taxonomy of travel techniques (Bowman & Hodges, 1999)

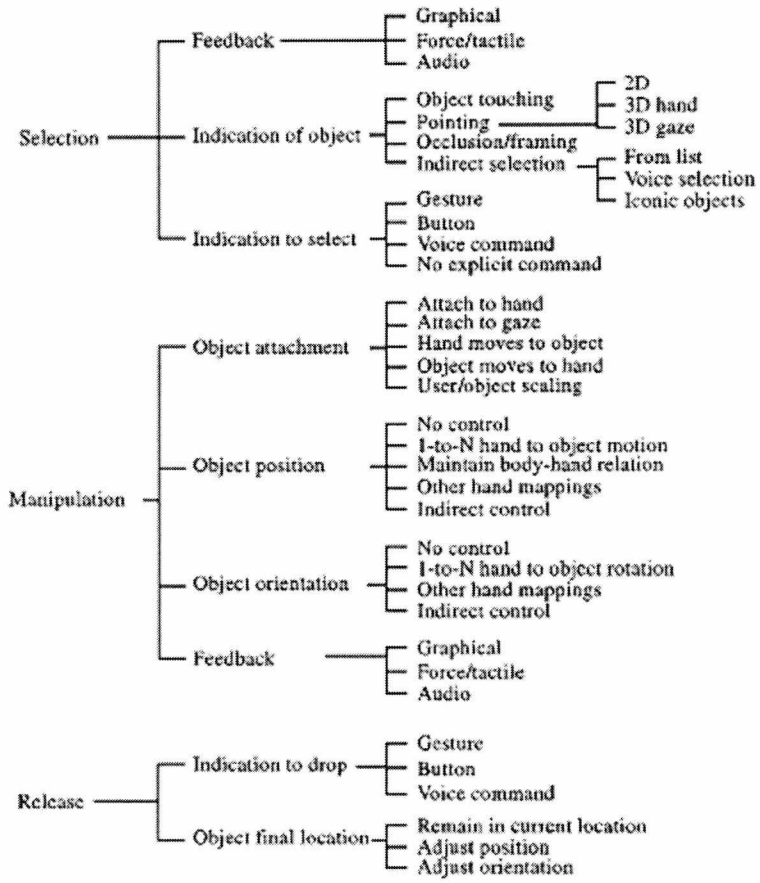


Figure 4: A taxonomy of object selection and manipulation techniques (Bowman & Hodges, 1999)

2.2.3. Key Research Issue: Navigation in Virtual Environments

User feedback reveals that there are still several barriers that impede the sustainable and appropriate use of VEs in industry, including barriers concerning the integration of technologies, barriers due to insufficient knowledge concerning the impact of such technologies on the user, as well as usability barriers (Crosier at al., 2000; Bowman, Gabbard & Hix, 2001). Navigation is a key task in any type of VE. Navigation can be considered as a combination of *travel* and *wayfinding*. Travel is the minimum interaction capability offered by any VE and involves the ability of the users to control the position (i.e., to move) and orientation (i.e., gaze direction) of their virtual body. Wayfinding means that the user is aware of his/her current location and of how to get to a desired destination. Although there have been numerous efforts in this area,

navigation still remains a major challenge, since observations from numerous studies and usability analyses indicate that this task (especially in large-scale VEs) can be very difficult, and may result in user disorientation and distress (Darken & Goerger, 1999; Darken & Sibert, 1993; Ellis et al., 1991; McGovern, 1993; Vinson, 1999).

The reasons for which navigation can be so cumbersome in VEs can be summarized in the following:

(a) *Navigation is also a difficult task in the real world.* Humans may have difficulties when dealing with unfamiliar or complicated and unstructured physical environments (e.g., a forest, a highway, or a modern building). To overcome these difficulties navigation support tools have been developed including maps, compasses, signs, and electronic global positioning systems (GPS). Thus, even if virtual environments were indistinguishable from the real ones, navigation would still be a major challenge.

(b) *Lack of constraints* (Chen, 2003). In the real world, several constraints exist when moving from one location to another. There are paths to follow, doors to go through, insurmountable obstacles, and distance or time restrictions that significantly decrease movement possibilities. In most VEs, the user has the capability to fly or to be instantly be transported to remote places, and often minimal actions and physical effort result in dislocation over very large distances.

(c) *Lack of cues* (Vinson, 1999). Humans, in order to navigate in large spaces of the physical world, subconsciously reconstruct an abstract mental representation the environment, known as a "cognitive map". This representation is created through a combination of spatial and proprioceptive cues. Spatial cues may include landmarks, the relative position of structures and buildings, but also sounds and smells. Proprioceptive cues are collected through the physical locomotion of the body and its parts. VEs provide significantly fewer spatial cues than real environments. First, there are technical and cost limitations for producing high-fidelity visual representations. Additionally, there is high re-use of 3D models in order to ease the task and cost of populating and displaying virtual worlds, and thus sometimes it is difficult to differentiate between different parts of the environment. Non-visual cues are rarely integrated in VEs, and proprioception is

engaged only through devices and techniques that are still in the form of research prototypes. Finally, in non-immersive VR applications, where the VE is viewed at a small scale and through a limited, external, viewpoint, cues are hard to notice and internalise.

Navigation in virtual environments is concerned with the challenges of creating of a novel, intuitive, interaction metaphor [C1] for a non-typical interaction domain [C2] (see Chapter I).

2.2.4. Related Work

Related work can be broadly classified in the following complementary research directions:

- Informing the design of Virtual Environments
- Development of appropriate input techniques and devices for user movement in Virtual Environments
- Development of VE navigation and wayfinding support tools.

2.2.4.1. Informing the design of Virtual Environments

This research direction is concerned with the development of appropriate guidelines (mainly by exploiting existing environmental design principles) for the creation of well-structured spaces that inherently aid orientation and wayfinding. For example, Charitos (1997) presents a taxonomy of possible objects in VEs (namely landmarks, signs, boundaries, and thresholds), as well as of the spatial elements that these objects define (places, paths, inter-sections, and domains), and suggests how these elements can be used to support wayfinding, based on architectural design and on the way humans conceive and remember space in the real world. Along the same line of work, Darken & Sibert (1996) have studied wayfinding strategies and behaviors in large VEs, and suggest a set of environmental design principles that can also be applied in VEs. Hunt & Waller (1999) examined the relation of orientation and wayfinding between physical and virtual worlds, and the way existing knowledge can be transferred from the former to the latter, while Vinson (1999) offers a set of design guidelines for the placement of landmarks in a VE in order to ease navigation, based on concepts related to navigation in the physical world. On the other hand, research

findings presented by Satalich (1995) lead to the observation that human behavior with regards to navigation in the real world is not identical to the behavior exhibited in VEs, and thus, it is likely that existing tools and principles may not be adequate or sufficient if directly transferred from one domain to the other.

A basic limitation of these approaches is that they require the modification of the virtual space's contents. This may not always be possible or desirable, as, for example, in the case of VEs that are based on real world environments, thus rendering these approaches inappropriate for a large number of widely used VE applications, such as simulations, engineering design and architectural walkthroughs.

2.2.4.2. Development of appropriate input techniques and devices for user movement in Virtual Environments

This research direction aims to provide easier and more intuitive navigation in VEs through the definition and development of appropriate hardware, as well as of related input techniques and metaphors that allow the user to move more 'naturally' in a VE. For example, the Omni-Directional Treadmill (Darken et al., 1997) and the Torus Treadmill (Iwata & Yoshida, 1999) aim to offer novel hardware solutions for naturally walking or jogging in a VE. Peterson et al. (1998) propose a new input device in the form of a body-controller interface called *Virtual Motion Controller* (VMC), and compare its performance in navigating a virtual world with the performance of a joystick. Templeman et al. (1999) present *Gaiter*, another input device and an associated interaction metaphor that allows users to direct their movement through virtual environments by stepping in place. A different approach is followed by Razzaque et al. (2002), who introduce a new interaction technique supporting locomotion in a VE, named *Redirected Walking* that does not require any special hardware interface.

2.2.4.3. Development of VE navigation and wayfinding support tools

The third research direction includes techniques and tools that are not directly integrated in VEs, but come in the form of additional virtual objects aiding users to

identify their current (or target) location in a virtual world, as well as to construct an overview or mental model of the overall VE. A variety of such tools have been developed, including position coordinate feedback (Darken & Sibert, 1993), 2D maps (Darken & Sibert, 1993), 3D maps (e.g., the *Worlds in Miniature* (WIM) metaphor by Stoakley, Conway & Pausch, 1995), metaphors for exploration and virtual camera control (Ware & Osborne, 1990), dedicated windows offering alternative viewpoints (e.g., the *Through-the-Lens techniques*, Stoev et al., 2001), 3D thumbnails for providing memorable destinations to return later (e.g., *Wordlets* by Elvins et al., 1997) and position coordinate feedback that mimics global positioning systems (Darken & Sibert, 1993).

2.2.4.4. Discussion

The work performed in the context of this thesis in the domain of virtual environments is mostly related to the research direction presented in section 2.2.4.3. A relevant concept is that of *breadcrumbs* (Darken & Sibert, 1993), which are markers in the form of simple unmarked cubes hovering just above the ground plane. They can be dropped (manually or automatically) at the user's path as a means for marking trails or encoding locations, but do not offer any directional information or interactivity. The notion of breadcrumbs is also referred to as *trailblazing* (Edwards & Hand, 1997) and *Hansel & Gretel technique* (Darken & Peterson, 2002). Edwards & Hand (1997) have also suggested that such a technique could be extended to support the notion of 'embodied bookmarks', allowing the user to record and return to specific previously marked positions. More recently, Darken & Peterson (2002) revisited the notion of breadcrumbs by adding directional cues to them, admitting that "*a better analogy is that of footprints since footprints are directional and breadcrumbs are not*". They argue that the footprints technique can be useful for exhaustive searches, since it allows users to know if they have already been in some place before. Furthermore, they report two related problems, namely that: (a) as navigation proceeds, the environment becomes cluttered with footprints; and (b) when the user crosses paths, it becomes difficult to disambiguate which footprints belong to a given trail. Finally, the concept of *environmental landmarks* that can be explicitly placed by the user (Darken & Peterson, 2002), and the notion of private and shared annotation in 3D environments (Thomas et al., 2002), also share a few common attributes with the

presented work. In conclusion, the main issues that have been moderately explored by existing work are: (a) marking trails, and (b) creating personal landmarks and annotations in VEs.

The work presented in Chapter III builds upon and extends these efforts by introducing new concepts, such as the Virtual Handprints (leaving tracks of the user's interaction with, as opposed to mere movement in, the environment), as well as by proposing uniform, appropriate, ways and mechanisms for interacting with all types of virtual prints and for utilizing them in an effective and efficient way. It is also worth observing that the most prominent characteristic of the presented approach, in comparison to previous work, is that instead of being passive visual aids, virtual prints are interactive and configurable. Users can manipulate and use them to navigate, access various types of information, communicate, annotate a virtual environment and perform several additional actions.

2.3. ACCESSIBLE COMPUTER GAMES

Worldwide, at least one person out of ten is disabled due to physical, mental or sensory impairments (i.e., more than 500 million people worldwide), while at least one out of four is adversely affected by disability (UN, 2004). In addition, aging very often results in limitations in vision, hearing, memory, or motor functions, and world-wide there are around 600 million persons aged 60 years and over, a number that is estimated to double by the year 2025 and reach 2 billion by 2050 (WHO, 2004). Unsurprisingly, the number of older people playing computer games also increases. In 2003, 41% of most frequent game players were over 35 years old (ESA, 2003).

Nowadays, computer games have become one of the major sources of entertainment. For example, one out of two Americans play video games, and in the USA the sales of games outnumber the sales of books (Digiplay, 2004). Furthermore, the world market for video games continues to grow at much faster rates than the cinema box office, VHS/DVD rental and music retail (Screendigest, 2004). More recently, there is a serious trend for introducing games for training and learning (known as *game-based learning*, Prensky, 2000) in order to take advantage of the unparalleled motivation and engagement that computer games can offer to learners of all ages.

Unfortunately, most existing computer games are quite demanding in terms of motor and sensor skills needed for interaction control, while they often require specific, quite complicated, input devices and techniques. This fact renders computer games inaccessible to a large percentage of people with disabilities, and in particular to the blind, and those with severe motor impairments of the upper limbs.

2.3.1. Computer accessibility

Computer accessibility is usually associated with access to interactive computer-based systems by people with disabilities. In the past few years, there has been an ever-increasing interest in accessibility issues and especially Web Accessibility. This can be attributed to several reasons:

(a) the advancement, wide availability and affordability of Information and Communication Technologies;

(b) the evolution of the World Wide Web into a key resource for information and interaction in all areas of the society;

(c) the initiatives of non-governmental organizations, such as the Web Accessibility Initiative (WAI, 2005) of the World Wide Web Consortium (W3C) in the field of accessibility.

As a result, today there is increased awareness about accessibility, as well as several related guidelines, methods and tools for designing, evaluating and repairing software accessibility. Furthermore, computer accessibility is gradually introduced in the legislation of several countries. For example, in the USA, since 1998, section 508 of the Rehabilitation Act (Access Board, 2000) requires that "any electronic information developed, procured, maintained, or used by the federal government be accessible to people with disabilities". In Europe, the eEurope action plan (European Commission 2002) and related resolutions of the European council commit the Member States and European institutions to design public sector web sites and their content to be accessible, so that citizens with disabilities can access information and take full advantage of the Information Society.

2.3.1.1. Disabilities affecting computer accessibility and related solutions

There are several types of disabilities that can affect computer accessibility. Although there is no single universally accepted classification, an indicative list of impairments includes the following:

(a) Visual impairments

This category comprises blindness, low vision and color blindness. Blindness implies a total (or almost total) loss of vision in both eyes, while low vision (or partial sight) implies severe visual impairment that cannot be totally corrected through the use of visual aids (e.g., glasses or lenses). People with low vision may be able to read text and distinguish forms, objects and pictures under specific conditions (e.g., very large fonts, high contrast, particular lighting conditions) but usually also rely on other senses, such as hearing and touch. Color blindness refers to the inability to discriminate differences in colors, mainly red and green.

(b) Motor or dexterity impairments

Motor and dexterity impairments include the total absence of limbs or digits, and paralysis, lack of fine control, instability or pain in the use of fingers, hands, wrists, or arms. Individuals with motor impairments mainly face difficulties in using standard input devices, i.e., the keyboard and the mouse.

(c) Hearing disabilities

Hearing disabilities may range from total deafness (i.e, the person is not able to hear at all), to slight loss of hearing (the person can sense sounds and speech, but finds it hard to identify their content). Deaf people communicate using sign and written language, while hard of hearing individuals may rely on lip-reading and hearing-aids.

(d) Cognitive disabilities

This is a very broad category, which roughly includes difficulties in the performance of mental tasks. These can range from limited and focused problems affecting a very specific cognitive function (e.g., the ability to understand math), to severe cases (e.g., brain damage) were the individual is unable to take care of daily living activities. The

most common types of cognitive disabilities are[5]: mental retardation, language and learning disabilities (e.g., dyslexia), head injury and stroke, Alzheimer's disease (i.e., memory retention problems) and dementia. Cognitive impairments are often localized (Seeman, 2002) and thus people with cognitive impairments can be of average (or above) intelligence. Sometimes an impaired cognitive function maybe combined with one or more overdeveloped cognitive functions.

(e) Speech impairments

Speech impairments are quite rare and sometimes are combined with other disabilities but they do not indicate limited intelligence. Individuals with speech impairments may have articulation problems (e.g., stuttering), be unable to speak loudly or clearly, or even to speak at all. Obviously, they have problems in using speech recognition systems. Depending on the severity of their case, they may use communication aids, to substitute speech.

(f) Illiteracy

Illiteracy is the lack of ability to read and write in any language. Although illiteracy is not a disability, it creates considerable barriers to computer accessibility and is often treated in the overall context of computer accessibility.

According to a study commissioned by Microsoft Corporation and conducted by Forrester Research, Inc. (Microsoft, 2003), among adult computer users in the USA:

- One in four (25%) have a visual difficulty or impairment.
- Nearly one in four (24%) have a dexterity difficulty or impairment.
- One in five (20%) have a hearing difficulty or impairment.
- About one in seven (16%) have a cognitive difficulty or impairment.
- Very few (3%) have a speech difficulty or impairment.

Age-related disabilities are frequently referred to as a separate category, but all related problems fall within the above categories. Disability-related accessibility problems are usually tackled through a combination of (Figure 5):

[5] http://trace.wisc.edu/docs/software_guidelines/software.pcs/disabil.htm

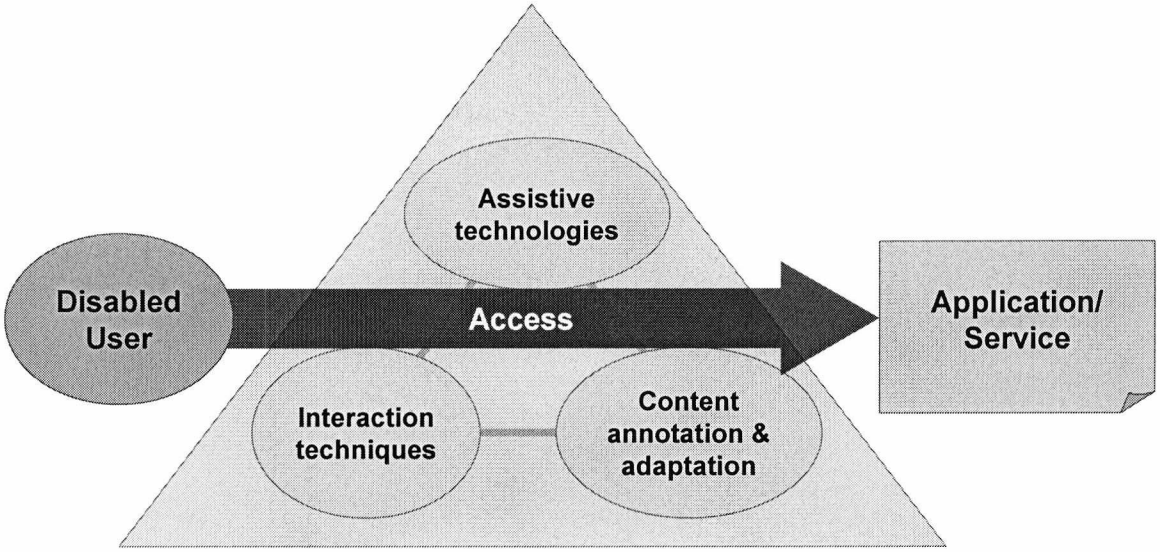


Figure 5: Approaches for tackling accessibility problems

(i) ***assistive technologies***, i.e., devices that are suitable for, or compensate to some extent for a specific disability;

(ii) ***interaction techniques***, that on the one hand are appropriate for the disabled person's interaction capabilities and needs and, on the other hand, can work with, and take advantage of, any available assistive technologies;

(iii) ***content annotation and adaptation***, so that it can be rendered in a format that can be optimally perceived and used through the employed assistive technologies and interaction techniques.

The basic strategies for making computers accessible by each impairment category are the following (W3C, 2005; IBM, 2002; NDA, 2001; Access Board, 2000):

(a) ***Visual impairments***

1. Content annotation with semantic information.
2. Provision of the content through alternative modalities, such as audio and tactile (in the form of Braille).
3. Support for content enlargement, e.g., control of font size, zooming facilities.
4. Customization of color combinations to improve contrast and simplification of visual complexity (e.g., replacing background images with solid colours) to improve legibility.

5. Support of input through the keyboard, Braille devices and speech.
6. Ability to serially and hierarchically browse the content and interaction objects in a logical order.

(b) ***Motor or dexterity impairments***

7. Support / provision of alternative input devices and techniques, such as switches, specialized keyboards, mice, trackballs and joysticks, scanning, visual keyboards and speech.
8. Speed and timing control and adjustment to suit different response times.
9. Ability to serially and hierarchically browse the content and interaction objects in a logical order.

(c) ***Hearing disabilities***

10. Visual representations of auditory information.
11. Augmentation of speech with sign language.

(d) ***Cognitive disabilities***

This is probably the hardest category since sometimes, depending on the type and level of disability, solutions must be provided at an individual basis. In general, all related solutions include:

12. Provision of alternative (simplified, illustrated) versions of the content.
13. Simplification of tasks, e.g., through step by step procedures and wizards.
14. Avoidance of blinking and flashing at particular rates that can cause photosensitive epileptic seizures in susceptible individuals.

(e) ***Speech impairments***

15. Support of alternative input / communication methods when speech is required.

(f) ***Illiteracy***

16. Content simplification.
17. Provision of textual content through illustrations, audio and video.

It should be noted that each impairment has different severity levels, possibly requiring different solutions, and that sometimes people have combinations of disabilities, thus raising compatibility issues between individual approaches.

2.3.2. Design for All and Universal Access

The term *Design for All* (or *Universal Design* - the terms are used interchangeably) is rooted in engineering disciplines, such as, for example, civil engineering and architecture (Story, 1998). In the context of HCI, the term is defined as (Stephanidis et al., 1998): *"the conscious and systematic effort to proactively apply principles, methods and tools, in order to develop IT&T products and services which are accessible and usable by all citizens, thus avoiding the need for a posteriori adaptations or specialised design"*. Design for All promotes a design perspective that eliminates the need for "special features" and fosters individualisation and end-user acceptability. Design for all, in contrast to the common practice of designing a single solution for an illusionary "typical" or "average" user, suggests the development of products integrating numerous alternative solutions that allow them to adapt in order to suit the broadest possible end user population.

Universal access (Stephanidis et al., 1998) implies the global requirement for computer accessibility by individuals with different abilities, requirements and preferences, in a variety of contexts of use. Universal access is not only concerned with people with disabilities, but with every aspect of diversity that may affect computer accessibility, such as: (a) the target user population profile (including people with disabilities) and their individual and cultural differences; (b) the scope and nature of tasks performed; (c) the technological platforms and associated devices (including assistive technology) through which information is accessed.

2.3.3. Key Research Issue: Universally Accessible Games

Following the principles of Universal Access and Design for All, a key research issue for the domain of computer games is the creation of *universally accessible games*, i.e., games that:

(a) are proactively designed to optimally fit and adapt to different individual gamer characteristics without the need of further adjustments or developments;

(b) can be concurrently played among people with different (dis)abilities even when sharing the same computer;

(c) can be played on alternative technological platforms and contexts of use using a large variety of devices (including assistive technologies).

The underlying vision is that through such games people will be able to have fun and compete on an equal basis, while interacting easily and effectively, irrespective of individual requirements, skills and preferences, the technology used and the current user location. Furthermore, this approach has the potential to render accessible several "physical" games that in their original form are not accessible to several groups of people with disabilities (e.g., the "physical" chess game is not accessible to the blind or the motor-impaired).

The above are related to the challenges of designing for accessibility [C4] (and more specifically for Universal Access), as well as of designing based on incomplete knowledge [C5] (see Chapter I).

2.3.4. Related Work

2.3.4.1. Computer games accessibility

In contrast to Web accessibility, relatively few efforts have been devoted to game accessibility. Currently there are no related official guidelines or standards, nor any world-wide initiatives comparable to W3C-WAI promoting game accessibility, and evidently no related governmental or legislative actions. Game accessibility is mainly a concern of groups of disabled people (e.g., Audyssey on-line gaming magazine and AudioGames.net for the blind[6], DeafGamers[7] for the deaf) or companies producing related products (e.g., GamesForTheBlind.com and BSC GAMES[8] for the blind,

[6] http://www.angelfire.com/music4/duffstuff/audyssey.html
[7] http://www.deafgamers.com/
[8] http://www.bscgames.com

Arcess[9] and Brillsoft[10] for the motor-impaired). Public awareness is quite low and limited to a handful of Web articles (e.g., Williamson, 2003; O'Modhrain, 2004; D'Amico, 2001). The only related organized effort is the Game Accessibility Special Interest Group[11] of the International Game Developers Association (IGDA), formed in 2003 with the aim "*to develop methods of making all game genres universally accessible to all, regardless of disability*".

Indicative is the fact, that, till recently, there was no definition of the term game accessibility. In 2004, the Game Accessibility SIG (GA-SIG, 2004), defined *game accessibility* as "*the ability to play a game even when functioning under limiting conditions. Limiting conditions can be functional limitations, or disabilities — such as blindness, deafness, or mobility limitations*".

A drawback of this definition is that it does not take into account people who may not have a disability, but may have "functional limitations" due to their current context of use (usually called "situationally-induced impairments and disabilities - SIID"; Sear et al., 2003). For example, a person in a very noisy environment (e.g., a train) is situationally deaf, someone using a very small screen has deteriorated vision, and lack of enough or adequate space for using a mouse can create temporary "motor-impairments". Thus, an all-encompassing definition would rather state that game accessibility is "the ability to play a game even when functioning under limiting conditions, irrespective of their cause".

Another important issue, not tackled by the suggested definition, is the quality of interaction or *quality in use* (Bevan & Azuma 1997; ISO, 1998), which concerns the extent to which a system meets the real world needs of its intended users and the support it provides to achieve their particular goals. The notion of quality is a user-perceived attribute of the system, and goes beyond the "traditional" concept of usability (i.e., effectiveness, efficiency, satisfaction, etc), to include aspects such as usefulness, suitability for the task, tailorability, etc. Requiring just the provision of access is not enough, especially in such an interaction-intensive domain as computer

[9] http://arcess.com
[10] http://www.brillsoft.com
[11] http://www.igda.org/accessibility

games. For example, a chess game can be made accessible to a blind person by providing just an oral description of the whole board, and of the currently selected piece. However, this means that every time this person wants to access even some simple game-related information (e.g., find out if the selected piece can move to a specific square), will first have to listen to the description of every square on the board and then mentally isolate the desired piece of information. Obviously, such an approach would create unnecessary high mental workload and very prolonged interaction times. Thus, beyond simple access, the player should also be provided with augmented interaction capabilities that meet personal characteristics, needs and preferences. In the specific example, the blind player should be able to quickly and easily access in a non-visual format any piece of game-related information that sighted players can infer by looking at the visual interface.

Currently, the only support available to developers for creating accessible games is limited to some indicative approaches to game accessibility briefly described in a White Paper of the IGDA Game Accessibility SIG (GA-SIG, 2004), as well as general-purpose guidelines for developing accessible software, such as the IBM Software Accessibility Checklist (IBM, 2002), the W3C-WAI Accessibility Guidelines (W3C, 2004), Parts 1194.21 and 1194.22 of Section 508 of the Rehabilitation Act Amendments (Access Board, 2000), the Irish National Disability Authority IT Accessibility Guidelines (Irish National Disability Authority, 2001), etc.

The main techniques currently used to provide game accessibility can be summarized as follows (GA-SIG, 2004):

(a) Visual impairments

1. Use of standard text for presentation instead of graphics, so that it can be rendered by assistive technologies (e.g., screen readers).
2. Integration of self-voicing capabilities.
3. Customizable fonts.
4. Speech feedback.
5. Support of keyboard-driven interaction.

6. Alternative color schemes, including high-contrast modes and support for full color customization.
7. Non-visual navigation aids for 3D games, e.g., audio sonar and GPS.

(b) Motor or dexterity impairments

1. Support of keyboard-driven interaction.
2. Control customization.
3. Support of "special" input devices.
4. Speed and difficulty adjustment.
5. Simplified versions of the user interface.

(c) Hearing disabilities

1. Subtitles.
2. Visual feedback.
3. Alternative sound selection (that may aid the hard of hearing).

(d) Cognitive disabilities

1. Guidance.
2. Context-sensitive help.
3. Speed and difficulty adjustment.
4. Simplified versions of the user interface.

(e) Speech impairments

1. Support of alternative input / communication methods when speech is required.

2.3.4.2. Technical approaches

From a technical point of view, two main approaches have been adopted in order to address the issue of computer games accessibility in general:

1. Games are developed to be compatible with the use of assistive technologies, such as screen readers, mouse emulators or virtual keyboards (a solution that can practically work only with non-action games, which do not rely upon fast reflexes and user reactions).
2. Special-purpose games are created, optimally designed for people with disabilities, like audio-based games for the blind, switch-based games for the motor-impaired, etc.

The first "reactive" approach typically suffers from low interaction quality access, and achieves limited accessibility (Stephanidis, 2001a). The second approach, though being the most promising from the quality point of view, has two key drawbacks: (a) the cost of developing high quality games is prohibitive when the potential target group is so limited; and (b) there is an evident hazard of segregation between able-bodied and disabled gamers, leading to potential social exclusion.

A more recent trend is the development of games for both visually-impaired and fully sighted, as for example the online card games developed by *All inPlay*[12] and the game Terraformers[13] by Pin Interactive, a visual / audio hybrid 3D game that can be played with its visual 3D graphics layer on or off, and is intended to support players with all degrees of visual ability or impairment through a sophisticated layer of sound. Although these games can not be considered to be universally accessible, they can provide valuable insights in solving related design problems, help in promoting the overall concept of game accessibility and prove that there is a related public and market interest.

2.3.4.3. Accessible chess games

Currently there are two chess programs for Windows and two (one in beta version) for Linux that can be made accessible to the blind or the visually impaired. WinBoard[14] is a free graphical chessboard that can serve as a user interface for GNU Chess or the Internet Chess Server, and can also be used to play out games manually or for loading

[12] http://allinplay.com/
[13] http://www.terraformers.nu
[14] ftp://ftp.freedomscientific.com/users/hj/winboard/WinBoard.exe

game files. Winboard is keyboard-driven and is adapted for compatibility with the JAWS for Windows[15] screen-reader, through which it can offer several screen review options. KChess[16] is a widely-used and very powerful commercial chess program that also offers lots of features to help in learning to play better chess. It can also be controlled using the keyboard and can provide some basic game information using speech, as well as extensive textual information about the state of play, which can be picked up by screen reader software, such as Window-Eyes[17]. Additionally, the game screen can automatically adjust when Windows is set to use large fonts. GNU chess[18] is a free chess game for the Linux operating system, that can be used with the Speakup[19] screen review package for the blind. Emacs-chess[20] is a chessboard display (in beta version) which allows playing chess via Internet Chess Servers against other human opponents from within Emacs (a popular Linux text editor) and, in combination with Emacspeak[21] (a free Linux speech interface), and can be played by the blind.

Concerning motor-impaired people or people with combinations of impairments, there are currently no chess programs addressing their particular needs. Furthermore, there are no computer games in general that can be concurrently played by people with different disabilities sharing the same computer.

2.4. SOFTWARE AGENTS

Until recently, the main paradigm of computer use was that of explicit instruction. The software provided just an interface to a number of functions, and in order for the users to achieve their goals they had to perform every action required by themselves. The computer would do exactly what commanded, and would not try to interfere or offer active support in any way. However, this paradigm, also known as *direct manipulation*, works only when the users already know how to translate their goals to

[15] http://www.freedomscientific.com/fs_products/software_jaws.asp
[16] http://www.arkangles.com/kchess
[17] http://www.gwmicro.com/products
[18] http://www.gnu.org/software/chess
[19] http://www.linux-speakup.org/speakup.html
[20] http://emacs-chess.sourceforge.net
[21] http://emacspeak.sourceforge.net

computer commands (which means that they can be considered as experts) and also when the information the user acts upon is limited and readily available.

Currently, as on the one hand the vast majority of computer users are non-experts, and, on the other hand, information is available at overwhelmingly big amounts due to the growth and use of Internet and the World-Wide Web, there is an increasing need for goal-driven computer applications, i.e., applications that based on a user goal, can act on behalf of the user deciding how it is best to carry out all the required steps to accomplish the related tasks. To cater for this need, a novel paradigm of human-computer interaction has been suggested: *software agents*.

Software Agents (also known as *intelligent agents* or *softbots*) originate from the research field of Artificial Intelligence (AI) and follow a human-like approach for interacting with their (virtual or real) environment and the computer user(s). Their intrinsic characteristics, beyond the development of more "intelligent" software, also allow for the creation of believable, life-like, virtual characters that can be exploited for radically enhancing the experience and value provided by entertainment software, which presently represents a vast and rapidly growing industry.

This section first provides a brief introduction to the field of artificial intelligence and a more elaborate account of the concepts of intelligent and embodied agents. Then, it presents an open Key Research Issue faced by the field that is related to this thesis, along with (the few) existing related efforts.

2.4.1. Artificial Intelligence

According to the Webster's dictionary, *intelligence* is "the ability to learn or understand or to deal with new or trying situations" but also "the ability to apply knowledge to manipulate one's environment or to think abstractly as measured by objective criteria (as tests)", while according to the Encyclopedia Britannica, it is "the ability to adapt effectively to the environment, either by making a change in oneself or by changing the environment or finding a new one". Intelligence is a quality mainly attributed to humans, but animals – and even plants – are considered to have some, rather limited, form of intelligence, that may vary according to the species.

The American Association for Artificial Intelligence (AAAI) on its home page[22] defines Artificial Intelligence as the "*the scientific understanding of the mechanisms underlying thought and intelligent behavior and their embodiment in machines*", while John McCarthy, one of the founders of the field of AI, defines it[23] as "*the science and engineering of making intelligent machines, especially intelligent computer programs*". In contrast to AAAI, McCarthy notes that although AI is related to the task of using computers to understand human intelligence, it does not have to confine itself to methods that are biologically observable. Thus, it could be said that artificial intelligence is the domain of science that aims to create intelligent hardware and software systems, based on, but not limited to, re-creating or emulating biological (i.e., human and non-human) intelligence.

Although John McCarthy was the one who coined the term artificial intelligence, as the topic of a conference held at the summer of 1956 at Dartmouth College in Hanover, New Hampshire, the first known researcher on the topic was the English mathematician Alan Turing who gave a relevant lecture in 1947. Turing also suggested a test (known as "the Turing test") to judge whether a digital computer can be considered as "intelligent". The test consists of an observer that has to tell apart a human from a computer mimicking human behaviour, through answers they both provide to questions through teletype. If the observer cannot distinguish the machine from the human, then the machine is considered to be intelligent.

2.4.1.1. Research Domains

One of the earliest research domains of AI, tracing back to the 70's, are *expert systems* (Payne & McArthur, 1990), which are also known as *knowledge-based* o *rule-based* systems. Expert systems store human know-how (according to a specified format) in the form of rules and data in a knowledge-base and then use an inference mechanism to predict the probability of a solution under set conditions. Expert systems have been widely used in the domain of medical informatics, for diagnosis and treatment. In fact, one of the first expert systems was MYCIN in 1974, which diagnosed bacterial

[22] http://www.aaai.org/
[23] http://www-formal.stanford.edu/jmc/whatisai/node1.html

infections of the blood and suggested treatments. One more renowned example of this domain is IBM's Deep Blue chess playing supercomputer[24] that in 1997 won the world chess champion Garry Kasparov.

Another popular domain is *natural language processing* (NLP) (Pereira & Grosz, 1994). NLP involves making computers to understand and generate natural human language. Although existing NLP systems are far from achieving human ability, there are already some NLP technologies widely used today, as for example speech analysis and recognition, natural language search systems and machine translation programs. One of the most famous AI programs was ELIZA (Weizenbaum, 1966), that tried to assume the role of a psychoanalyst talking with a patient.

The domain of *computer vision* (Boyle & Thomas, 1988) aims to extract, analyze and interpret information received from the physical environment, through sensors as video cameras, but also heat and proximity detectors, sonars, etc., in order to allow computers to perform human-related tasks, ranging from tedious, repetitive routines such as production line quality control, video surveillance, and identifying people by their characteristics (biometrics), to automatically driving vehicles or controlling robots. This is why this field is closely related with *robotics*, a discipline which aims at creating machines (robots) that can act in an intelligent way and autonomously perform tasks.

Since intelligence requires acquiring knowledge upon to which to draw future decisions, a relevant research domain was developed, namely *machine learning* (Langley, 1996). This domain encompasses techniques to model data and to develop new knowledge based on past experiences in order to improve future performance. Machine learning minimizes the knowledge that has to be pre-fed in an AI application, saving valuable development time, allows adaptation to changing environments and unforeseen situations, and provides for dynamic optimization to real use parameters. For these reasons, machine learning is used by all other AI application areas.

[24] http://www.research.ibm.com/deepblue/

Finally, the most recent and dynamically evolving AI domain is that of *artificial life* (Magnenat-Thalman & Thalmann, 1994). This domain is concerned with the emulation of the behavior of real-world living organisms and the creation of synthetic creatures through a variety of methods, ranging from simple rules to genetic algorithms and neural networks. The underlying idea is that of, avoiding programming complex models and interactions by implementing small-scale independent components, each of which is specialized in performing a specific, well-defined task, and then combining and linking these components through decision-making and communications mechanisms, and letting complex behaviours emerge, as it is also deemed to happen in nature with real creatures. This domain shares a common ground with *robotics* and has originated the notion of *intelligent software agents*.

2.4.2. Intelligent Agents

There are several different definitions of the term (Franklin & Graesser, 1997) that usually correspond to the type of agents that are developed by the definer. Although there is much debate about what an intelligent agent is and what it is not, a common denominator of all definitions is *autonomy*, meaning that an agent is an entity that has control over its own actions. Furthermore, an agent is *reactive* and *pro-active* (Wooldridge & Jennings, 1995). *Reactive* means that the agent can sense changes in its environment and respond to them, whi le pro-active means that the agent has specific goals and takes initiatives to achieve them, not only responding to the environment. Finally, an agent must have *social ability* (Wooldridge & Jennings, 1995), so that it can interact with the user or other agents. In order to achieve the aforementioned characteristics, AI techniques from various domains are used. Additionally, several theories for conceptualizing agents, as well as software architectures and programming languages for implementing them, have been suggested.

Typically, the term is used to refer to "smart programs" to which the user (or the computer) can delegate a task, e.g., search for specific information on the Web, recommend a movie according to personal interests and help shopping on the Internet, or that can dynamically assist the user, e.g., providing some form of context sensitive help. Software Agents of this type that are "living" in the environment of the

computer memory without having a visual embodiment are sometimes called *softbots*, *inforbots* or *knowbots*.

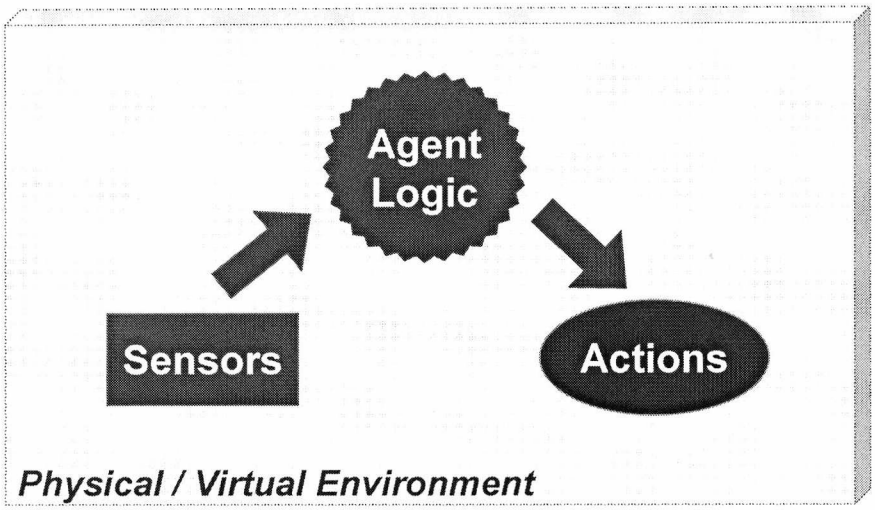


Figure 6: Abstract agent architecture

An abstract representation of the architecture of an agent is provided in Figure 6: an agent is always situated in an environment, the state of which (as well as any changes to it) are perceived be the agent through "sensors". Then, some kind of logic is used (e.g., a set of rules, a script, a neural network) to analyze this information, usually in combination with internal data (e.g., the agent's goals, state, characteristics, interaction history) and to decide upon the agent actions to the environment.

2.4.2.1. Embodied Agents

Intelligent agents can be much more than just software programs; they can also be manifested as animated characters that inhabit 2D or 3D virtual environments or even as robots situated in the real physical world. In this case, the agent is said to be *embodied*, i.e., it has a body that is subject to the rules of its surrounding environment and through which it can interact with it. Thus, in analogy to the agent architecture presented in Figure 6, an abstract depiction of the architecture of an embodied agent is presented in Figure 7: the brain analyzes data that are received through the body's sensors and decides and instructs the body for corresponding actions, while the body constraints the sensory data that the agent can receive, as well as its repertoire of actions. Embodied agents are also called *synthetic creatures* or *animats*.

Today, one of the most popular application areas of embedded agent technologies is the creation of virtual characters in movies. Instead of having to script every frame of animation, the synthetic characters are controlled by agents that can act in believable ways according to their characteristics. For example, the massive battle scenes of the Lord of the Rings movies were created by combining few real actors with hordes of computer generated soldiers, developed through Massive[25] 3D animation system, for generating crowd-related visual effects and character animation.

A much more challenging task than rendering passive 3D animations is that of creating life-like characters that can interact in real-time between them and / or with humans. There have been several research efforts towards this goal, such as, for example, the work of Reynolds (1993) on a vision-based behavioral model for controlling the motion of groups of creatures, the artificial fishes created by Tu & Terzopoulos (1994), the ALIVE project (Maes, et al., 1995) that studied full-body interaction between a human participant and virtual animated autonomous agents, and the Computer-Animated Improvisational Theater (CAIT) by Hayes-Roth et al. (1995), in which children and animated characters collaborated to create and act out stories.

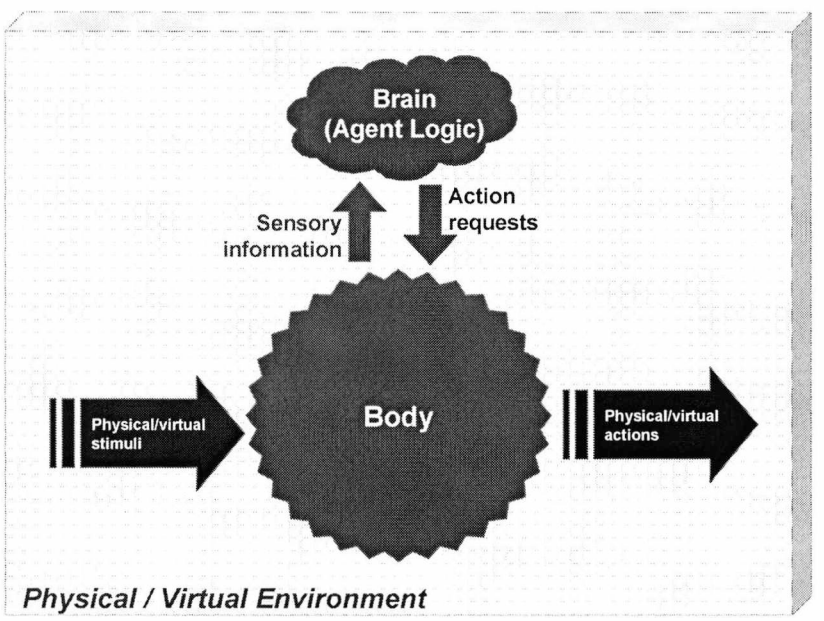


Figure 7: Abstract embodied agent architecture, where the thinking component ("brain") is explicitly separated from the agent's physical/virtual manifestation ("body")

[25] http://www.massivesoftware.com/

But except research efforts, the technology of intelligent agents is making its way to computer games. Common limitations of video games is the repetitive nature and the shallowness of non-player characters, that act in prescribed, easily predictable ways, making players to get quickly bored or even frustrated. The use of intelligent agents can have a profound impact on all games genres by providing human-like behaviours to enemies, partners, and support characters. The game industry has embraced embedded AI techniques, entitling them "new AI" or "nouvelle AI" (Champandard, 2003).

There have already been successful commercial programs utilizing *artificial life* techniques to construct virtual pets, such as *Creatures* (1996), by Millennium Interactive, *Virtual Petz* (1997) and *Babyz* (1999) by PFMagic. More recently, two games using related techniques, *The Sims* (2000) by Maxis Interactive and *Black and White* (2001) by Lionhead Studios, became extremely popular and profitable, raising the game developers and companies awareness and interest for the field, and their desire to integrate related technologies in their products. Furthermore, the academic AI community shows an ever increasing interest and respect in computer games (Laird & van Lent, 2001), which are now deemed as an ideal AI application domain, and where, due to the market competition, innovation is funded abundantly.

2.4.3. Key Research Issue: Sensing a Virtual World

A key requirement for achieving any form of intelligence is the ability to collect and interpret information about one's own environment, i.e., the ability of *sensing*. For *disembodied agents* (i.e., software programs) sensing is quite straightforward, since it simply means acquiring digital data from an electronic environment, a procedure that can be easily achieved through function calls, message exchanging or analysis of textual information, depending on the agent's task. Moreover, robotic agents use different types of hardware sensors and related techniques to sense the real world, such as video cameras, radars, infrared scanners, heat and collision detectors, etc. Thus for both cases, it is quite clear what needs to, or can, be sensed, and how this can be achieved.

But what about *embodied agents* that live in virtual worlds? In this case, there is no tangible information to be manipulated. There are just graphics (i.e., coloured pixels), animations and sounds, which can be perceived and attributed with semantic meaning only by a human. Thus, senses – but also what can be sensed, i.e., stimuli – need to be simulated through adequate models and processes.

A key research issue for the software agents domain is elaborating a generic approach to creating a synthetic sensory system, which will be applicable to any type of embodied agent in any type of virtual world, and will not depend on the nature or the intrinsic characteristics of the senses or stimuli that will be simulated.

This is related to extending the capabilities of a novel interaction paradigm (embodied software agents) [C1] providing higher design flexibility and more options to designers and also designing for openness and extensibility [C5] (see Chapter I).

2.4.4. Related Work

Although literature on almost any other aspect of agents, such as acting, navigating, decision-making, strategic thinking and learning, comes in abundance, ranging from academic research papers to step-by-step guides and technical articles, the published material on the subject of sensing is very limited both in size and detail. The two main sources of related information are: (a) research work on synthetic actors, usually aiming to accurately re-create human (or animal) senses, and (b) computer games, following more "relaxed" approaches to create the illusion of sensing creatures, but having as a main concern low design and implementation complexity, as well as the minimal use of computational resources, such as memory and processor load.

2.4.4.1. Research work

Vision

Reynolds' *critters* (Reynolds, 1993) use synthetic vision to avoid colliding with their fellows and obstacles or being killed by predators. The approach followed is quite simple: when a critter wants to look for something (obstacle, friend or predator) in a particular direction, a relevant function is called that casts a ray starting from the

critter' s position in the specific direction. Then, for every object of the requested type, it is calculated if it intersects with the ray. In case several objects are found, the one closest to the critter is considered to be visible and a numerical value representing the estimated distance is returned.

Tu & Terzopoulos (1994) have created a virtual marine world inhabited by artificial fishes. The fishes are equipped with two types of sensors: a vision and a temperature sensor. The vision sensor is modeled as a 300 degree spherical angle (Figure 8). An object is "seen" if any part of it intersects the sensor's volume and is not fully occluded by another object. To get more information about the object, the sensor can directly access data from the simulation engine (e.g., color, size, illumination, object id and velocity). The temperature sensor gets information about the temperature of the water at the center of the artificial fish's body.

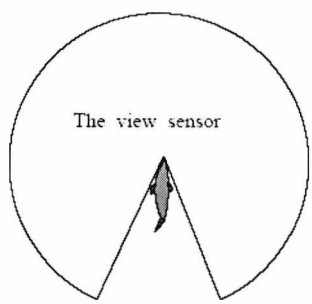


Figure 8: Artificial fish's vision sensor (Tu & Terzopoulos, 1994)

The ALIVE system (Maes et al., 1995) allows the interaction of a human with autonomous agents inhabiting a virtual world through body and hand gestures. A video camera captures the image of the human, and projects it in the virtual world, as a kind of "special" agent. Agents use vision sensors to detect other agents (including the user) and inanimate objects. Vision sensors cast several rays in a 2D plane across an arc of a specified angle. Then, for each ray, the closest point of intersection with another object is calculated, and the properties of the intersecting object are retrieved.

A much more elaborate approach to synthetic vision than the ones mentioned above was introduced by Renault et al. (1990), as a combination of image recognition

techniques and direct access of information from the virtual environment. A synthetic actor perceives the world through a small window in which the world is rendered from his point of view. Then, using on the one hand the color of the "visible" pixels (all objects are color-coded) and, on the other hand, information about relative distance, the actor tries to locate visible objects. To illustrate the capabilities of the synthetic vision system, the authors have developed several examples: the actor going out of a maze, walking on sparse foot locations and playing tennis.

Hearing

Noser & Thalmann (1995) use a real-time sound renderer to model the 3D acoustic environment, which at the same time passes semantic (e.g., identifier, source) and positional information about all currently active sounds to the audition sensors (the *ears*) of synthetic actors.

Touch

Huang et al. (1995) have suggested a method for allowing synthetic actors grasp objects based on attaching multiple sensors that are modeled as spheres (for their efficiency in collision detection) to the articulations of a virtual hand (Figure 9). When grasping an object, the sphere sensors detect collision with the object or other sensors in order to position the fingers naturally around its surface.

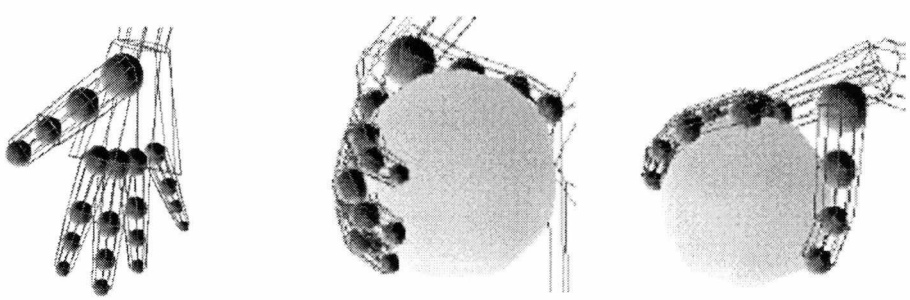


Figure 9: A virtual hand with sphere sensors at each joint (left) and an example of using sensors in grasping (right) (Huang et al., 1995)

2.4.4.2. Computer games

In most games sensing is restricted to some type of "vision", i.e., a way to detect other creatures or objects and their distance from the sensing agent. This is typically represented by a two-dimensional view cone, and a line-of-sight method (e.g., ray casting) is used to detect whether an object is visible or occluded by another one. Furthermore, all games support *collision detection*, which although it is the digital counterpart of the sense of touch, is typically treated as a special animation function for disallowing creatures to go through solid objects or identifying hits by bullets and enemy blows. Three representative approaches to sensing in computer games are provided below.

Quake (id Software, 1996)

In *Quake* a very simple visual system is implemented (Dalmau, 2004). A monster assumes to see a player if the player is within a certain range, and simple algorithms are used to test for collisions with the game world.

Creatures (Millenium Interactive, 1996)

Creatures provides a simulated environment in which a user can interact in real-time with a number of synthetic agents. The creatures' behavior is controlled by genetically-specified neural networks interacting with a genetically-specified biochemical system. Creatures have simulated senses of sight, sound, and touch, modeled using semi-symbolic approximation techniques (Grand, Cliff & Malhotra, 1997). For example, if a certain object is within the line of sight of a creature, a neuron representing the presence of that object in the visual field becomes active. Sounds attenuate over distance and are muffled by any objects between the creature and the sound-source.

Half–life (Sierra Online, 1998)

Half-life is a very popular first-person shooter where the player assumes the role of a scientist in a Research Facility, having to fight against hundreds of strange monsters, which were the result of some mysterious experiments that went wrong. Half-life characters are equipped with a simple sensory system, comprising vision, hearing and smell (Leonard, 2003).

Vision is modeled using a distance, a view cone, line-of-sight, and eye position (Figure 10). More specifically:

(a) The creature periodically tests if there is another entity within a specified distance.

(b) If there is one and the creature is interested in that entity, it checks if the entity's position is within a predefined viewcone, i.e., a 2-dimensional triangle, representing the field of vision.

(c) If the previous step succeeds, a ray is cast from the creature eyes to the other entity's eyes. If the ray is not blocked by another object, then it is considered that the entity can be seen.

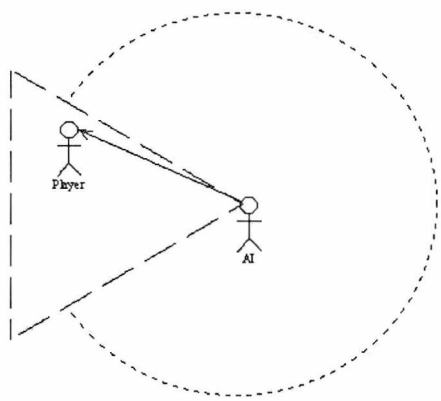


Figure 10: Modeling vision in *Half-life* (Leonard, 2003)

A much simpler procedure is used for hearing. When a sound is played, the distance within which it can be heard is calculated by multiplying the sound volume with a "hearing sensitivity" value. Smell is deemed as a special type of sound.

Thief (Looking Glass Studios, 1998)

This game is also a first-person shooter, but the gameplay revolves around the issues of stealth and strategy, instead of rapid action and massive kills. The player's goal is to stay unnoticed while accomplishing the objectives within each game mission. The player must try to move slowly, avoiding to make alerting noises and also crawl and hide in the shadows in order not to be seen by the AI-driven guards.

The game's sensory system in general resembles the approach followed in Half-Life, but it is much more elaborated in order to support the game requirements (Leonard, 2003). The main differences between the two are the following:

a) *Multiple 3D viewcones*: Instead of the single 2D viewcone used in Half-Life, Thief uses several 3D viewcones (Figure 11). Viewcones have different priorities, so that each time only one is used, depending on the position of the sensed entity, and each one of them can be parameterized to be sensitive to different types of stimuli and at different degrees. The use of multiple viewcones allows supporting direct vision, peripheral vision, and the distinction between objects directly forward and on the same Z plane as opposed to forward but above and below.

b) *Visibility values*: Each object that can be seen has a visibility value that depends on factors such as its lighting, movement, size, and separation from other objects.

c) *Multiple raycasts*: In some cases, (e.g., when the player is seen) multiple raycasts are used to assess in more detail the level of the seen object's exposure.

d) *More sophisticated sounds (and smells):* The entities can sense the direction of sounds. Also, sounds can "carry" semantic information (e.g., a message) that agents can understand. Smells again are treated as a special category of sound.

e) *Output values*: Instead of a Boolean value of the type "have sensed" / "have not sensed", the output of each sense is expressed as a level of certainty about the presence, location, and identity of an object of interest, to which the game developers refer as "awareness". This information can also be communicated and shared among different agents of the game.

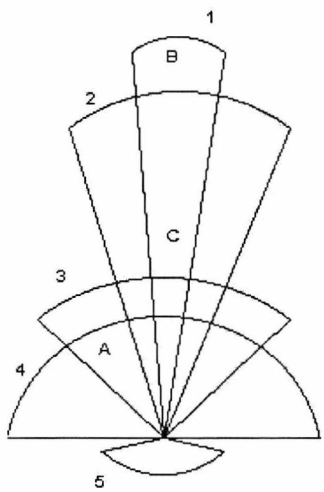


Figure 11: Multiple viewcones used in *Thief* (Leonard, 2003)

2.5. INTERACTIVE APPLICATIONS FOR YOUNG CHILDREN

Nowadays, children come into contact with modern technology from a very young age. In fact, it is quite frequent that children start using home computers and game consoles at the same time (or even before) they start learning how to read and write. This situation very often divides parents and educators in two opposite groups: the enthusiasts and the skeptics. The first, are the "believers", i.e., those who think that such an early encounter is definitely favourable for the children's mental and physical development, while the latter raise serious concerns about the possible related negative effects. In practice, as with any other form of technology, the truth is that interactive applications for young children can be as good or bad for their users as they are designed to be.

A typical design approach of the recent past was to regard children as "little adults" and try to "scale" or adapt adequately the existing corpus of HCI knowledge. But, although the high-level principles of "good" interaction design can be applied to practically any domain, it has to be noticed that most of the HCI design wisdom has resulted from the field of software applications for the workplace / office. This means that it is mainly concerned with the design and development of application software targeted to be used by adults. The key design goals are to maximize the users' efficiency and effectiveness and minimize errors, improving productivity, taking also care of issues related to ergonomics, as well as the user's health and safety.

Trying to directly re-use or transfer the related guidelines and principles to the development of children's software may have dubious results, since there are significant differences among the characteristics, requirements and preferences of the intended users and the respective contexts of use. Beyond the obvious bodily and mental dissimilarities, there are several important differentiation factors between young children and adults, such as:

(a) Motivation for using the technology

A fundamental difference between the two groups is the reason for which they use technology. In the workplace, the computers are used to get a job done. Young children, even when they use software for educational purposes, interact with technology for the sake of *enjoyment* (Inkpen, 1997). Adults are task-oriented and follow specific strategies and steps to achieve their goals. Children like to explore and play with technology.

(b) Interaction capabilities

There is significant evidence that children may have particular problems with specific interaction techniques employed by the windows-based graphical user interfaces. For example, Inkpen, et al. (1996) have studied children playing two versions of the same puzzle-solving educational computer game and noted that they had more difficulty operating a drag-and-drop type of interaction than a point-and-click interface and furthermore, that this effect is more pronounced for girls than for boys. Double clicks, mouse button distinctions and shift modifier keys can also be difficult for young children (Berkovitz, 1994). Berkoviz (1994) also observed that children had a lot of trouble when using a "marquee" rectangle to select multiple objects (a typical action required by drawing applications) and suggested an alternative, easier to use, version of the technique. Object manipulation can be also a problem when the objects are very small (Steiner & Moher, 1992) since children may have difficulty in targeting and selecting them. Also, object selection needs to be indicated by adequate visible and audible feedback (Steiner & Moher, 1992). Response speed is another important factor, since children may be confused (Steiner & Moher, 1992) or distracted if they are not provided with adequate feedback when the system is busy.

(c) Comprehension / adequacy of metaphors

Typical computer applications are based on the desktop metaphor and the office environment. This metaphor may make sense to adults who are familiar with folders, files, etc. but it is not appropriate for young children (Jones, 1990). For example, a metaphor quite popular with young children is that of a storybook (Steiner & Moher, 1992).

(d) Shoulder-to-shoulder collaboration

When referring to computer-supported collaboration in an office environment, remote collaboration is implied, i.e., several single users operating computers interconnected through a local or wide area network. At home or at school it is very common that more than one children share the same computer to achieve a common goal, in what is called *shoulder-to-shoulder collaboration.* Based on this observation, research has been conducted towards equipping a single computer with multiple input devices (e.g., Inkpen et al., 1995, Bricker et al., 1999, Stewart, Bederson & Druin, 1999a), from which the domain of Single Display Groupware (SDG) (Stewart et al., 1999) has emerged.

(e) Gender issues

When designing for young children gender issues are also important. Research findings (e.g., Passig & Levin, 2000; Coley et al., 1995; Durandell, Glissov & Siann, 1995) indicate that there is a significant difference in the level of satisfaction between boys and girls depending on the design of user interfaces. In general, boys prefer to have full control over the system and clear navigation mechanisms. Girls are more interested in the appearance of the interface and its colours and also care a lot about receiving at any point help on using it. Girls prefer calm games based on reflection and thinking, while boys strive for more action-based games with a lot of physical motion and dynamically moving interface elements. Further differences include the preference for specific colours (with girls finding more appealing red and yellow, while boys green and blue), or even in the shape of the interface's buttons.

A common mistake when designing for young children is the so called "Second Childhood Fallacy" (Oosterholt, Kusano & de Vries, 1996) where adults think they can explain why a child likes a particular feature without involving children in the design process, while this is very rarely the case. A related typical mistake of developers of new technologies is that instead of asking children directly about their needs and preferences, they rely on the "expert" opinion of their parents or teachers (Druin, 2002).

2.5.1. The role of children in the design process

According to Druin (2002) children can play four alternative roles in the design process: *user*, *tester*, *informant*, and *design partner*. These roles are not distinct, since each one of them extends the previous one(s), as depicted in Figure 12 (Druin, 2002). The first role (*user*) is the oldest and most widely employed. The children use / play with the technology while being observed by adults, who asses the usability of products and their impact on child users. When acting as *testers*, children usually interact with prototypes of new research or commercial products, in order to help shaping them before being fully developed, or released. In this case, the children except just being the subjects of observations, actively comment on their experience and answer to related questions posed by the evaluators. In the role of *informants*, children provide input to the very initial stages of the design process, for example by being observed with existing technologies, or by providing feedback on design sketches or low-tech prototypes. Finally, a more recent trend (Druin, 1999) is that children participate in the design process as equal *design partners* actively contributing their opinions and views and directly affecting its outcomes. This is a very challenging role, which is not easy to achieve (both for adults and children), but is also the most promising one, since it can have tremendous impact on the overall results of the process.

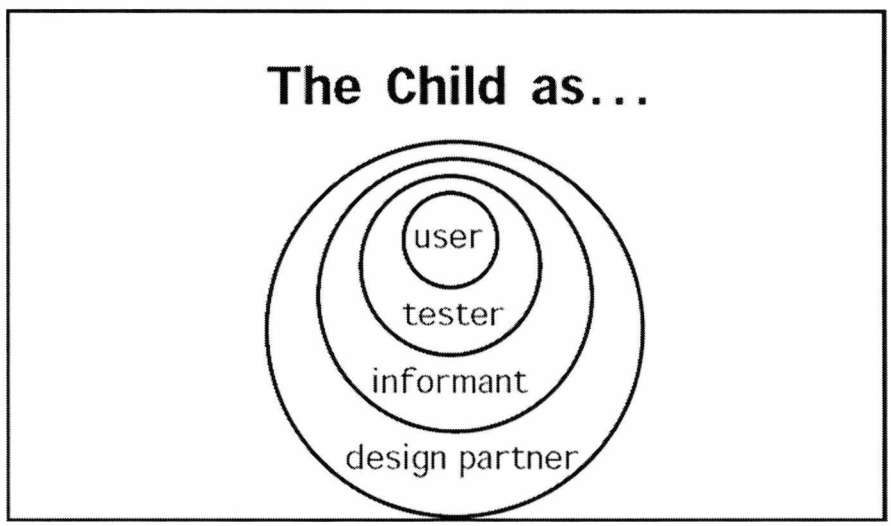


Figure 12: The four roles that children may have in the design of new technologies (from Druin, 2002)

2.5.2. Main characteristics of young children

According to the developmental psychologist Jean Piaget (1988), there are four stages of cognitive development for children: the *sensorimotor* (0 to 2 years), the *preoperational* (2 to 7 years), the *concrete operational* (7 to 11 years), and the *formal operational* (11 to adulthood).

In the *sensorimotor* stage, the child's knowledge of the world is based on physical interactions and experiences. The child constantly experiments and learns through trial and error, but it does not realize yet that objects exist even when out of sight. The child can differentiate itself from objects and starts to act intentionally.

The *preoperational* stage consists of two phases: the *preconceptual* (2-4 years) and the *intuitive* phase (4-7 years). In both phases, thinking and speech are egocentric and things cannot be viewed through the perspective of another person. In the *preconceptual* phase, children can use words and images to represent objects, but all objects sharing a common characteristic are deemed as identical. In the *intuitive* phase, intelligence is based on senses and intuition and not on logical or rational procedures. It is not possible for children to reverse procedures or tasks (reversibility), nor understand that the amount of a material does not change when its shape changes (conservation).

The *concrete operational* stage is characterised by the use of logical and systematic thought. Children can understand the concepts of reversibility and conservation, and also classify, order, and separate objects based on multiple characteristics. They furthermore comprehend time and related notions, such as sequences, the past and the future. Thinking becomes less egocentric.

In the *operational* stage, young adolescents are able to form and tests hypotheses, employ logic and think abstractly. They can make conclusions by going from specific observations to generalizations, such as finding analogies for constructing relationships between objects.

The work performed in the context of this thesis in the field of interactive applications for young children is mainly concerned with children 4-8 years old. At this stage of development, there can be significant differences even between children that have an age difference of a single year. An overview of the main characteristics of each distinct age group is provided below (Koutra et al., 2000):

a) Children of age 4-5

A characteristic of this age is 'action/play' (experience/learning through playing), that is, the children's need for motion and movement, a condition which usually limits the attention span and time of engagement in focused activities. A child of this age may well be in the company of other children, however, a high degree of co-operation should not be expected. The sense of time is not well developed yet (the present is well comprehended, but the future and the past can be not clearly oriented). Events are experienced in a highly subjective way. Children of this age do not co-operate for long times and do not stay in a queue.

b) Children of age 5-6

At this age, children cooperate more easily and act in a more conscientious way. The time span of their attention increases as is their structuring ability. Their vocabulary is richer so that they can describe events, feelings etc. They are aware of their 'image', develop social relations, control better their movements, show interest for reading and writing, and become more aware of the flow of time. Groups of 3 or 4 children can now form easily.

c) Children of age 6-7

At this age children are more developed. They can cooperate, observe, describe with ease. They can help themselves, control their bodies, relate better, choose friends, have developed spatial-temporal capability, can add and subtract, read and write. Teamwork becomes more prominent (working on a story or a play or a science experiment) as is archiving/retrieval through subjects and themes. Writing text is not a big problem.

d) Children of age 7-8

Children at this age can decide themselves for certain things and have well-defined interests. They can read and write well and they are more confident to explore. They want to create and can become competitive. Gender issues are emerging. Children argue and they can join together to argue, e.g., 'against' their teacher. Technologies come under their control. Layout environments can become more complex.

2.5.3. Key Research Issue: Social Multi-Perspective Interactive Tools for Young Children

Most of the current educational software usually focuses on "teaching" to children specific content and rarely offers either the child or the adult facilitator options for "deeper interaction" (Bouras, 2000). The term "deeper interaction" is used to refer to a procedure where the children take active control over the material presented to them, reflect upon this material, annotate it alone or in teams, and create their own version(s) of reality. In order to support such activities, the Today's Stories project (ESPRIT-i3-ESE Project No. 29312) funded by the European Commission in the framework of the Intelligent Information Interfaces (i^3), Experimental School Environment Programme, proposed a new software paradigm aiming at supporting different stages of reflection, levels of abstraction and opportunities for cooperation in grounded context that are accessible to young children. The ultimate goal was to enhance children's development both at cognitive, physical and social communication level.

The main novel aspects of the proposed system laid in its potential to support social interaction by allowing (Koutra et al., 2000):

- "Multi-User Multi-Perspective" capture of events.
- "Co-construction context and tools" that would facilitate and support exploration of these perspectives in a pedagogic context.

In this context, the key research issue consists of designing an interaction environment, suitable for young children 4-8 years old, which will empower them to

reflect on recorded events of their daily lives, presented through alternative perspectives and also annotate them collaboratively.

The above research issues are related to the challenges of designing the user interface of a non-typical application (multi-perspective collaborative application) [C2] (including the creation of new metaphors [C1]) for a non-typical user group (young children) [C3] based on several constraints and requirements stemming from the end-users (children), the domain experts (educators) and the technology used (hardware and software limitations) (see Chapter I).

2.5.4. Related Work

From an interaction designer's point of view, relevant bibliography on interactive applications of young children is extremely limited. Usually, when related work is reported in the literature, a best case scenario is that the final interface design is presented, sometimes along with the high-level process followed to achieve it, but typically a number of important issues are neglected or missing, such as:

(a) user interface design alternatives and decisions;

(b) the design rationale;

(c) empirical data concerning the usability and value of the designs after testing them with children, and on how testing affected the redesign of the system.

Some helpful general user interface design guidelines for children can be found in (Nicol, 1990) and (Druin, 1999b), as well as an illustrating example of a movie authoring and design system (targeted to 12- to 14-year-old children). Furthermore, a comprehensive list of guidelines for the design of educational software is made available by the Temple University, (1999), but it is oriented towards course-based software. Valuable input for the interaction design process can be derived from collections of guidelines on more 'conventional' user interfaces, such as (Galitz, 1997; Howlett, 1996; Norman, 1988; Shneiderman, 1998; Tognazzini, 1996; Weinschenk et al., 1997; Wood, 1998), as well as sources on cultural diversity issues (Galdo & Nielsen, 1996; Fernandes, 1995; Nielsen, 1990). Additionally, an interesting

insight and guidelines on direct manipulation and the "drag and drop" interaction method is available in Cooper (1995).

In order to develop the user interface of the Today's Stories project, the interfaces and interaction characteristics of several related research and commercial software applications were studied.

2.5.4.1. Research work

Research work related to the proposed system can be classified in three categories:

(a) Story writing environments for young children

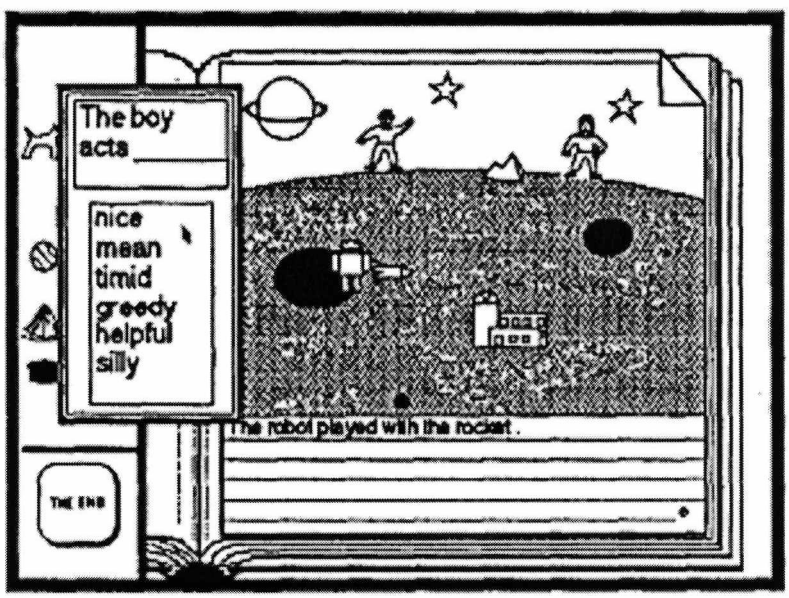


Figure 13: User Interface of the Graphic StoryWriter (Steiner & Moher, 1992). To the left, the pop-up menu for selecting character attributes is visible.

The Graphic StoryWriter by Steiner & Moher (1992) was an interactive system that enabled early readers "write" stories and learn about story structures through a simple point-and-click user interface. The children could select a background setting and then drop in it characters and objects (Figure 13). Based on these elements and their relative positions, a rule-based story engine automatically generated a relevant text. Whenever a character was added to the story, a pop-up menu appeared which allowed

children to select relevant descriptive attributes. When a valid object was selected a sound was played. When object placement was possible, the cursor changed from a selection pointer (a pointing hand) to a placement pointer (a large "X"). Characters and objects had associated animation and sound effects, since it was found that these could hold user interest and reinforce the desire to explore the software. Except viewing their creations on-line, children could also print them.

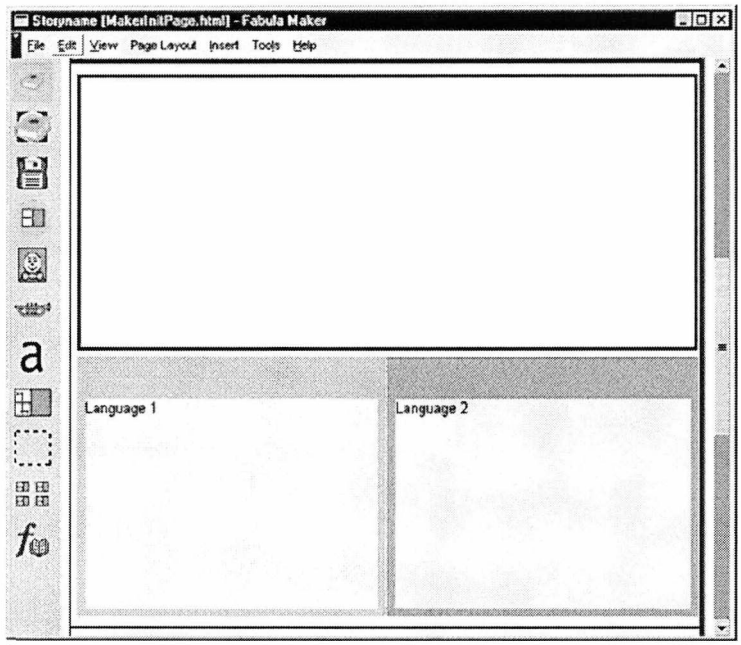


Figure 14: The user interface of the Fabula Maker (Edwards et al., 2002)

Figure 15: The user interface of the Fabula Reader (Edwards et al., 2002)

Fabula (Edwards et al., 2002) was a research project that aimed to develop educational software for learning European minority languages. The software developed consisted of an authoring tool for creating bilingual interactive multimedia digital books (Fabula Maker) and of a viewer application for reading the created story books (Fabula Reader). The user interface of the Maker is quite simple (see Figure 14). There are two separate text areas (one for each language) and a picture area. No formatting options are offered. Both texts can be accompanied by a sound file in which the text is read. Users can add several types of links. For example, they can add links from the picture to sound files (containing speech or effects), labels or speech and text bubbles. Furthermore, corresponding words or phrases from the two text areas can be cross-linked and there is also the option of linking words to a bilingual glossary. The Reader (Figure 15) allows children to browse the pages of a story book, click on links that have been added to the pictures, look up words in the glossary and experiment with correspondences between the two languages. There is also a mode (named "Read to me") where the interface automatically reads the story and goes through all its pages, without any user intervention.

Isis (Kim, 1995) is an interactive multimedia document building tool for children, developed as part of the Home Health-Care Prototype System for children with leukemia, which IBM has developed jointly with New England Medical Center. In Isis, objects such as videos, photos, drawings, texts, sounds and cartoons were treated as electronic building blocks for creating "elastic" stories (Kim and Song, 1995). A story is deemed as "elastic" when its length can vary according to time constraints and preferences set on its contents. To achieve this, a spring metaphor is used to represent all multimedia objects, which are augmented with three parameters: a minimum, a maximum, and an optimum length. Children can create multimedia stories by arranging multimedia objects on the screen and then setting relationships between them, such as "start together", "end together", "occur together" and "meet". When the creation phase is over, the system, using time scheduling algorithms, tries to build a single multimedia story that best meet all the requirements and constrains set.

(b) Video annotation applications

Although these research efforts are not targeted to children, they can provide some interesting ideas for the design of a video annotation interface.
CueVideo by Ponceleon et al. (1998) is a video retrieval system that supports video annotation with image, text, speech etc. The annotation process is performed through a typical GUI form-based interface that can also be used through voice recognition.

Zodiac (Chiueh et al., 1998) is an interactive video authoring system, which provides users an innovative branching history model of edit operations in order to organize the authoring process, and navigate among versions of authored documents. Zodiac also features a very interesting video annotation capability. Instead of annotating entire frames, users can associate annotations in the form of text, image, audio, or video, to moving objects in a video sequence.

(c) Multi-user single display environments

An early example of such an environment is the Multi-Device, Multi-User, Multi-Editor (MMM) developed at Xerox PARC (Bier & Freeman, 1991). MMM supports through a set of user interface techniques the concurrent use of multiple text and drawing editors by multiple users, each one operating a separate mouse device, but all sharing a common display and keyboard.

Inkpen et al. (1995) studied the effects of children using multiple mice to play a game on a single computer. Two control passing protocols were tested: *give*, where one child passes control to the other, and *take*, where a child can directly get the control from the other. The results of the study suggest that the ability of both children to control the game increases the performance of a pair of children playing on a shared computer. The type of protocols used to transfer control have very different results based on gender.

Bricker, Baker and Tanimoto (1997) have developed a set of interface objects that facilitated close collaboration among the users of a multimedia computer program. A sample application, named CoImage, has been developed, supporting multi-user

activities such as collaborative image warping, a jigsaw puzzle, a drawing program and skill game reminiscent of Etch-A-Sketch. Each user has a mouse and a corresponding colored cursor on the screen.

Stewart, Bederson & Druin (1999) report on the creation of a collaborative drawing application for (two) children, as well as on the results of an evaluation conducted to test the success of their design decisions. The application was developed using a "local tool" metaphor, according to which tools are represented as separate icons augmented with user data. Once the user has chosen a tool, he/she is the only person who can configure the tool to behave differently. Overall, the presented evaluation results show that children prefer the collaborative over the single-user use of the application.

2.5.4.2. Commercial software

In addition to the research efforts presented above, commercial software products exist that are targeted to roughly the same user groups. These are mainly related to multimedia authoring and storyboarding. These products include:

- Kid Pix Studio Deluxe by Brøderbund Software[26], California
- Magic Theatre by Instinct Corporation[27], Silicon Valley
- Stanley's Sticker Stories by Edmark Corporation[28]
- Kid Works Deluxe by Davidson[29]
- The Multimedia Workshop by Davidson[30]

An overview of the basic functionality and characteristics of these products is provided in Table 1 (adapted from Bouras et al., 2000).

[26] http://www.broderbund.com/
[27] http://www.magictheatre.com/
[28] http://www.riverdeep.net/products/early_learning/stanleys_ss.jhtml
[29] http://www.davd.com
[30] http://www.davd.com

Feature	Kid Pix Studio Deluxe	Magic Theatre	Stanley's Sticker Stories	Kid Works Deluxe	The Multimedia Workshop
Authoring platform	x	x	x	x	x
Playback platform	x	x	x	x	x
Age range	3-12	pre-readers	3-7	4-9	8 and up
Create video	x	x	x	x	x
Insert video	x				x
Create sound	x	x	x	x	x
Insert sound		x	x	x	x
Text-to-speech	x			x	x
Verbal help		x	x	x	
Icon-based UI			x	x	
Automatic saving and naming of files		x			
Spelling			x	x	
Wizard				x	

Table 1: A collection of features from commercial diary composing tools for younger children (adapted from Bouras et al., 2000)

The main characteristics of the aforementioned commercial products and their user interfaces are provided below.

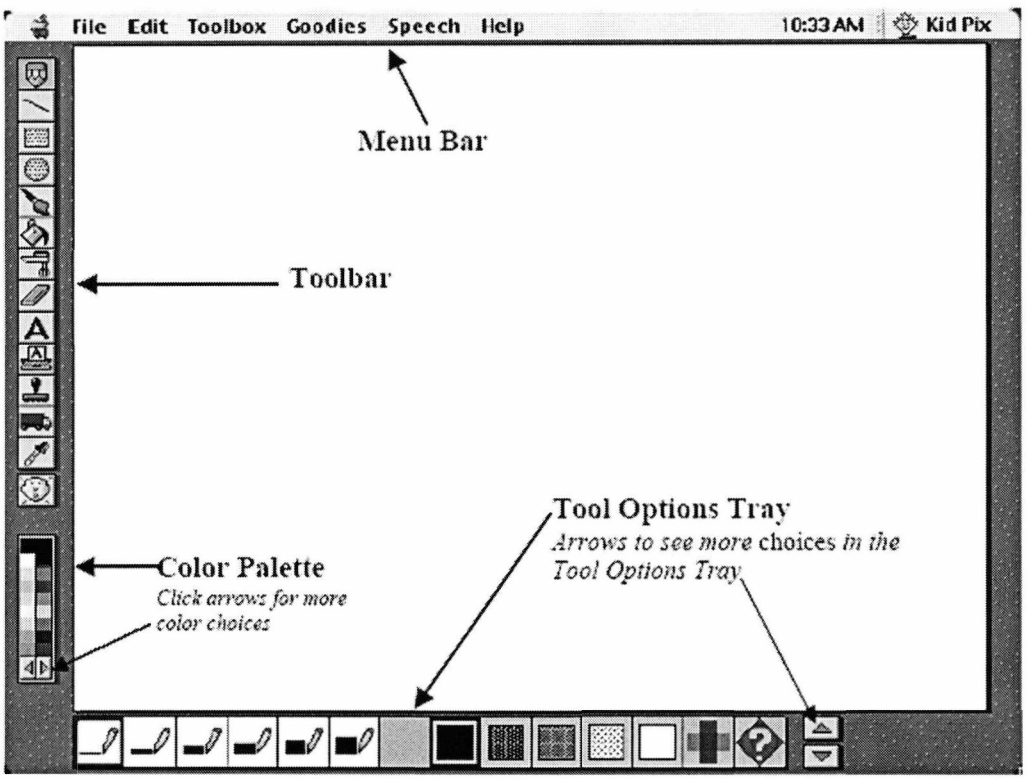


Figure 16: User interface of the Kid Pix Studio Deluxe by Brøderbund Software

The basic user interface of *Kid Pix* (Figure 16) consists of a drawing area with several pull-down menus at the top of the window and fourteen drawing tools and a color palette on the left. To the bottom of the screen several alternative options for the selected tool are available. Each tool has a corresponding sound, and the program has a feature that allows the creation of additional sounds, so that users can input cues or instructions. Beyond typical tools found in any drawing application, such as shapes, brushes, text areas and fill-in buckets, Kid Pix also features some utilities oriented to children, such as an "Electric Mixer" that adds special effects to pictures, a "Rubber stamp" which is used to add pre-made drawings on the screen, a "Talking Alphabet Stamps" to put letters, numbers, and punctuation marks on the screen, and a "Wacky Brush" that produces squiggles, dribbles, and other design elements. There is also a "small kids mode" for younger users, disallowing the possibility of accidentally locking up or erasing an item, or quitting out of the program.

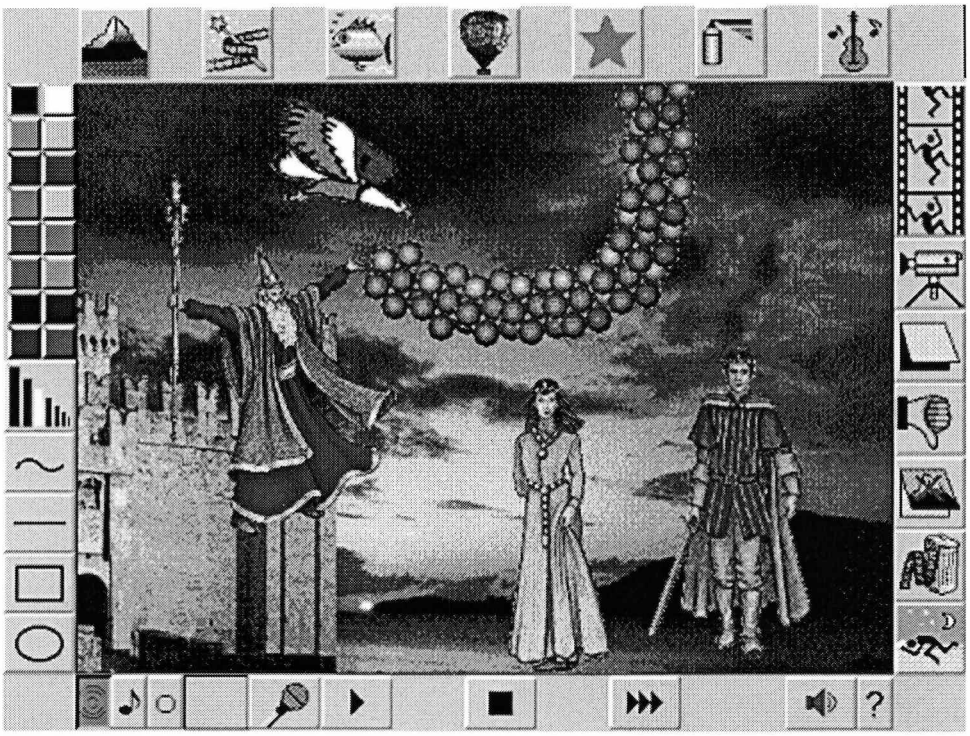


Figure 17: User interface of the Magic Theatre by Instinct Corporation

The *Magic Theatre* (Figure 17) is an animated movie making program for children aged 4 and older. Children can drag and drop scenery, characters, animations, objects, sound effects and music from several available libraries. Additionally, children can use paint tools to make their own graphics. In the retail box of the commercial program a microphone is included which can be used to record and add to the movies narration, songs and sound effects. Movies can consist of multiple scenes and are named and saved automatically. The sound and graphics coordinate automatically, avoiding synchronization problems.

Using *Stanley's Sticker Stories* (Figure 18) children can make their own animated storybooks selecting among over 300 animated character, letter, number and object stickers. Stickers can be customized (e.g., made larger or smaller, associated to sounds, or animated) and they get bigger as they're dragged to the front of the scene, while they shrink as they're moved into the background. A Sticker Spelling Book is available through which children can see and hear the spelling of a sticker. Sound can be recorded to make characters talk, sing, and laugh. Help is provided orally, so that the program can be also used by non-readers. Formatted text can also be added to the

stories. An interesting feature is that, by clicking on Stanley when making a story, he offers some interesting ideas.

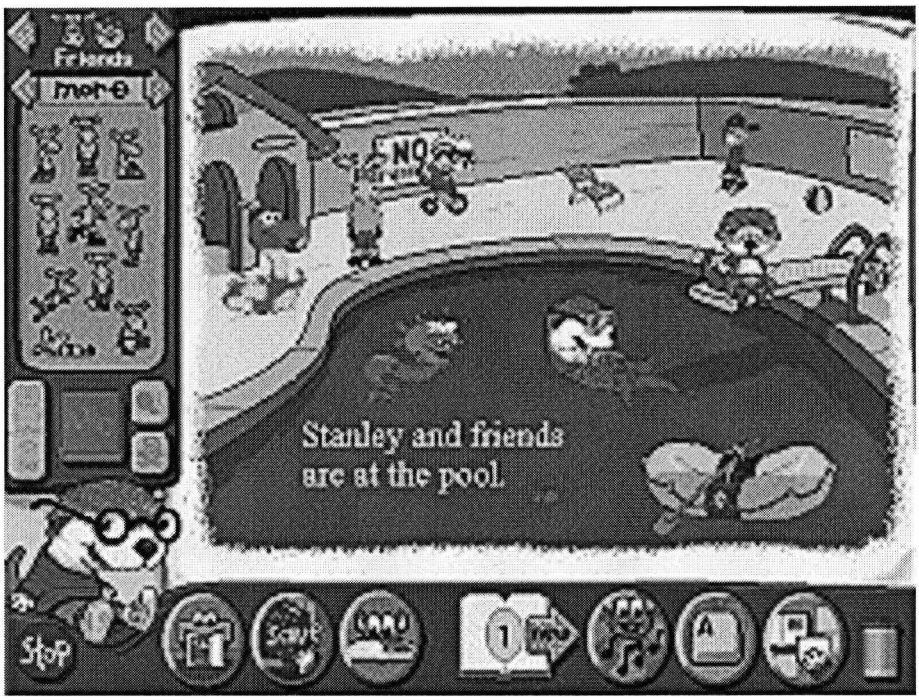


Figure 18: Stanley's Sticker Stories by Edmark Corporation

Kid Words Deluxe (Figure 19) combines a word processor and paint program to create multimedia stories. It includes drawing tools, animated stamps, sound effects, and a special feature that lets the computer read words and stories. Children can switch the figures back and forth from words to pictures. There are also "story starters," templates to help children begin writing that are common features in children's word processing software. The resulting creations can be shared through the Internet.

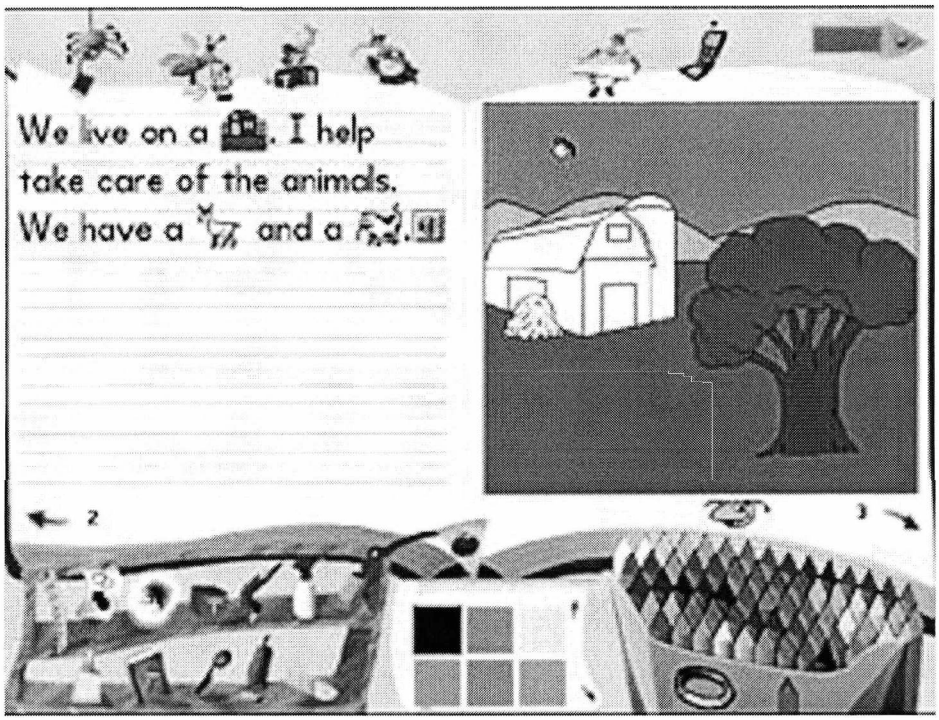


Figure 19: Kid Works Deluxe by Davidson

The *Multimedia Workshop* consists of three modules: (a) the writing workshop, for creating paper presentations; (b) the video workshop, for making animated scenes and movies; and (c) the paint workshop, for creating or modifying graphics that can be used in either of the other two modules. Additionally, a multimedia library is included as a separate CD-ROM which contains over 200 photographs, 300 pieces of clipart, 75 movie clips, 200 sound effects and 35 music clips. The video workshop comprises a scene maker and sequencer. In the scene maker, individual scenes are created which can then be linked through the sequencer. A scene can contain text, photos, clip art or video. In the sequencer a movie can be put together using two tracks: a scene and a sound track. Both tracks can be adjusted to synchronize the pictures and sound.

Chapter III

NAVIGATION IN VIRTUAL ENVIRONMENTS

3.1. INTRODUCTION

This chapter describes the work conducted in the context of this thesis for the creation of a novel, intuitive metaphor for a non-typical domain and the process followed for defining and refining it, as well as testing its value and validity.

In the real world, every living organism constantly leaves traces of its existence and its 'interaction' with the physical environment: deer leave their paw marks on the soft forest soil, dolphins carve foam traces on the surface of the sea, flies leave annoying black spots on windows, and young children imprint their dirty handprints on the freshly painted house walls.

Since the early years of their presence on earth, humans observed this inherent property of the environment and learned to use it in various ways in order to make their lives easier. For example, they learned to recognize the paw prints of animals to track down their prey or to avoid ferocious creatures, they used footprints to explore unknown territories or find their colleagues in search and rescue operations (Kearney, 1999), they examined fossils to study human history and evolution (Tattersall, 1995), and they revealed and analysed fingerprints to solve crimes (Beavan, 2001).

In contrast to real environments, Virtual Environments (VEs) do not allow their 'inhabitants' to leave any trace behind, thus suffering from an '*extreme cleanness syndrome*'. Walk into your house after leaving your children alone for the weekend and you can instantly realize that a wild party took place while you were away. Walk into a virtual chat room seconds after a meeting of two hundred people has finished and it will be exactly as if no one has ever been there before.

Inspired from these observations, this chapter introduces the concept of Virtual Prints (ViPs) (Grammenos et al., 2002) as the digital, interactive, counterparts of real-life tracks. The basic idea is that while a user is moving in a VE, Virtual Footprints (FootViPs) are left behind, whereas each time an interaction with an object occurs, the

user's Virtual Handprints[31] (HandViPs) are 'imprinted' on it. Both FootViPs and HandViPs can be time-sensitive and gradually fade, as real - or virtual - time goes by. Virtual Markers (MarkerViPs[32]) are permanent marks coupled with user-defined data (e.g., a textual or audio message) which can be attached to the environment, or to any virtual object, and can act as personal landmarks, annotations, or "anchors". Virtual Prints address the challenge of creating a novel, intuitive, interaction metaphor for supporting navigation in the interaction domain of virtual environments.

The rest of the chapter is structured as follows: section 2.2 provides an overview of the concept, describing the distinctive properties and characteristics of each type of ViPs, while section 2.3 provides an account of how ViPs can be instantiated in a virtual environment through a related software mechanism, and of how end-users can interact with them. Section 2.4 illustrates challenges that may potentially arise when putting ViPs to real use, along with suggestions and ways for overcoming such challenges. Section 2.5 provides a comprehensive overview of possible uses of ViPs beyond navigation, orientation and wayfinding. Section 2.6 describes the process that was followed for making the transition from early concept formation to a full-functioning software implementation, including the exploratory studies which were conducted, as well as several inspections and formal experiments with both experts and potential end-users. Section 2.7 summarises the lesson learnt and consolidated experience stemming from experimentation with the concept and implementation of ViPs. Finally, section 2.8 concludes the chapter and offers an insight into future work.

3.1.1. Deployment Context

The concept and software implementation of ViPs was systematically studied and elaborated in the context of the "VIEW of the Future" (IST-2000-26089) project, funded by the European Commission in the framework of the Information Society Technologies (IST) Programme, contributing to one of the project's goals which was to develop new interaction, navigation and manipulation concepts for VE-

[31] Virtual Handprints were originally named Virtual Fingerprints, but the conducted studies revealed that the concept of Handprints is far better both in terms of usability and intuitiveness (e.g., fingerprints are too small to be noticed and to interact with).
[32] Virtual Markers were originally termed Virtual Fossils, but they were renamed as a result of user testing.

applications. During the project, a comprehensive "User Requirements Document" was compiled (Basso et al., 2002), presenting typical usage scenarios and related user needs of real VE applications used by the industrial partners. These needs were analysed and grouped, resulting in a list of ninety-three (93) distinct common requirements, which were used to drive all the project's development activities. In this context, the ViPs mechanism was designed to fully, or partially, meet thirty two (32) of the aforementioned requirements (mostly related to navigation and interaction), thus having the potential to support a substantial number of related application tasks and scenarios.

3.2. CONCEPT OVERVIEW

In correspondence to real world marks, ViPs can be personalized (Figure 20). Therefore, they provide users with a sense of existence and individuality, and help participants of multi-user VEs acquire awareness of other users and activities. ViPs can be represented in various ways, depending on the characteristics of the application and on the user's requirements and preferences.

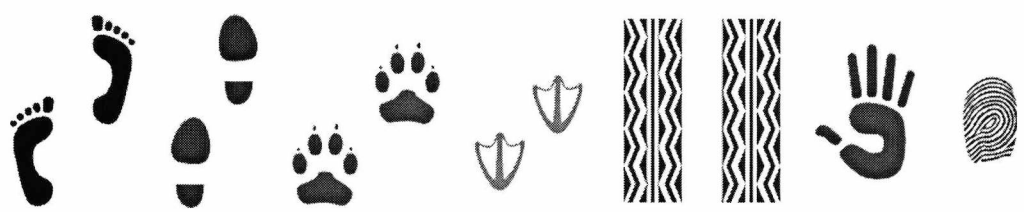


Figure 20: Examples of alternative personalized ViPs

ViPs can be time-sensitive (for example they can fade or change shape as time goes by, Figure 21), thus avoiding ViPs pollution (see section 3.4.1) and helping users distinguish older from new(er) ones, and keep track of time. Related ViPs can (upon user request) be connected through connecting lines (see section 3.4.2).

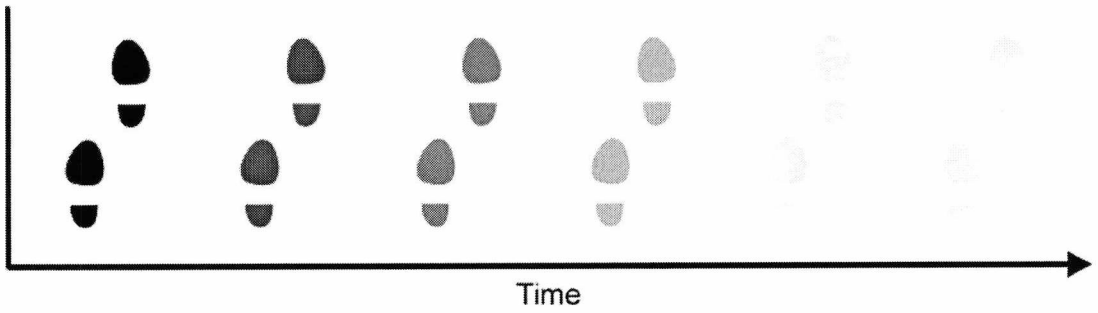


Figure 21: Example of a time-sensitive FootViP

3.2.1. Virtual Footprints (FootViPs)

While a user is moving within a virtual world, Virtual Footprints (FootViPs) are left behind (examples are illustrated in Figure 22 and Figure 26). A FootViP can store and provide:

- spatial information, i.e., the position and orientation of the user in the virtual world;
- chronological information, i.e., time and date of creation, last access or modification;
- personal information, e.g., owner name, e-mail, current position, status;
- information about related HandViPs or MarkerViPs.

FootViPs can be released anytime, either upon user demand or automatically at specific time or space intervals, as well as each time a HandViP or a MarkerViP is released. All FootViPs belonging to a single user that share common spatial (e.g., are in a specific virtual area), chronological (e.g., were created during a specific time period), or semantic (e.g., were created while performing a specific task) characteristics can be grouped and visualized as *Virtual Paths*.

3.2.2. Virtual Handprints (HandViPs)

Every time a user interacts with a virtual object, a HandViP is 'imprinted' on it (see Figure 27). At the same time, a FootViP, to which the HandViP is associated, is

automatically released to record the position and orientation of the user at the moment the interaction took place. For example, if a pointing device is used, the user's HandViPs are 'imprinted' (i.e., visualized) at the pointed position, e.g., as a cube or, more realistically, as a model of a Handprint. Non-visual HandViPs (e.g., generated from speech-based interaction) can also be released. Similarly to FootViPs, HandViPs can store and provide:

- spatial information;
- personal information;
- information about the interactive component on which they are released, e.g., name or the type of interactive object;
- descriptive information about the performed user action, e.g., "left-click on the interactive device";
- any other information about the induced effect, such as semantic information, e.g., "the door opened".

3.2.3. Virtual Markers (MarkerViPs)

MarkerViPs are permanent marks coupled with user-defined data that can be left anywhere within the virtual world or attached to virtual objects. Just like HandViPs, whenever a MarkerViP is released, a FootViP is automatically created to record the position and orientation of the user. MarkerViPs can store:

- spatial information;
- personal information;
- a message in any digital form, such as text, audio, or multimedia, e.g., instructions of use for an interactive component;

A MarkerViP can be associated with any other ViP (i.e., acting like a shortcut), allowing quick transportation from one location to another within the virtual world. MarkerViPs can also be used as a context-sensitive help system. For example, a number of MarkerViPs can be attached to specific components of a VE, providing related descriptions and guidance for inexperienced users. Furthermore, MarkerViPs

can act as bookmarks to points of interest. MarkerViPs can be both visual and non-visual, and can be represented by any object, such as for example pins, yellow stickers, road signs, wall signs / posters, or computer characters.

3.3. INTERACTING WITH VIPS

In a VE, the ViPs concept is instantiated and supported through a software mechanism (the *ViPs mechanism*) that on the one hand implements all the required functionality for generating, tracking, configuring and handling ViPs, and, on the other hand, provides a user interface for interacting with them. This mechanism allows the user (e.g., through a visual or a voice menu) to release a new ViP, activate or deactivate the automatic recording of ViPs, search for specific ViPs, and modify the ViPs generation and display configuration.

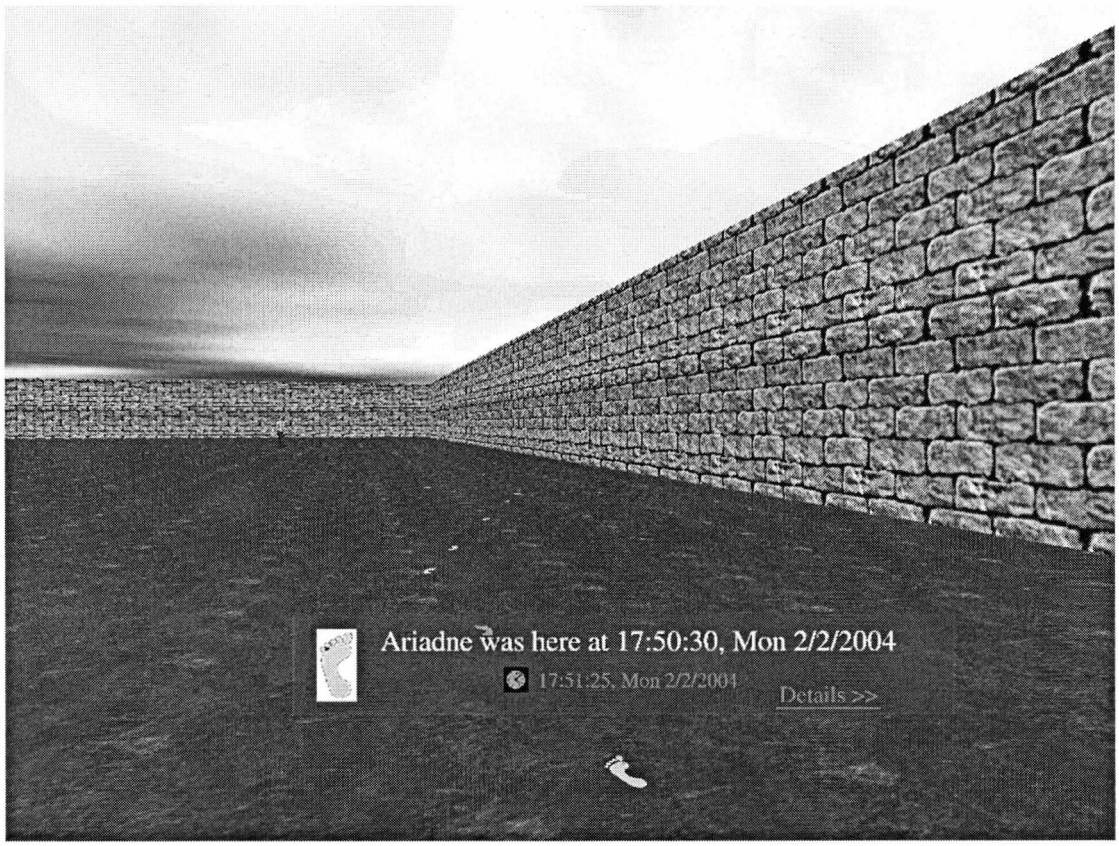


Figure 22: An example of accessing information related to a Virtual Footprint

Each ViP is associated with miscellaneous data, such as type, owner, creation time and date. This information can be presented to the user in multiple ways, depending

on the application and user requirements. For example, it can be visualized through an information sheet (such as the one illustrated in Figure 22) triggered by a 'virtual pointing device', e.g., the mouse cursor, or a virtual hand.

When a ViP is selected, the supporting mechanism offers the following options:

a. Set display options, e.g., modify the way ViPs are depicted; hide / display specific type(s) of ViPs; visualize connecting lines between related ViPs; hide the VE and see only the ViPs; scale the ViPs up or down (i.e., an 'inflate' / 'deflate' effect). Figure 23 depicts a visual user interface as an indicative example for providing these options.

b. Perform ViPs-based navigation, e.g., manually or automatically follow (forward and backward) existing tracks, or hyper-jump to the first / last of them. Figure 24 depicts a visual user interface as an indicative example for providing these options.

c. Turn on / off the connecting lines related to the selected ViP.

d. Find related ViPs (e.g., find the closest HandViP or MarkerViP that belong to the same owner). Figure 25 depicts a visual user interface as an indicative example for providing these options.

e. In case the ViP belongs to the user, change ViP creation options.

Figure 26 and Figure 27 illustrate examples of 3D menus for interacting with FootViPs and HandViPs respectively.

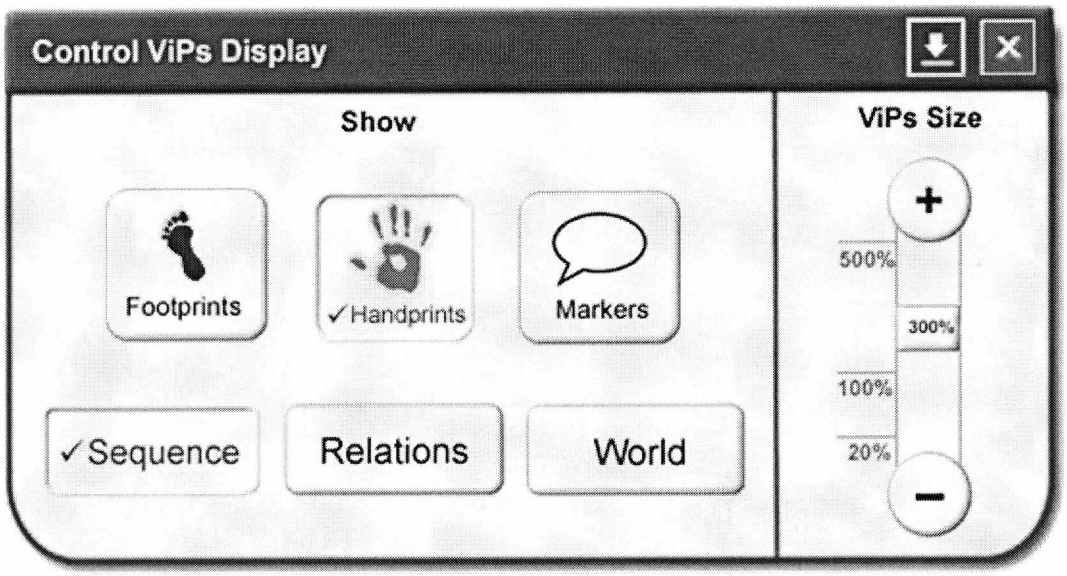


Figure 23: Example of a visual user interface for controlling ViPs display options

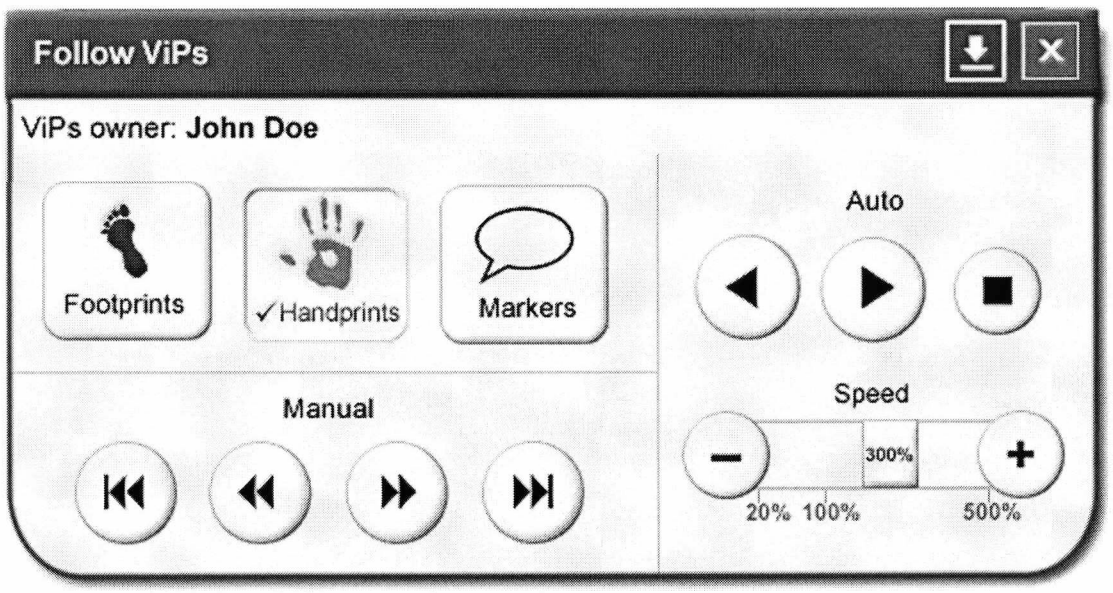


Figure 24: Example of a visual user interface for performing ViPs-based navigation

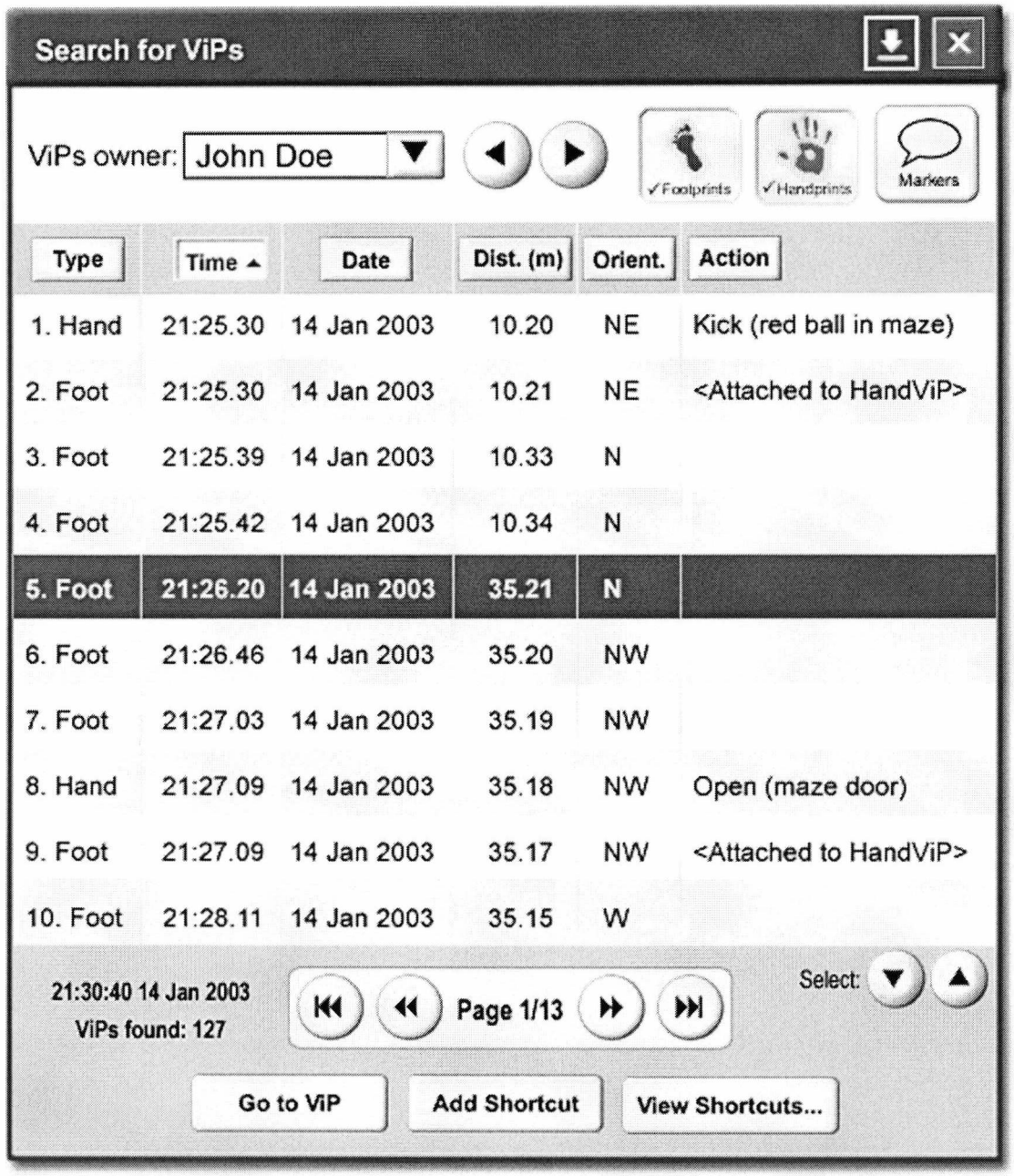


Figure 25: Example of a visual user interface for searching for ViPs

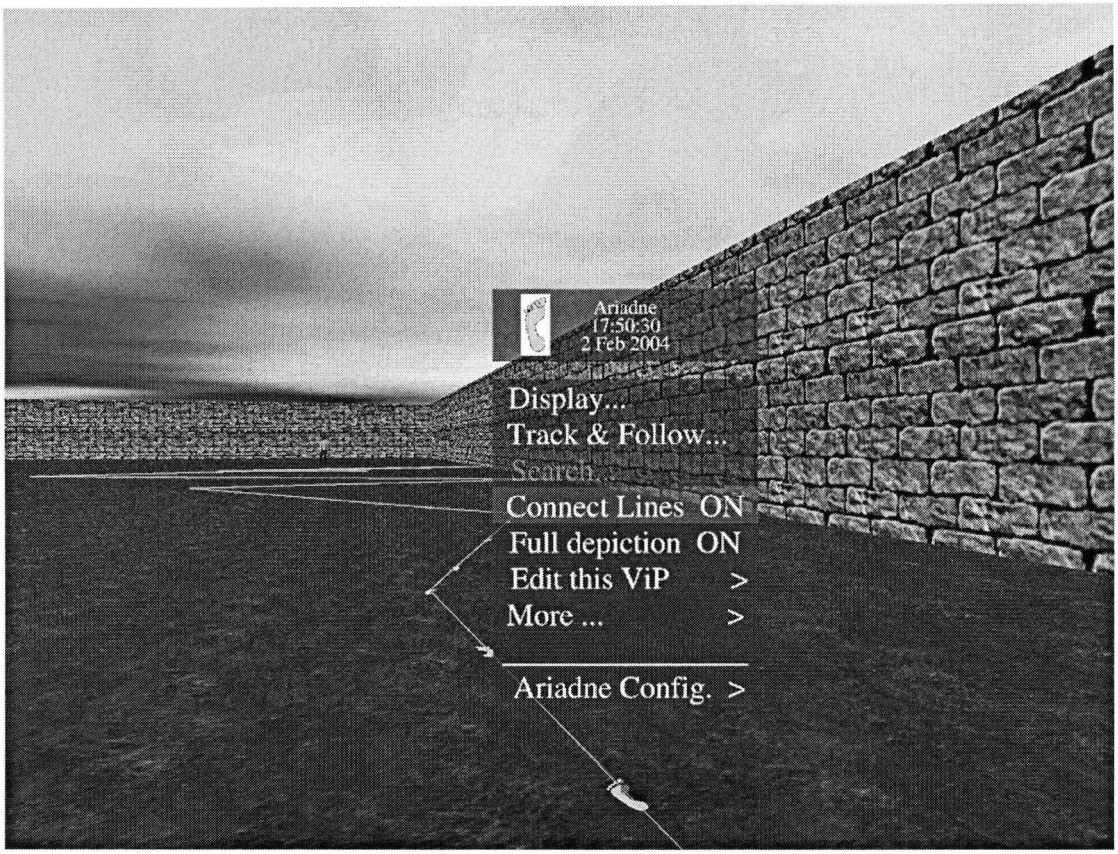


Figure 26: Example of interaction with a Virtual Footprint

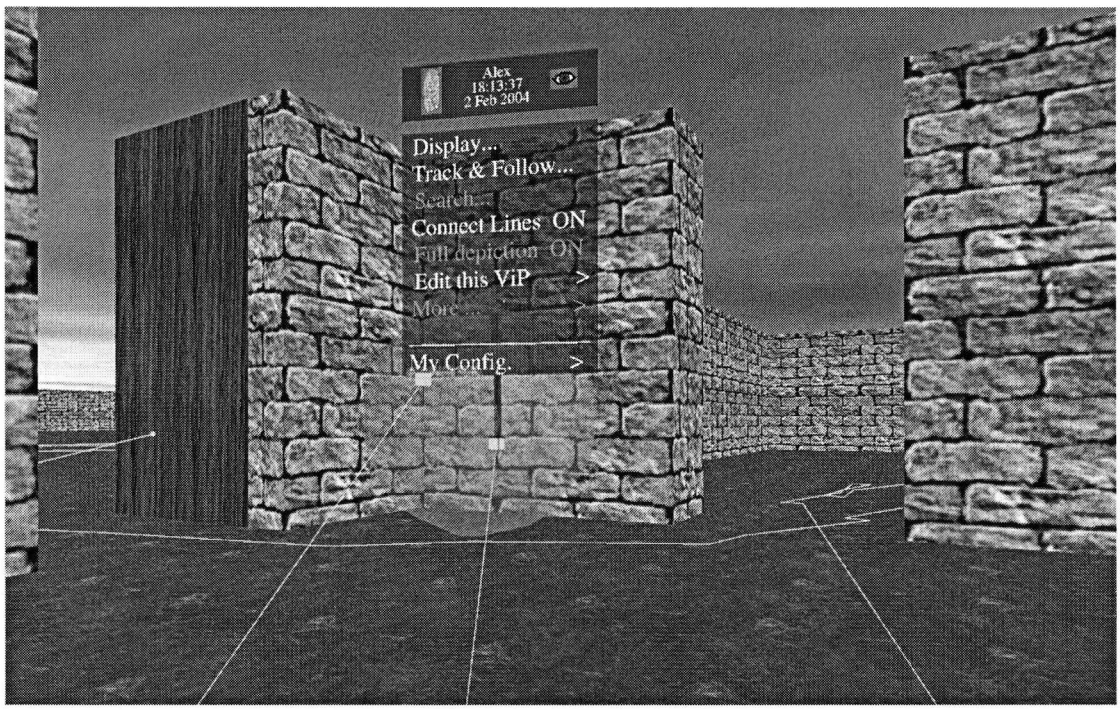


Figure 27: Example of interaction with Virtual Handprints (on the red ball)

3.4. DESIGN AND IMPLEMENTATION ISSUES

In the context of this thesis, the development and integration of a ViPs mechanism in an interactive virtual environment has revealed several issues that needed to be addressed.

3.4.1. ViPs "pollution"

According to Darken & Peterson (2002), as navigation proceeds, the environment can become cluttered with footprints. This, of course, becomes far worse in a multi-user environment. The ViPs mechanism supports alternative methods for overcoming this problem:

(a) A filtering mechanism allows the users to select which ViPs they wish to see, according to several alternative parameters such as their type (i.e., FootViP, HandViP, MarkerViP), creator (e.g., user's own ViPs, all ViPs, ViPs belonging to specific users or user groups), creation time (e.g., last X minutes, from time A to time B), proximity (e.g., the Y closest ones) and number (e.g., only the Z most recent ones).

(b) ViPs can be time-sensitive and thus disappear after a specific time period (see Figure 21). Of course this can create new problems. For example the user will be no longer able to tell if a place has been visited after the ViPs have disappeared. This can be dealt with through a filtering mechanism that allows ViPs that have 'disappeared' to be viewed.

(c) ViPs can be viewed by using a simplified representation. This option is quite useful for places that are overcrowded with ViPs (especially if these have different colours, shapes and sizes). They can be all represented, for example, as small dots or cones, offering to the user a clearer view of the environment in combination with useful ViPs-related information (e.g., amount of ViPs present, areas of interest, paths). This option requires considerably less rendering resources and thus can improve the system's performance.

(d) The number of visible FootViPs can be reduced by using a "smart" elimination algorithm. For example, intermediate ViPs in a more or less straight path can be eliminated, unless a HandViP of MarkerViP is attached to them.

3.4.2. ViPs continuity and relation

The problem of *continuity*, also reported by Darken & Peterson (2002), arises when the user crosses paths while leaving footprints, since in this case it can become difficult to disambiguate which footprints belong to the same trail. In the proposed approach, this problem can be solved by displaying a connecting line between all FootViPs belonging to the same path (see Figure 28).

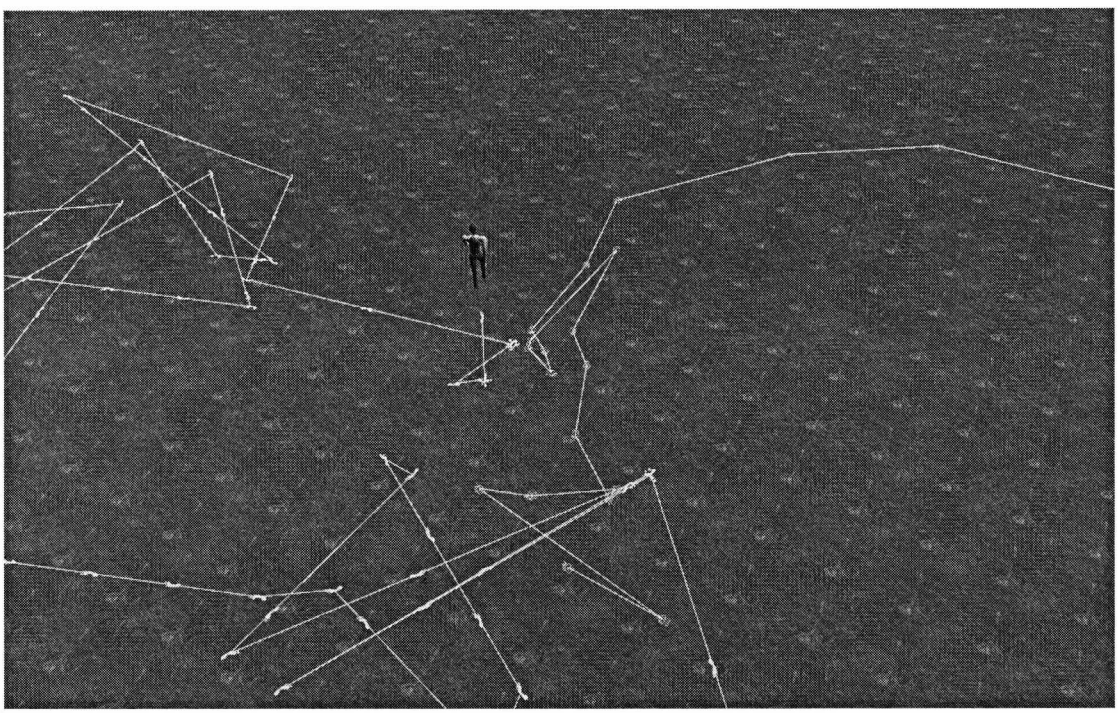


Figure 28: Connecting ViPs to visualize the user's path

Another problem that may arise, due to the fact that users can interact with objects located at different distances, concerns *relation*. This means that a FootViP released upon user interaction (representing the user's position and orientation at that time) can be positioned considerably far from the corresponding HandViP (see section 3.2.2), or even out of sight. Thus, for a user coming across one of them (e.g., the HandViP), it may be difficult to locate the other (i.e., the FootViP) or infer the relation between the two. The same may also occur between a MarkerViP and its corresponding FootViP (see section 3.2.3). This problem is also addressed through the use of connecting lines (preferably of different style than the lines used for continuity).

3.4.3. Overlapping ViPs

When two or more adjacent ViPs overlap, apart from the aforementioned problem of continuity, visual and interaction ambiguity problems may also occur. Visual clarity may be lost when, for example, two or more ViPs of similar colour or shape overlap, while interaction ambiguity occurs when the pointing "device" is concurrently intersecting more than one ViP and it is not clear which one the user wishes to interact with. This problem is usually solved through scaling ViPs up or down (i.e., through an 'inflate' / 'deflate' effect).

3.4.4. Privacy and protection of personal data

Since ViPs can be considered as a mechanism that collects, and thus can also potentially expose, personal information, adequate policies should be adopted to protect the privacy of the participants of a VE, in accordance to established relevant guidelines and principles, such as those included in the European Community Directive on Data Protection (95/46/EC)[33] and the Privacy Guidelines by the Electronic Privacy Information Center[34].

To this purpose, the ViPs mechanism collects only data that are relevant and legitimate for the purposes of its proper functioning. Furthermore, it allows each user to:

(a) View at any time the personal data that have been (and are) recorded and destroy any or all of them.

(b) Define what information will be made available to other users.

(c) Turn the recording mechanism on or off at any time, as well as select to manually place ViPs in the VE.

(d) Use the system anonymously.

(e) Grant or restrict access to own ViPs and related information.

[33] http://www.acs.ohio-state.edu/units/law/swire1/psecind.htm
[34] http://www.epic.org/privacy/internet/EPIC_NII_privacy.txt

3.5. USES OF VIPS

ViPs were originally conceived as a means to support navigation in a VE. For example, leaving trails on a surface or in space allows the user to quickly refer back to them and easily reorient when the continuity of the current orientation tracking has been interrupted. Furthermore, ViPs, and especially MarkerViPs, can act as personal landmarks, thus allowing the user to find the way back to previously visited locations, or follow the tracks of other users in order to locate or follow them to a desired location. Additionally, through ViPs-based navigation facilities (e.g., Figure 24), users can easily move in a VE following predefined paths, "jumping" to specific places / shortcuts, returning to a specific course, etc., or employ alternative, 'non-traditional' navigation techniques, such as navigation by query (van Ballegooij & Eliëns, 2001).
Beyond the aforementioned uses, ViPs can also be employed for several other purposes, such as:

a) Locating other participants (e.g., friends, enemies, people the user wishes to meet or avoid) in multi-user and collaborative entertainment (i.e., gaming) environments (e.g., "multi-user dungeon – MUDs" games, chat worlds).

b) Supporting *social navigation* (Munro et al., 1999), a concept based on the fact that when people are looking for information (e.g., directions, recommendations) they usually turn to other people rather than using formalized information artefacts. For example, participants of a multi-user VE can easily identify popular places or options through the number of HandViPs or FootViPs on them.

c) Creating tutorial sessions. A tutor can leave behind a number of ViPs that the 'students' may follow, for example, to learn a specific procedure in a "step by step" procedure.

d) Developing virtual tours. Visitors can tour virtual museums, exhibitions, stores, etc., by following the ViPs of virtual guides.

e) Visualizing and tracking the path of moving objects, e.g., observing (or even predicting) the paths of friendly and enemy units in military Command & Control Centre applications or visualizing the orbit of planets and other celestial objects in virtual planetariums.

f) Facilitating content annotation and marking / identifying (non) visited areas.

g) Providing context-sensitive help with the use of MarkerViPs.

h) Studying user navigation (e.g., Ieronutti et al., 2004) and interaction in 3D environments and supporting user-based evaluation of VEs (e.g., path analysis; replaying user actions; providing statistics related to distance travelled or least / most visited areas; finding unused or underused interactive elements, etc.).

i) Performing measurements related to distance and time.

j) Providing functions and concepts which are popular in conventional 2D user interfaces, such as shortcuts, bookmarks, undo / redo functions, versioning and collaborative review, marking / identifying (non) visited content, content annotation and highlighting.

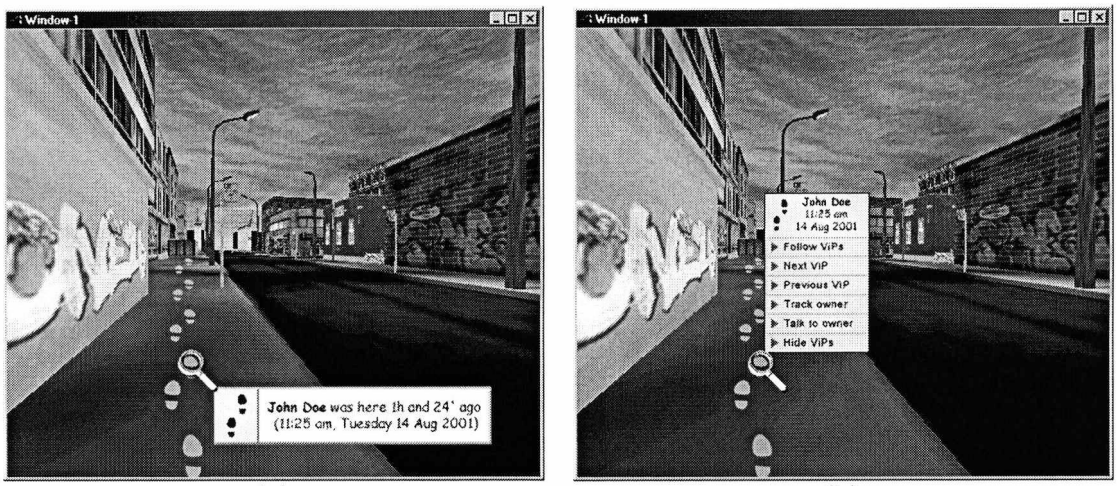
3.6. FROM CONCEPT FORMATION TO SOFTWARE IMPLEMENTATION

A user-centred approach was followed for the development of a mechanism instantiating the ViPs concept. A co-operative design and evaluation approach was followed, which included the participation of interaction designers, usability experts, developers and potential end-users. The development of the ViPs software mechanism included a series of steps that are discussed in the subsequent sections.

3.6.1. Concept formation and prototyping

The suggested concept, as well as the related work, were studied and analysed, resulting in preliminary functional and interaction requirements for the ViPs mechanism software. Based on this description, the 'look & feel' of the system was initially developed as a 'pencil and paper' prototype. These sketches were subsequently converted to a digital (non-interactive) prototype using Sense8's WorldToolkit VR software for the creation of elaborate pictures of a 3D world, and subsequently Adobe Photoshop for overlaying interaction objects such as menus and pop-ups on the Virtual World (two examples of the concept development prototype are presented in Figure 29a and Figure 29b). Then, the digital prototype was presented to interaction designers, usability experts and potential end-users to obtain their preliminary opinion on the understandability, utility and usability of the concept and

its software instantiation. This was accomplished through semi-structured interviews based on a small set of relevant key points. Furthermore, the prototype was discussed with VE developers to assess its technical feasibility.



(a) (b)

Figure 29: Two examples of the ViPs concept development prototype

3.6.2. First interactive prototype & exploratory studies

Based on the outcomes of the aforementioned preliminary study, a prototype VE equipped with a simplified ViPs mechanism has been implemented, in order to study ViPs in practice, as well as their envisaged functionality and properties. The software used to develop the prototype was Maverik for Linux, a publicly available Virtual Reality toolkit developed by the University of Manchester (for more details, see http://aig.cs.man.ac.uk/maverik/).

Using this prototype, two separate studies of explorative nature were conducted, with the aim to collect qualitative data, such as comments, ideas and opinions about the overall concept of ViPs and its potential usefulness and shortcomings, as well as about the adopted design and implementation approach of the ViPs mechanism. These data were very helpful not only towards validating in practice the usefulness of the concept, but also for fixing problems and improving the overall usability of both the design and the implementation before proceeding to further formal user testing.

The pilot experiments were performed using a non-immersive version of the system that was projected on a 17'' monitor. The interaction devices used included a standard keyboard and a mouse for user movement (the users could select their preferred device), while the mouse was also used for interacting with virtual objects and ViPs. Users were able to move forward, backward and turn left and right, but not up and down (i.e., fly above the virtual 'floor'). For the needs of the experiments, an outdoor maze-like virtual environment was constructed, as well as some simple interactive 3D objects.

A study was performed with two expert interaction designers / usability experts with the aim to present the concept and the design of ViPs and obtain comments about their value and usability, but also useful ideas for improving the design and identify potential usability problems. The methods used included *Heuristic Evaluation* (Nielsen, 1994) and *Cognitive Walkthrough* (Wharton et al., 1994).

A second study included six participants, potential end-users of the system (4 males and 2 females), all experienced computer users, but with different level of expertise in the use of VEs. Two of the users had experience in using immersive VEs, such as a CAVE or HMD-based system. Two users were familiar with VRML-based worlds, 3D games and 3D chat applications. The remaining two users had no previous experience in using a VE. The method used was *thinking aloud* (Nielsen, 1993), where participants were allowed to express their thoughts, comments and feelings at any time during the test and also interact with the two observers that were present. The role of the observers was to prompt the users for comments but also for alternative ways of performing the task. Due to the qualitative nature of the study, user-observer interaction was highly encouraged, since user performance was not traced or evaluated. The conversations were recorded using a digital audio recorder. To support the evaluation process, a list of indicative tasks was used, that prompted participants to explore the functionality and facilities offered by the ViPs mechanism. After using the system, a small debriefing session was held, where the overall impression of the user's interaction with the system was requested, as well as suggestions for improvement, modifications and personal preferences.

The overall impression of the participants with respect to the concept of ViPs and the pilot implementation was positive. All of them agreed that ViPs can be really useful in several cases, and that the overall metaphor that ViPs introduce in the context of moving and using a VE is very easy to grasp and utilise. Indisputably, the favourite part of the system was the option for personalized ViPs. All users spent considerable time browsing the relevant list of images to pick their favourite, and most of them changed it quite a few times while using the system, trying to find the one that they preferred or that they considered best-suited to their personal 'image'. Furthermore, all participants contributed with a number of ideas and suggestions about the instantiation of alternative images. This fact comes as no surprise, since the image of a user's ViPs is actually the user's representation in the virtual world and is often implicitly associated with character and personality traits.

The two interaction designers who participated in the experiment were mostly concerned with potential usability problems of the system. They contributed their views with respect to the problem of overlapping ViPs, and commented on potential interaction patterns, organization of the presented menus, and alternative parameters that could be employed for creating and viewing ViPs.

All participants who had previous experience with (any type of) VEs did not face any particular problems in using the system. Users of multi-user VEs expressed their concerns about privacy issues and the way these could be handled. Novice users mainly had difficulties in navigating and effectively using the input devices. Half of the users commented that there were too many suggested ViPs viewing and creation parameters, and that they would not be able to use them effectively without prior training.

An unexpected result of the tests was that one of the participants used ViPs in an artistic and playful way that was not foreseen when the system was designed, to draw patterns on the ground the way people make sketches in the sand.

Overall, the findings of the exploratory studies confirmed the initial hypothesis that ViPs can constitute a handy tool and a useful navigation aid, but also a feedback and history mechanism. In addition, the support that ViPs can provide for collaborative

environments and social navigation was considered significant and innovative. The studies also allowed to identify potential usability problems of the initial design and missing functionality, to collect user preferences that could help in refining and improving the concept and the resulting system, as well as to identify areas where further experimentation and testing was needed.

3.6.3. Second interactive prototype & sequential evaluation

The outcomes of the exploratory studies were used to develop a second version of the interactive prototype in which the usability and technical problems that were detected were corrected and further functionality was added (examples are illustrated in Figure 22, Figure 26, Figure 27, and Figure 28). For example, a Navigation and a Display Console (Mourouzis et al., 2003) were introduced. To evaluate this second prototype, the most appropriate process and methods had to be identified. In this direction, a potential problem was the lack of widely used and validated VE evaluation processes and tools. Also, due to the experimental nature of the developed system, it was necessary to acquire as much input (both qualitative and quantitative) as possible by the widest audience (experts and users). Thus, it was decided (Mourouzis et al., 2003) that the *sequential approach*, suggested by Gabbard et al. (1999), was the most suitable candidate, since it addresses both design and evaluation of VE user interfaces and combines several alternative techniques providing multiple complementary data. This approach (see Figure 30) involves a series of evaluation techniques that run in sequence, including *Cognitive Walkthrough* (Wharton et al., 1994), *Heuristic Evaluation* (Nielsen, 1994), and *Formative and Summative evaluation* (Scriven, 1967; Hix & Hartson, 1993). In general, a sequential evaluation uses both experts and users, produces both qualitative and quantitative results, and is application-centric.

Two different computer setups were used for the evaluation as, on the one hand, it was necessary to test the concept and its effect in both non-immersive and immersive VR systems, and, on the other hand, a non-immersive version was required to facilitate the Co-operative Evaluation of the system. The first setup was an immersive VR system using a stereo HMD (Virtual Research V8, with 640x480 true VGA LCDs), while the second was a typical desktop system using a 17'' monitor. Both versions were running on Dual Pentium III 1Ghz PCs with Linux Slackware 8.0 with

a Geforce-Ti4200 graphics card, using a conventional 2D mouse with three-buttons for navigation and interaction.

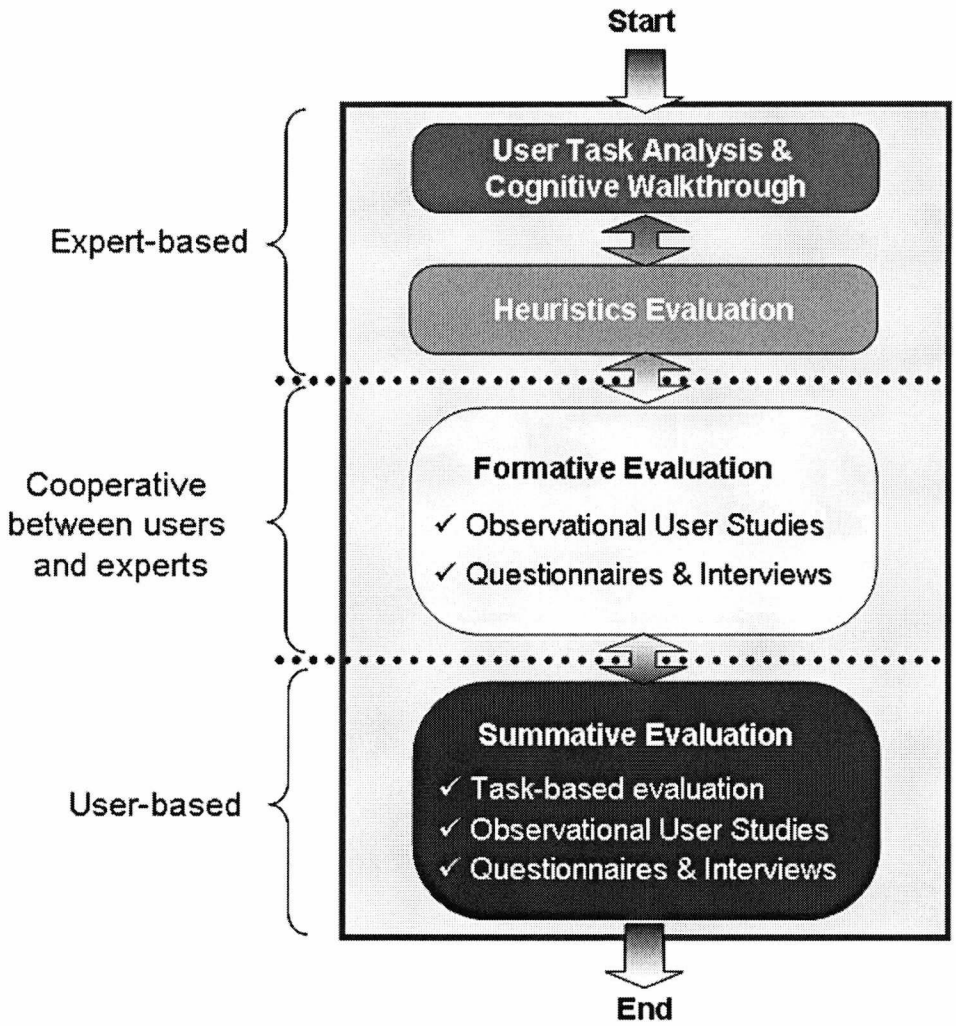


Figure 30: Approach followed for the sequential evaluation of the second ViPs prototype (adapted from Gabbard et al., 1999)

The virtual environment used for the study was a maze that included several corridors and rooms, populated with simple objects with which the user could interact (such as a door that opens when selected, a 3-item menu, and a ball). Although the test environment was a single-user system, in order to simulate and demonstrate the use of ViPs in multi-user environments, a number of computer-driven avatars (simulating 'other' users) were placed in it, leaving behind their own ViPs. In addition to this environment, a simple room (the "warming up room") equipped with interactive

objects was also constructed for user familiarization before the actual test with the system and its interaction facilities.

The evaluation procedure started with an expert-based inspection of the ViPs Prototype, using *VIEW-IT* (Tromp & Nichols, 2003), a method for assessing VEs in terms of utility and usability. The method's main instrument consists of forms that guide the assessor through the visual assessment of the interface by judging its compliance with a number of heuristics and principles suggested by the creators of the method. The evaluation team consisted of five assessors (3 males, 2 females) with a rich background in usability engineering and interaction design of 2D applications.

Next, a *Co-operative Evaluation* (Wright & Monk, 1992) was conducted with twenty people (8 females, 12 males) who were first asked to freely explore a desktop version of the system and then perform some simple tasks to help users familiarise themselves with the concept and functionality of ViPs. This evaluation step was conducted by two evaluators who were free to ask the participants questions at any time. Debriefing interviews and pre- and post-hoc questionnaires were also used to collect more specific information on the participants and their experiences with the system. In order to assess the usability of the system, a post-test questionnaire, focusing on ViPs, based on a usability questionnaire for VEs (Patel et al., 2003) was used. A few days later, all individuals who were present in the Co-operative Evaluation session were also asked to participate in a task-based usability evaluation using the HMD version of the system. In this final step, the users had to perform specific task scenarios, while user errors, actions in the VE, physical reactions and comments were recorded by means of digital audio, digital video, and screen video capture. Once more, several questionnaires were used to measure a number of parameters, such as *stress* (Gotts & Cox, 1988), simulator sickness (Kennedy et al, 1993) and *presence* (Witmer & Singer, 1998). Two structured task scenarios were employed to drive the tests:

- The first scenario was based on the famous ancient Greek myth of the Minotaur and the Labyrinth of Knossos. The scenario included the tracking of other participants, marking areas, wayfinding and backtracking, interacting with simple 3D objects. Users were free to decide whether to use ViPs or not. At the end of the

session participants were also "taught" how to fly over the maze and observe the imprinted paths.

- The second scenario mainly aimed at assessing ViPs as an orientation aid for VEs. Participants had to navigate in the virtual space and attempt to construct paths in the form of simple shapes, such as a square and an equilateral triangle. The participants were asked to "draw" each shape twice, once without and once using ViPs.

The three types of experiments conducted had complementary objectives, and overall facilitated: (a) the refinement of the VIPs concept; (b) the upgrading of the ViPs software mechanism behaviour and functionality; (c) the collection of suggestions for improvement of the user interface. In short, the results of these studies, which are reported in more details in (Mourouzis et al, 2004), were:

- *Expert-based review*: In this study, the level of presence of the system, as measured by the VIEW-IT tool, was found to be moderate. It was suggested that this could be improved by increasing the level of detail and the quality of the display (e.g., using an HMD of higher resolution), using visual and auditory feedback, and increasing and stabilising the system response rate. On the other hand, the level of VR-induced sickness was low. According to the assessors, this could possibly be lowered even more by keeping active user movement to a minimum (i.e., minimise the need of movement while interacting with the ViPs interface). Finally, the overall usability of the system was also found moderate, and a number of suggestions were produced for improving it, including providing adequate and consistent feedback, as well as cues to permissible interactions, introducing error messages and error management, and improving the quality of graphics. The evaluators expressed the opinion that the functionality and support provided by the ViPs mechanism has the potential to improve the usability of a VE, both in terms of ease of learning and user satisfaction, and in terms of efficiency, memorability, and error rates.
- *Co-operative evaluation*: This step produced massive evaluation data. Video recording was used to capture users' verbalisations about ViPs. More than 2100 verbalisations were recorded. About 220 of them were related to the usability of the system, out of which 100 were identified by both assessors as having high

value for improving the design and use of ViPs. To transform the produced qualitative data into quantitative, a variation of the method suggested by Marsh & Wright (1999) was employed, in which the users' verbalizations were assessed on the basis of: (a) their quantity, by counting all verbalizations and scoring a single utterance, statement, or sentence, or group of these relating to the same issue, as one; (b) their quality, which was judged by the evaluators. In general, most participants, orally and through their answers to relevant questionnaires, agreed that ViPs are easy to learn and use and that they can be a handy tool for navigation, wayfinding and annotation. An important comment made by all the participants was that that they enjoyed using the ViPs mechanism.

- *User-based assessment:* This last step reinforced the study's hypotheses and validated the findings of the two previous steps, since its results were in-line and not conflicting. Almost all the participants stated that they found the scenarios they had to perform easy to learn and remember, but also motivating and entertaining. The major problems reported were related to the hardware that was used (e.g., graphics quality, mouse navigation, system response time) and not to the tested ViPs mechanism. This step provided complementary input towards a more complete view on the overall usability of the ViPs prototype in practice, through a detailed analysis and grouping of user errors and actions, that helped in identifying common usability problems and patterns of use. Participants completed a post-test Enjoyment Questionnaire. Most participants reported feeling highly motivated, and sometimes even happy and excited. After using each different setup (non-immersive and immersive) participants completed a 5-point scale Usability Questionnaire (see Appendix A). Some indicative results related to ViPs are provided in Table 2 (scores in this table are measured in a scale from 0 to 4, where 0 corresponds to totally disagree, 1 to disagree, 2 to neutral, 3 to agree, and 4 to strongly agree).

The implemented ViPs Mechanism...	**Non-immersive setup**	**Immersive setup**
1. is easy to learn	3.05 (±0.59)	2.90 (±0.83)
2. is easy to use	2.80 (±0.87)	2.60 (±0.73)
3. is comfortable to use	2.65 (±0.79)	2.60 (±0.92)
4. can improve the ease of task performance	2.95 (±0.80)	2.85 (±0.79)
5. can improve the efficiency of task performance	2.70 (±0.71)	2.65 (±0.79)
6. is a handy tool for orientation	3.05 (±0.67)	2.75 (±0.77)
7. is a handy tool for navigation	3.00 (±0.63)	2.90 (±0.70)
8. is a handy tool for wayfinding	3.20 (±0.75)	2.90 (±0.94)
9. is a handy tool for annotations	2.70 (±0.71)	2.80 (±0.51)
10. is overall a handy tool for VEs	2.90 (±0.62)	2.85 (±0.79)
11. is intuitive to use	2.50 (±0.67)	2.45 (±0.80)
12. is enjoyable to use	3.15 (±0.79)	2.85 (±0.65)

Table 2: Overview of user-based assessment results

(in the parenthesis, the standard deviation is provided)

Summarising the accumulated experiences with all the aforementioned methods, the following conclusions can be drawn:

- *Expert-based Review* was very efficient and cost-effective, as it quickly captured all major usability problems. A basic advantage of this method was that, since the assessors were experts, they could: (a) work with a less elaborate version of the prototype; (b) envisage potential usability problems, even for functionality or parts of the system that were not yet implemented; (c) indicate not only problems but also suggestions for solutions. Also, this method helped in finding tasks and parts of the system that should be tested with the users.
- *Co-operative Evaluation* and *User-based Assessment* were very resource demanding both for conducting the tests, but also during the analysis of the data. Questionnaires were very efficient at providing an overall insight of the

usability of the system, but in order to underpin interaction problems, their origin and the context in which they appeared, a series of questions were required throughout the co-operative evaluation and a systematic review of the user testing videos.

- When conducting a *Co-operative Evaluation* both *Thinking Aloud* and *Question and Answer* protocols should be employed, since the results of the conducted studies have shown that these two techniques produce different kind of data that have very small correlation.
- Due to the particular characteristics of the system, the results of a single method would not necessarily produce valid results, since:
  - in the case of *Expert Review*, experts might not be able to safely predict the level of usability and potential user problems due to lack of accumulated related knowledge;
  - *Co-operative Evaluation* might not allow tracking of important issues related to real "uncontrolled" use of the system such as unpredicted behaviours, usage barriers, problems and improvised solutions;
  - If solely *User Testing* was used, participants might not be able to interact with the system or know how to express the difficulties they encountered without the aid of an experienced assessor.

3.7. LESSONS LEARNT

After a long period of experimentation and testing, the concept and implementation of ViPs have been substantially re-elaborated, and undergone several enhancements. The consolidated experience from the evaluation sessions can be summarised as follows:

ViPs provide an instant sense of involvement and empowerment

One of the most prominent qualities of ViPs is that they make users feel they can affect the virtual environment, since even their simplest actions (e.g., movement) have a direct effect on it. This fact creates a higher sense of involvement and achievement, which considerably contributes to a positive user attitude towards the virtual

environment and a will to further explore and experiment with it. Furthermore, ViPs, in their simplest form (i.e., when just the very basic interaction options are offered), are very intuitive to grasp and use, even for young children and inexperienced computer users.

There is a need for task-based support tools

Initially, all ViPs-related functionality was offered through a number of context-sensitive menus. This approach was adequate for simple interactions (e.g., retrieve information about a ViP, or identify its owner) but in order to take full advantage of the capabilities of the ViPs mechanisms, tools for supporting specific tasks (e.g., navigation, display, search) must be built on top of it. In the current implementation, such tools are provided in the form of *consoles* (e.g., Figure 23, Figure 24, Figure 25) floating in front of the user's viewpoint.

The position of support tools should be user-adjustable

When active, the support tools mentioned in the previous paragraph are located inside the user's viewpoint. Thus, the user should be able to move (and rotate) the tools around, so that they do not obstruct viewing and also reside in a position that allows comfortable interaction. Such adjustments should be "remembered" and automatically set whenever the tool is used, but the user should also be able to reset initial settings and temporarily hide / minimize the tool.

Eye gaze vs. feet orientation

When released, ViPs store information about the user's virtual body position and orientation. Often, however, the user's eye gaze orientation may be totally different. Consequently, someone may miss important information when following these tracks. For example, in a virtual museum tour, the user may be required to look up at a specific position in order to see an item hanging from the ceiling. Thus, future versions of the system will need to consider storing both body and eye gaze orientation information, as well as using and visually representing them accordingly.

ViPs grouping in Virtual Paths

Practice has shown that the visualization of ViPs paths (see Figure 28) is very useful. A related problem is that, after some time, paths become very long and complex. A

potential solution is the creation of sub-paths (Virtual Paths). For example, a user may define the beginning and end of a path, also attaching semantic information which can be used for future reference and identification (e.g., route to place X).

"Smart" ViPs creation and visual representation

Automatically releasing FootViPs at specific time or space intervals is not always efficient and effective. It is more appropriate to use an "intelligent" algorithm which combines several parameters (e.g., space, time, change of direction, speed, ViPs concentration). These parameters can also be used by a ViPs displaying algorithm in order to filter available ViPs and render only a subset of them (e.g., a fixed number or a percentage).

Interaction complexity vs. customization and control

There are several ViPs attributes that the user may potentially like to change, and at different levels. For example, users can alter one or more properties (e.g., the appearance) of all ViPs, a subset of them or just a specific one. The problem is that the more power and options are provided, the more complex the interaction process becomes. There is no universal solution for this trade-off, since different users in different applications and contexts of use require diverse levels of control. The most appropriate approach is to provide alternative layers of interaction complexity.

Text input should also be supported

ViPs are loaded with different types of semantic information (temporal, spatial, owner data, etc.). In order for the user to be able to effectively manage and use this information, beyond direct manipulation, a method for text input is also required (e.g., a physical or virtual keyboard or speech recognition). For example, the user may wish to tag specific ViPs with keywords, or group / retrieve ViPs sharing some common characteristics.

Context information can also be provided when a ViP is selected

When a ViP is selected, the only information provided is about its creation date / time and owner. Depending on the application characteristics and user preferences, information about the ViP's context can also be presented, as for example the length

of the path the ViP belongs to and its relative position in it, interactions / markers that are situated along the way, how many / which other users have followed the path, etc.

3.8. DISCUSSION

In the past few years, ViPs have been presented to several audiences of diverse ages and cultural and educational backgrounds, on occasions ranging from scientific conferences to in-house demos. Interestingly, there are two major observations that occurred each and every time. First, nobody has ever questioned the utility or intuitiveness of ViPs, and, secondly, almost everybody seemed to have a personal suggestion for a new potential use of ViPs. The latter observation could be interpreted as a fact illustrating that ViPs can have a rather positive effect on people's imagination, but it might also be an indication that they are a far more powerful concept than what was initially considered. In addition to these informal observations, as reported earlier in this chapter, several formal evaluation sessions employing various methods (Expert-Based Review, Co-operative Evaluation, user-based studies) have been conducted, on the one hand, to further study the concept in terms of intuitiveness and usefulness and, on the other hand, to assess the usability of the related prototype software.

In general, the findings of the conducted studies reinforced the hypothesis that ViPs are a powerful concept, while the related software instantiation has proved easy to learn and use, as well as a handy navigation support tool. Additionally, these studies provide strong evidence that a fully functional ViPs mechanism can significantly increase the usability of VEs. A by-product of the conducted experiments was the formation of a corpus of ViPs-related guidelines covering implementation, visualization and interaction issues.

A considerable advantage of ViPs is that they can be used in any VE, in combination with any other existing navigation support approach, since they do not require any alterations of the virtual space and they are not attached to a specific input interface metaphor or device. Furthermore, the fact that ViPs have real-life counterparts with which humans are very familiar renders them an intuitive and easy to use metaphor.

Future work shall seek to further develop the envisaged software mechanism, integrate it into existing VE systems in diverse application domains, and assess its impact on the usability of such environments. Furthermore, it is planned to contribute to the evolving research domain concerned with the evaluation VEs, by testing and evolving related structured processes. Finally, since the ViPs concept, in addition to virtual environments, is also directly applicable to Augmented Reality, it is planned to experiment with its use using relevant technologies.

Overall, it can be claimed that this chapter addresses the challenges of creating a novel, intuitive, interaction metaphor [C1] for a non-typical interaction domain [C2] in ECS by introducing a new interaction metaphor (Virtual Prints) for supporting navigation, orientation, way-finding, as well as a number of additional functions in Virtual Environments and a process that can be followed towards defining and refining it.

Chapter IV

UNIVERSALLY ACCESSIBLE GAMES

4.1. INTRODUCTION

This chapter is concerned with designing for accessibility for one of the most popular computer application domains, namely games, as well as with designing based on incomplete knowledge. In this context, as part of the work for fulfilling this thesis a universally accessible chess game was designed and fully developed and made available through the World-Wide Web.

Typically, when referring to accessible computer games, it is implied that such games are purposefully developed so that they can be played by a particular group of disabled people, such as for example the blind, or the motor-impaired. However, as mentioned in the introductory chapter, this approach has several drawbacks and is in contrast with current social, ethical, technical and legal trends which all converge towards software applications and services that can be used by anyone, from anywhere and at anytime, or, in other words, towards Design for All (Stephanidis et al., 1998).

Furthermore, in a broader sense, the term accessibility should not be related just to the disabled, since very often user preferences (e.g., a user does not like to use a mouse) and technological (e.g., small screen) or environmental (e.g., noise, sun glare) constraints may also set barriers to access to, and use of, an application or service.

Thus, there is a real need for *universally* accessible games, i.e., games that:

(a) are proactively designed to optimally fit and adapt to different individual gamer characteristics without the need of further adjustments or developments;

(b) can be concurrently played among people with different (dis)abilities also when sharing the same computer;

(c) can be played on alternative technological platforms and contexts of use using a large variety of devices (including assistive technologies).

Such an objective may at first seem to be extremely overwhelming, or even impossible to achieve, since the related design parameters and use cases can easily

become countless. The aim of the work reported in this chapter is to prove that universally accessible games are practically feasible and not just abstract theoretical constructs. Although the conducted research is concerned with the wider domain of accessible computer games, the focus of work presented in this Chapter is mainly on board games. The main reasons for selecting this game genre are:

(a) Board games have been a very popular game genre for more than four thousand years and proved in practice to have a very long (re) playability value.

(b) They appeal to almost everyone, irrespective of gender, age or cultural background.

(c) Most people are already familiar with them.

(d) They mainly exist in a physical form, implying that people can also "physically" experience them.

(e) In general, they have a small set of rules, which are simple, while not simplistic, and easy to learn, understand, and memorize.

(f) They are fun, intellectually challenging and support intensive social interaction.

(g) They are played on a board that represents a well-defined, physically constrained, static game world that allows to be rendered through alternative modalities.

(h) They are based on thinking, not on reflex-based reacting. This fact can compensate for any physical disabilities and also allow for longer interaction times.

(i) Most of them, in their original form, are not accessible to people with disabilities, but it is feasible to make their digital counterparts universally accessible.

In this context, this chapter presents UA-Chess, which is world-wide the first practical application of Design for All in the domain of computer games. UA-Chess is a fully-functional chess game that can be played through a standard Web browser. Its distinctive characteristic is that it can be concurrently played by people with different abilities and preferences, including people with disabilities (e.g., low-vision, blind and hand-motor impaired). UA-Chess supports two-player games over the Internet, as

well as games with two opponents sharing the same computer, where the game's user interface (input and output) is adapted to the active player's profile.

The rest of the chapter is structured as follows: section 4.2 describes the interaction capabilities of UA-Chess and how these can be employed by people with disabilities. A detailed account is provided of the user interface, of its adaptation capabilities and of the characteristics and mechanics of the interaction techniques used. Section 4.3 highlights issues related to the approach followed in the implementation of the game, as well as to the technology adopted. Section 4.4 reports on a preliminary usability evaluation of UA-Chess and presents the related results. Section 4.5 concludes the chapter with a discussion on the benefits and innovative aspects of this work.

UA-Chess is freely available on the Web through the site of ICS-FORTH, at www.ics.forth.gr/uachess.

4.2. INTERACTING WITH UA-CHESS

UA-Chess supports alternative input and output modalities and interaction techniques that can co-exist and co-operate in the game's user interface, in combination with customizable player profiles. Every aspect of the game's functionality is fully accessible through the mouse, the keyboard (or any type of switches emulating keystrokes) and speech recognition. UA-Chess has self-voicing capabilities, provided by a built-in screen reader that offers auditory access to every part of the interface. Additionally, the game can be sized according to user preference and zoomed in and out at different levels. Finally, several alternative interaction techniques (the parameters of which can be customized) are supported for each device. UA-Chess follows the official Laws of Chess, as these are set by the World Chess Federation (FIDE). UA-Chess also supports network games. Players connect to a virtual game meeting room, through which they can invite or be invited for an on-line game.

4.2.1. User profiles

UA-Chess can adapt to alternative user profiles. User profiles are sets of predefined (either by the system, or the user) preferences regarding the available interaction options, such as, for example, speech input / output, switch-based scanning, orientation of the board, etc. The game supports two types of profiles: fixed and user-defined. The parameters of fixed profiles can be altered while playing a game, but any changes made will be lost upon exiting the program. On the contrary, user-defined profiles permanently store (in the Web browser) user preferences. The utility of fixed profiles is that they can ensure a basic level of accessibility for a specific group of users, irrespective of any preferences set by the previous player. For example, the profile for a blind user would become inaccessible if speech output was turned off.

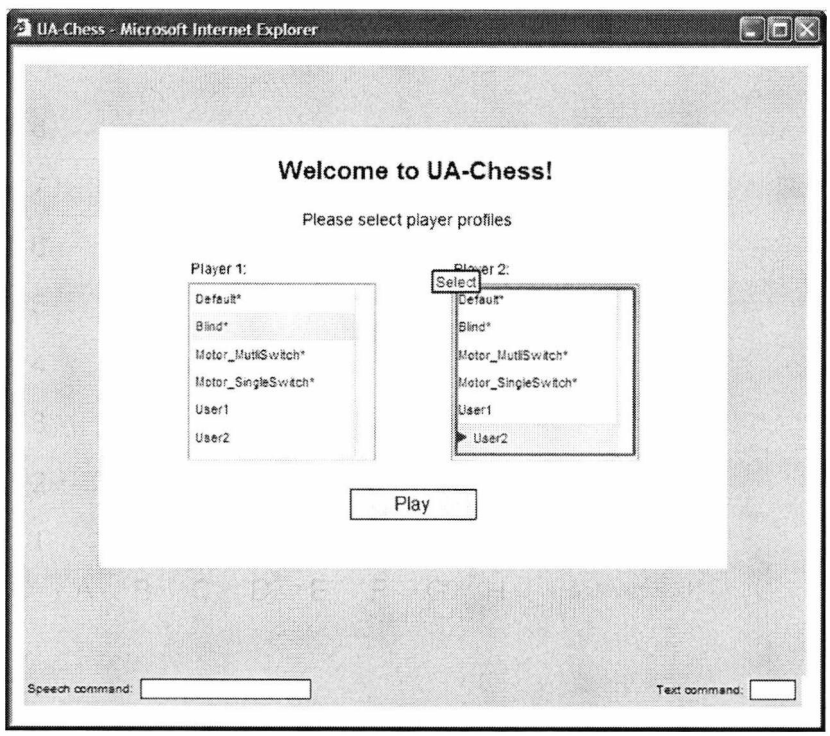


Figure 31: User profile selection interface in UA-Chess

Currently, the game offers four fixed and two user-defined profiles (see Figure 31 - fixed profiles are followed by an asterisk '*'). Both players can use the same profile, while the game "remembers" the profiles selected the last time it was played, so that it is not necessary to set profiles every time a new game starts.

4.2.2. Main User Interface

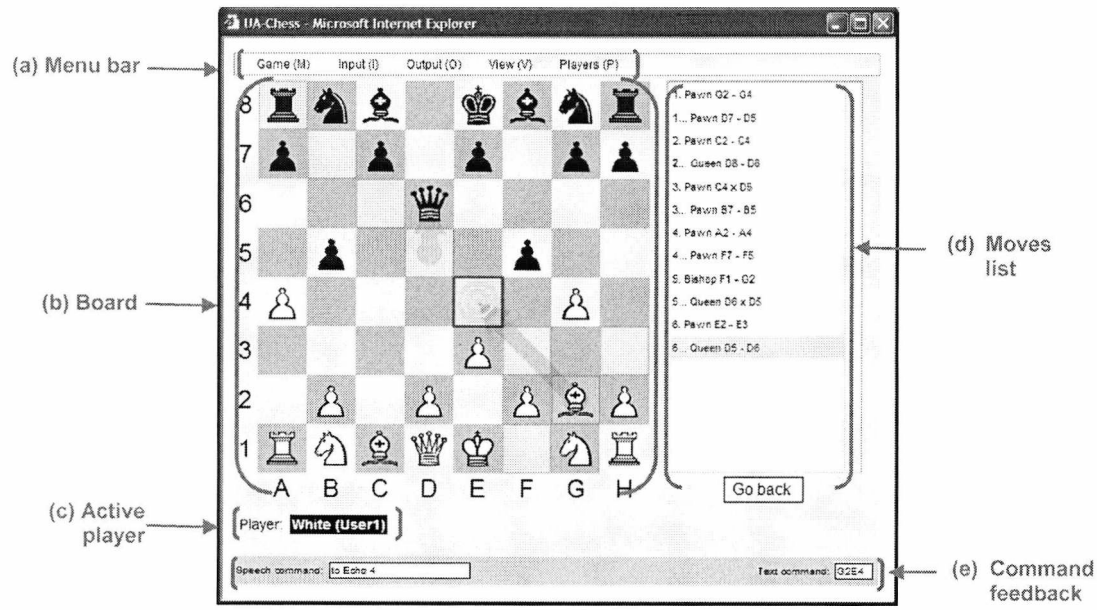


Figure 32: UA-Chess main user interface

The game's main user interface (Figure 32) comprises the following parts:

(a) ***Menu bar***: All the game functions (e.g., new, load, save, quit) are available, as well as user profile customization parameters (e.g., input / output preferences), are available from this menu.

(b) ***Board***: The Board is the part of the screen where the game takes place. It is composed of an 8 by 8 grid of 64 squares alternately light and dark. The eight vertical columns of squares are called "files" and are numbered from 1 to 8, while the eight horizontal rows of squares are called "ranks" and are numbered from "A" to "H". A straight line of squares of the same colour, touching corner to corner, is called a "diagonal". The board can present to the player (visually or orally) game-related information, such as valid moves, the last move played, if the king is in check, etc. When a pawn reaches the rank furthest from its starting position it must be exchanged as part of the same move for a queen, rook, bishop or knight of the same colour (Figure 33). Such an exchange of a pawn for another piece is called "promotion".

(c) ***Active player***: The active player is the one who will perform the next move. The related user profile is activated, to which the user interface is adapted.

(d) ***Moves history list***: It is a list displaying all the moves made so far in a game. The list uses an adapted shorthand chess notation for the description of moves, but when an item is read it is "translated" to plain language. For example, the entry "2... Queen G8 x G5" will be read as "move 2, black queen moved from D8 to G5 capturing a pawn". By pressing the "Go back" button, the game rolls back to the move before the one selected in the moves list.

(e) ***Command feedback line***: Provides feedback about speech and text commands that were recognized by the system.

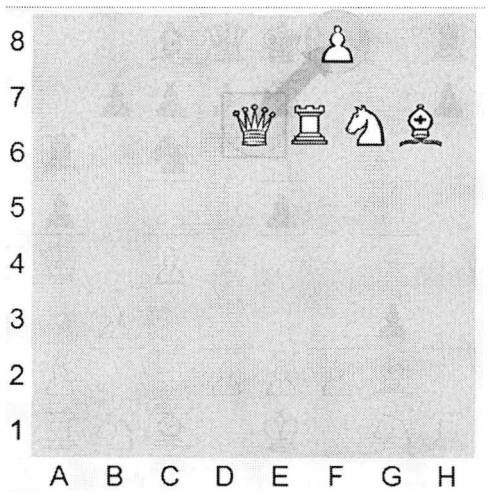


Figure 33: Pawn promotion

4.2.2.1. Moves list notation

Each item in the moves list represents a game move. Each list item comprises the following elements:

(a) The number of the move. The moves made by the Black player are followed by three dots ("...").

(b) The name of the piece that has moved.

(c) The name of the square from which the piece departed. The name consists of the "file" letter and the "rank" number (e.g., A2).

(d) An "x" if a piece was captured or a minus sign "-" if not.

(e) The name of the square to which the piece moved.

(f) If a pawn was promoted, an equal sign "=" followed by the name of the piece to which the pawn was promoted.

(g) If an "en-passant" capture was made, an "x" followed by the name of the square on which the captured pawn was.

(h) If kingside (short) castling was executed, "O-O".

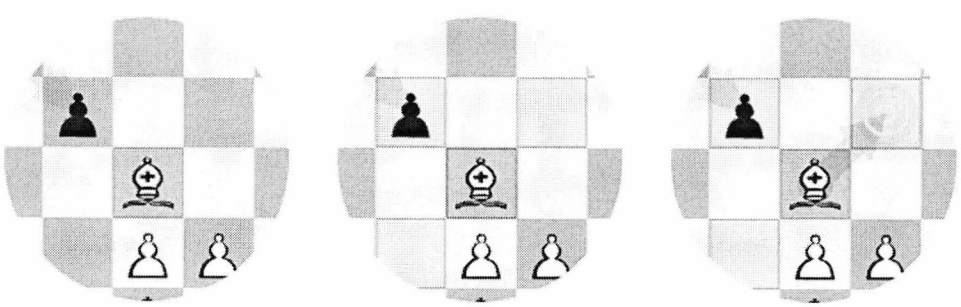
(i) I f queenside (long) castling was executed, "O-O-O".

4.2.3. Input

UA-chess supports several alternative input modalities, devices, and interaction techniques that can be used exclusively or concurrently.

4.2.3.1. Using the mouse

Similarly to any "classic" Windows application, the game can be played using a mouse. For example, when the cursor is over a piece that can move, the square on which the piece resides is highlighted with a yellow colour (Figure 34a). By clicking on the piece, it becomes selected, and the square is highlighted with orange, while all the squares to which the selected piece can move are highlighted with yellow (Figure 34b). When the mouse is over a square to which the piece can move, an orange target appears and a green arrow connects the selected piece with the target square (Figure 34c). By clicking on the target square, the selected piece moves there.



(a) **(b)** **(c)**

Figure 34: Selecting and moving a piece using the mouse

4.2.3.2. Using switch-based hierarchical scanning

Scanning is a technique that is mainly used for providing computer access to people with hand-motor impairments. The basic idea is that a special "marker" (e.g., a coloured frame) indicates the interaction item (e.g., a button, a menu, a piece) that has the input focus. The user can shift the focus marker to the next / previous interaction object using any kind of switches (e.g., keyboard keys, special switch hardware, or voice). When the focus is over an object the user wants to interact with (e.g., a chess piece to be selected, a button to be pressed), another switch is used that indicates "selection". Additionally, in cases where the user can use just a single switch, focus movement can be automatically generated by the system on constant time intervals. This variation of the technique is referred as "automatic scanning", while any other case is generically called "manual scanning" (even if hands are not used at all). UA-Chess supports scanning through the keyboard (or any switch emulating the keyboard events), speech recognition, and the mouse wheel.

In general, there are two types of objects the user can interact with:

(a) *Container objects*, used to group related objects and increase scanning efficiency. When such an object is selected, scanning is "locked" inside its contents until the user selects to exit. Examples of container objects are: the menus, the moves list, the chessboard, as well as the pieces, since each piece is considered as a collection of possible moves.

(b) *Simple objects*, which cannot contain any other object. When such an object is selected, a corresponding action is performed. Examples of simple objects are: buttons, destination moves, menu and list items.

Depending on the object type, the focus frame can have two different states, indicated by different colours and frame shapes:

1. *Select/ enter state* (green rectangle, Figure 35a). Upon selection, if the object's type is:

   (a) "*simple*", then a related action will be activated (e.g., a menu item will be selected, a button pressed, a piece will move to the selected square).

(b) "*container*", then the scanning focus will shift to its contents (e.g., the board's pieces, the moves list's or a menu's contents).

2. *Exit state* (red rectangle with an X, Figure 35b). Only "container" objects can be in this state. Upon user selection, the state changes to "enter", so that the user can either re-enter the object's contents (using "select"), or move on (using "next" / "previous"). Note: the "X" symbol is used so that colour blind players can differentiate the two states.

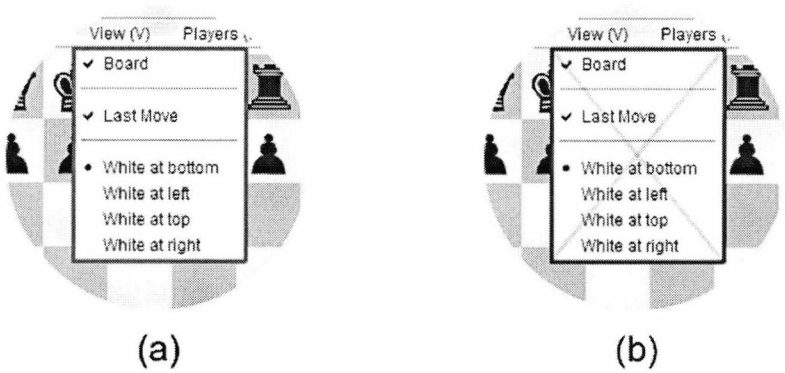


Figure 35: Scanning: Different states of the input focus

Example of play using scanning:

1. At first the focus is on the chessboard (Figure 36a). The user presses "`select`" to enter.
2. The focus (i.e., green frame) shifts over the first piece that can move, which is highlighted (Figure 36b).
3. Using "`next`" / "`previous`", the user switches between all the pieces that can move.
4. The user decides to move a specific piece and presses "`select`" when the focus is on it.
5. The focus is transferred to the first possible move (Figure 36c).
6. Using "`next`" / "`previous`", the user scans all possible moves. When the focus frame returns to the selected piece it is in an "exit" state, (red colour with an "X",

see Figure 36d), allowing the user to "unselect" the piece and move on to another one.

7. The focus is over the desired move, and the user presses "`select`" (Figure 36e).
8. The piece moves to the selected square and input control is handed to the opponent player (Figure 36f).

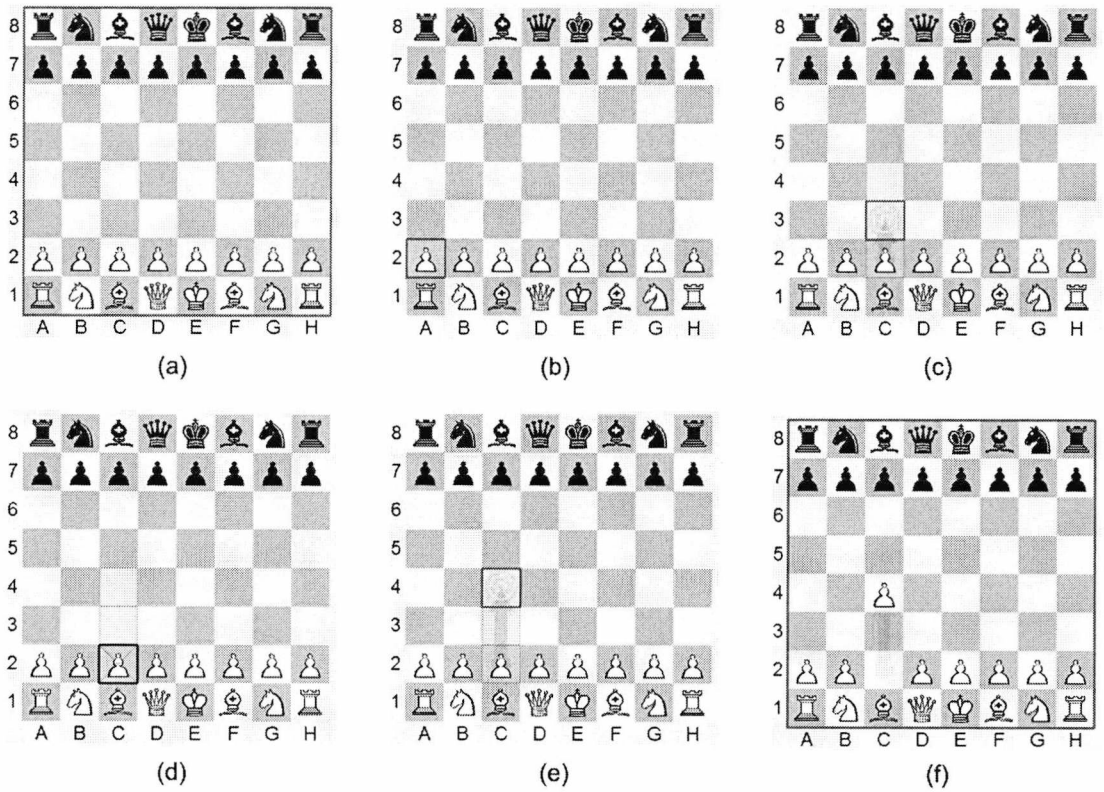


Figure 36: Selecting and moving a piece using scanning

4.2.3.3. Using the keyboard

The keyboard can be used in several alternative ways. First of all, as mentioned above, it can be used in combination with scanning for browsing and selecting interaction objects. Additionally, when hierarchical scanning is deactivated, the keyboard can be used to browse through all the available interaction objects sequentially (in the same way that the 'Tab' and 'Enter' keys can be used in Windows applications). The difference between the two approaches is that there is no "exit" state; when a "container" object is selected, the focus does not "lock" in its contents. When the last contained item is scanned, the focus moves to the next interaction

object. Hierarchical scanning is targeted to motor-impaired users, while sequential browsing is more suitable for blind users, or to anyone not using a mouse. A similar approach is supported through the arrow keys, which work much like the next / previous keys used with scanning, but in specific contexts have a "special" behaviour which allows for single-handed or non-visual use of the system. More specifically, arrow keys work as follows:

Up Arrow: When the focus is on a container object (e.g., on a menu), the object is selected and its contents are scanned.

Down Arrow: When the focus is on a simple object (e.g., a list item), it shifts to the previous / next item, respectively.

Left Arrow: When the focus is on, or inside a container object, it shifts to the previous / next object that is located at the same scan level as the object in focus.

Right Arrow: When the focus is on a simple object, it shifts to the previous / next object that is located at the same scan level as the object in focus.

Some indicative examples of use of the arrow keys are:

- With menus: when the focus is on a menu, by pressing the up/down arrow keys, the menu is selected and its contents can be browsed, while by pressing the left/arrow keys the next/previous menu becomes selected.
- With chess pieces: when the focus is on a chess piece, by pressing the up/down arrow keys, the piece becomes selected and the piece's possible moves can be browsed, while by pressing the left/arrow keys, the next/previous chess piece that can move becomes selected.
- With lists: when the focus is on a list, by pressing the up/down arrow keys, the list becomes selected and its contents can be browsed, while by pressing the left/arrow keys and the focus directly moves to next/previous object.

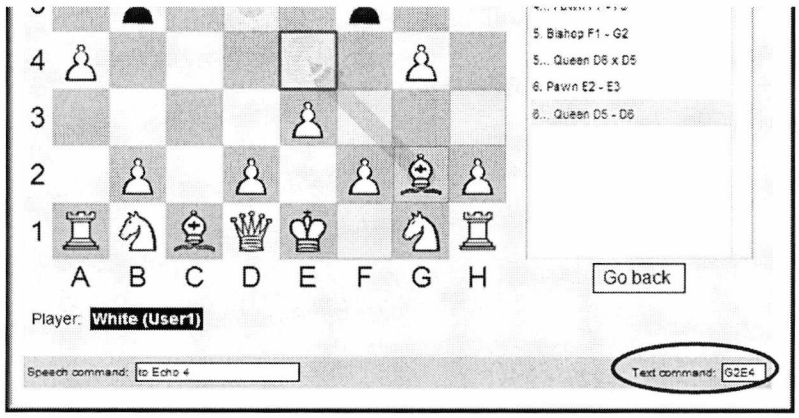


Figure 37: Text box in which recognized text commands appear

Another keyboard use is the direct selection of a piece or a valid move by typing the name of the respective square (e.g., F3). Additionally, a piece or move can be deselected, by pressing "Delete" or "Backspace" and a move executed by pressing "Space", 'END', 'Insert', '0(zero)', or "Enter". Text commands "recognized" by the system are presented in a text box labelled "Text command:" appears below the bottom right corner of the board (Figure 37). Such text commands allow easy and quick game playing for the blind, or anyone preferring (or having) to use the keyboard.

Finally, keyboard shortcuts are supported, i.e., keystrokes that provide quick access to some of the program functionality. The game supports menu shortcuts, as well as shortcuts to frequent user commands, e.g., selecting / unselecting or moving a piece, start playing, stop speaking, and repeating the last uttered sentence.

4.2.3.4. Speech recognition

Speech recognition is supported to allow totally "hands-free" access to all the functionality of UA-Chess. Every interaction object (e.g., piece, menu, button) accessible through the mouse and the keyboard is also accessible through speech. The game can recognize (depending on the current context) more than one hundred speech commands. When speech recognition is active, a text box labelled "Speech command:" is visible below the bottom left corner of the board (Figure 38), in which speech commands "heard" by the system are displayed.

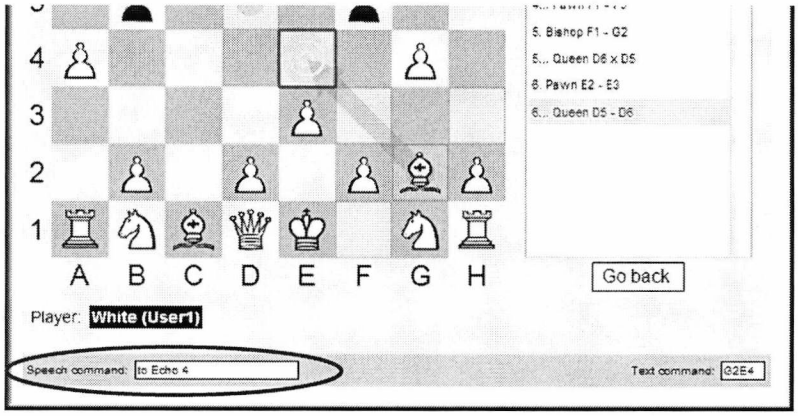


Figure 38: Text box in which recognized speech commands appear

The user can select a piece by saying "from <square name>". A <square name> is composed of the square's file (i.e., a letter from "A" to "H"), plus its rank (i.e., a number from "1" to "8"), e.g., A2, B7, C4. In addition to using the "traditional" letter names, the NATO phonetic alphabet can be used to improve speech recognition results. The NATO alphabet uses keywords for naming the letters, such as Alfa for A, Bravo for B, etc. To select a move, the user says "to <square name>" while to make the move "move".

4.2.4. Output

UA-Chess currently supports two output channels, namely visual and auditory, through speech synthesis. Users can select whether the last move made will be visible on the board. The last move is presented as an orange arrow (see Figure 39). Each player can select a different board orientation among the following: white at bottom, white at left, white at top, white at right (Figure 40). Also, an option that is quite popular for more advanced (sighted) chess players is the ability to hide the game board, since it helps in sharpening the players mental and memory skills.

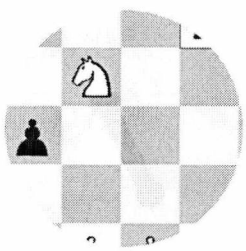


Figure 39: Representation of the last move

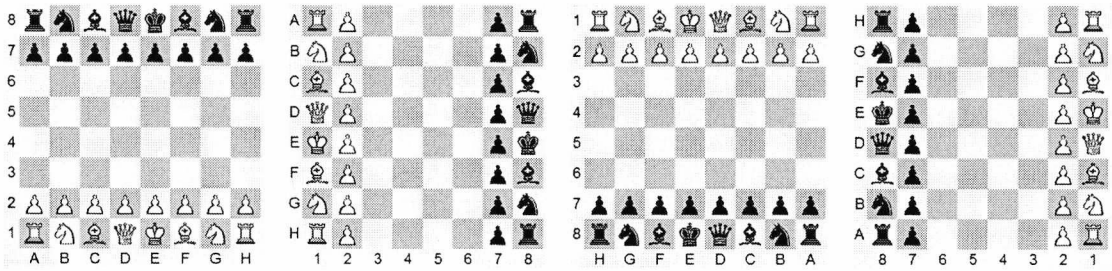


Figure 40: Alternative board orientation options

When speech is activated, the system reads audio descriptions whenever a piece or a possible move is selected, when a piece has moved and when a previous move is selected from the moves list. The game also has a built-in screen reader which, in addition to board events, provides audio information about the current focus, so that it can be used by individuals with different levels of visual impairments, or people who have dyslexia. When the screen reader mode is used in combination with hierarchical scanning, besides the current focus, the current state of the scanning frame is also described (e.g., "enter menu game", "exit board"), so that the system can be used by people who have a combination of visual and motor-impairments.

In addition to the above, the user, using the relevant menu or the corresponding shortcut keys, can ask the system to read information about the board, such as:

- the selected piece
- the contents of the chessboard
- the contents of the selected piece's rank
- the contents of the selected piece's file
- the contents of the selected piece's lower diagonals
- the contents of the selected piece's upper diagonals
- the positions of opponent pieces
- all the selected piece's possible moves
- which pieces can move
- who is playing

- the last move made
- all the moves made so far
- the current focus.

4.2.5. Network games

UA-Chess supports network games through the Internet. Players connect to a virtual game meeting room (Figure 41), through which they can invite or be invited for an on-line game. Network games are played similarly to local games.

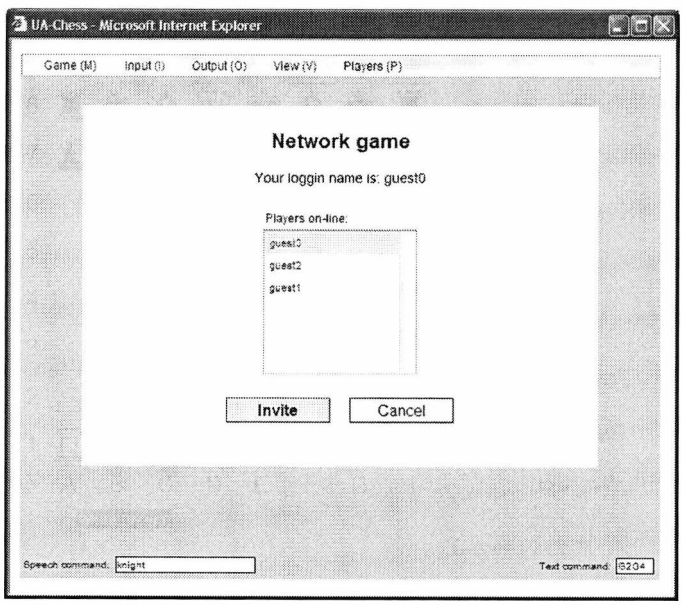


Figure 41: Network game: Inviting an opponent

4.3. IMPLEMENTATION ISSUES

A first prototype of the game was initially developed using Visual Basic (Grammenos et al., 2004) to further study the concept in practice. User tests and expert reviews allowed to improve interaction design issues and fix functional and usability problems. Experience showed that this development approach suffered from two drawbacks: (a) the game could only run on computers using the MS-Windows

operating system, thus excluding a large number of potential users; and (b) it required to be downloaded and installed, thus making it harder to get and use (especially for disabled persons), but also creating a problem with future patches and updates.

Thus, for the development of UA-Chess, Macromedia Flash MX Professional 7 was selected as the development platform. To play the game the respective plug-in is required that is available for a very large variety of operating systems (Windows, Mac OS, Linux and Solaris) and web browsers (e.g., MS Internet Explorer, Netscape, Mozilla, Opera). Since Flash does not offer speech input and output capabilities, in order to support speech recognition and synthesis Speech Application Language Tags (SALT) technology[35] was employed. SALT is an emergent standard for developing voice-enabled applications for the Web that extends existing markup languages, such as HTML and XHTML. Currently, the only SALT-compliant browser available is Microsoft Internet Explorer 6 (using a related plug-in), but the OpenSALT project[36] has announced its plans to make freely available a SALT compliant browser for Linux, based on the Mozilla web browser. In this context, a programming interface was developed to integrate SALT (that runs in the web page) with Flash (that runs as an object in the web page).

UA-Chess was developed following a modular approach. The building blocks of the game are modality-independent interaction objects, meaning that their input and output methods are generic and not tied to a specific interaction modality. These objects can adapt their presentation and behaviour according to the current user's profile. To achieve this, custom interaction objects were created (e.g., the chess board, the squares and the pieces, confirmation dialogues) and existing Flash objects were augmented (e.g., buttons, menu, lists) with related capabilities. A focus manager was also developed. This module is responsible for controlling the flow of interaction, keeping track of the currently focused interaction object, as well as "translating" multi-modal user input and correspondingly notifying all related interaction objects.

An abstract representation of the software components of UA-Chess is illustrated in Figure 42. The overall interaction process can be described as follows: whenever user

[35] http://www.saltforum.org
[36] http://hap.speech.cs.cmu.edu/salt/

input is detected (via any available input device and related interaction technique) the focus manager, based on the currently active interaction objects and the one having the focus, decides to do one of the following: (a) move the focus to another object; (b) select the currently focused item; or, (c) nothing. Sometimes, a single user input may result in multiple actions of the focus manager. For example, if the user types the name of a square on which there is a piece that can move, then the piece will get the focus and also become selected. In addition to user input, the focus manager can also receive input from the interaction objects, as in certain cases an object may pass the focus to another one. For example, when an entry from the "moves list" is selected, focus shifts to the "Go back" button, or when a confirmation dialogue closes, the focus returns to the chess board. There is also a case where the focus manager acts by itself; when the *automated scanning* technique is used, the manager changes the focus at fixed time intervals.

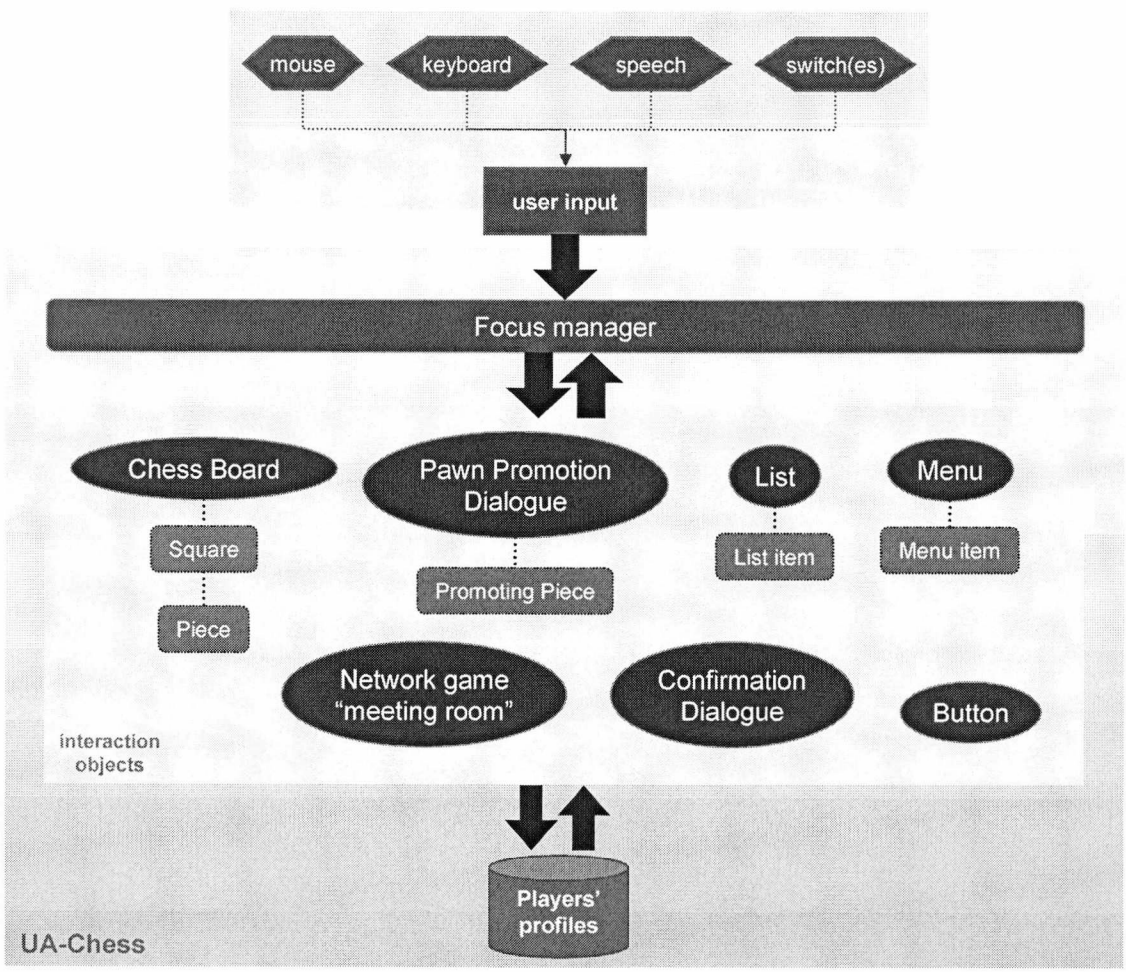


Figure 42: UA-Chess software components

4.4. USABILITY TESTING

The usability of UA-Chess has been tested both with experts and potential end-users at the Usability Lab of ICS-FORTH. The Usability Lab has professional, up-to-date equipment for assessing innovative interaction techniques and technologies, including assistive and virtual reality applications, and offers facilities for presenting a system to sample end-users in a controlled environment (e.g., control over the luminosity, temperature, and noise in the room); observing / monitoring and commenting the user's behaviour (such as facial expressions, physical movements and verbal descriptions) while interacting with the prototype; recording the user / system activity; and editing and analysing data. Figure 43 depicts views of Usability Lab's test room (left) and observation room (right), while a schematic overview of the Lab is depicted in Figure 44.

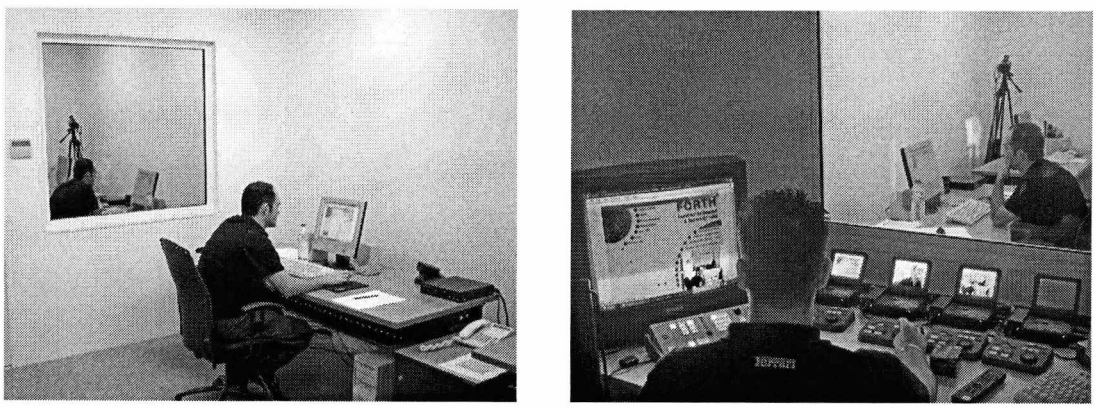


Figure 43: Views of Usability Lab's test room (left) and observation room (right)

To obtain preliminary feedback about the opinion of potential users of the system, an initial subjective usability evaluation process has been carried out with six participants, one of which was an able-bodied user, two blind users and three motor-impaired users. All users were familiar with using computers and the Internet and were able to speak and understand English. The evaluation process has been carried out around two basic scenarios, actually constituting two rounds per player in an informal competition tournament. All players had some prior knowledge of chess, but it was deemed necessary to spend some introductory time on the game's official rules. Furthermore, all the players were introduced to alternative the input / output methods

and techniques that could use in order to play the game, depending on their individual (dis)abilities and preferences. The player pairs for each round were randomly chosen.

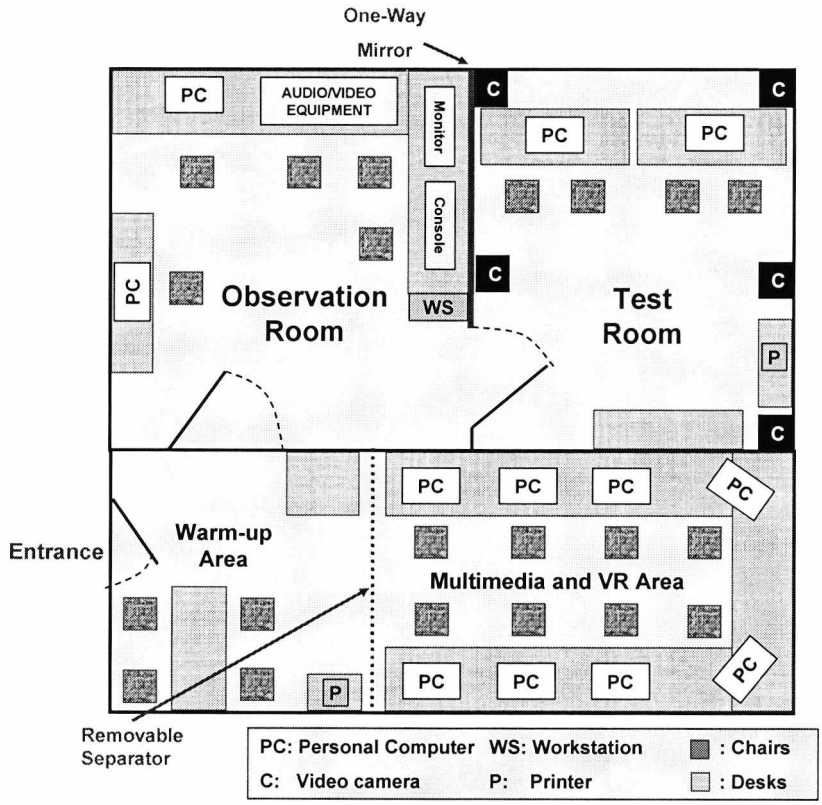


Figure 44: Overview of the Usability Lab

To solicit the user's opinion about the system, the IBM Usability Satisfaction Questionnaires (Lewis, 1995) were used. These questionnaires constitute an instrument for measuring the user's subjective opinion in a scenario-based situation. Two types of questionnaires are typically used; the first, namely After-Scenario Questionnaire (ASQ), is filled in by each participant at the end of each scenario (so it may be used several times during an evaluation session), while the other one, namely Computer System Usability Questionnaire (CSUQ) is filled in at the end of the evaluation (one questionnaire per participant). Due to the nature of the software some of the items of the questionnaire needed to be rephrased (e.g., the word "system" was replaced by "computer game", references to completing work, were replaced with "playing"). In case the user could not read or fill in the questionnaire due to a physical impairment, appropriate help was provided by the evaluator who also provided clarifications whenever needed. In the cases where the questionnaire data had to be

filled in by the evaluator, at the end of the filling in process all the questions and answers were shown (or read, depending on the case) to the user, in order to ensure that no error was made. The questionnaires used are included in Appendix B.

The primary criteria used to select the IBM Computer Usability Satisfaction Questionnaires as opposed to other questionnaires, include the following. Firstly, these questionnaires are available for public use, whereas most alternatives require the acquisition of a license from their vendors. Secondly, and most importantly, the IBM Computer Usability Satisfaction Questionnaires have shown to be extremely reliable (0.94). Thirdly, the IBM Computer Usability Satisfaction Questionnaires do not require any special software since they are not computationally demanding. Fourthly, the time required to analyse the results is very little. Finally, another important determinant was the fact that the IBM Computer Usability Satisfaction Questionnaires have been satisfactorily in use for several years now at various industrial sites and research centres.

The result of the subjective evaluation with the IBM Computer Usability Satisfaction Questionnaires is a set of metrics which can be summarised as follows:

- ASQ metric provides an indication of a participant's satisfaction with the system for a given scenario;
- OVERALL metric provides an indication of the overall satisfaction score;
- SYSUSE metric provides an indication of the system's usefulness;
- INFOQUAL metric is the score for information quality;
- INTERQUAL metric is the score for interface quality.

A small post-evaluation questionnaire was also used to provide information on the background of the subjects.

The results of the study were quite encouraging, although somehow expected. More specifically, apart from the very good scores given by the blind and motor impaired users for all metrics (all metrics are below 3, that is considered to be a very good score), it became very quickly clear that the disabled users were enthusiastic with the fact that they were given the opportunity to actually play a computer game. Moreover,

one very positive element in the overall evaluation process was that the participants did not know their opponent until the game was actually completed. This was proved to be another unexpected motivating factor that turned the study into a more interesting "event", something quite typical for computer-game play sessions.

The results of the evaluation study are summarised in Table 3. The very positive opinion of disabled users for the UA-Chess game is evident from the low-scores given for all metrics (lower values correspond to a better evaluation than higher values). On the contrary, the able-bodied user that has participated in the evaluation experiment was far less enthusiastic with the chess game, in comparison to the rest of the participants. In an informal interview with the this end-user it quickly became evident that the low score was mainly due to the expectations for more fancy visualizations, graphical effects and animations, typically found in today's action games. Although future work will include incorporating within UA-Chess such appealing features, those were not clearly the main objective of the original design of the game.

Participant	User Category	scenario 1 ASQ (play round 1)	scenario 2 ASQ (play round 2)	OVERALL	SYSUSE	INFOQUAL	INTERQUAL
Player 1	Blind	2.33	2.67	2.37	2.25	2.43	2.67
Player 2	Blind	2.67	2.67	2.53	2.63	2.43	2.67
Player 3	Motor-impaired	2.33	2.00	2.32	2.38	2.29	2.33
Player 4	Motor-impaired	2.00	2.00	2.05	2.13	2.16	2.00
Player 5	Motor-impaired	2.33	2.33	2.42	2.38	2.43	2.67
Player 6	Able-bodied	4.00	3.67	4.11	4.25	4.14	3.67
	Average	**2.61**	**2.56**	**2.63**	**2.67**	**2.65**	**2.67**

Table 3: The results of the chess usability evaluation process.

(The range is from 1 to 7, where 1 is the highest / best possible score)

Since UA-Chess is a free Web application, in addition to formal usability testing in the laboratory, user feedback is also collected through the Internet from real end-users of the game.

4.5. DISCUSSION

UA-Chess is a universally accessible Internet-based chess game that can be concurrently played by two gamers with different (dis)abilities, using a variety of alternative input / output modalities and techniques in any combination. Another innovative characteristic of UA-Chess is that, besides network-based play, it allows the two gamers to play sharing the same computer. In this case, the interface of UA-Chess is alternately adapted to the characteristics of the player who has the move.

UA-Chess is worldwide the first, and currently the only game that exhibits the above unique characteristics and was a finalist for the European Design for All Awards, constituting a practical demonstration of the application of Design for All principles, methods and tools in the development of software applications. In the overall context of promoting Universal Access to the Information Society and raising awareness in the software development community about Design for All issues, UA-Chess can be seen as a good practice example, demonstrating that Universal Access is a challenge rather than a utopia.

In summary, in UA-Chess the following main strategies for achieving universally accessible games have been implemented:

(a) Support of alternative interaction methods and modalities that can co-exist and co-operate in the game's user interface.

(b) Ability of the user interface to adapt to alternative user profiles (i.e., sets of preferences, requirements and needs).

Admittedly, accomplishing the goals of Design for All may not be a trivial task, but it certainly is a manageable task. It requires handling and understanding a very large design space, comprising diverse users, operating in several different contexts of use, which maybe not all be known at design time, and also mapping and transforming all the related requirements and (dis)abilities to coherent, usable and accessible interaction designs.

To this end, an essential prerequisite is *inclusive* and *open* interaction design. *Inclusive* means that the broadest possible population is considered during the design phase, while *open* means that, later on, it will be still possible to expand the design so that it can cater for more user categories and contexts of use, e.g., by supporting additional interaction methods, techniques or devices. A basic approach for achieving this is to separate the game's content and mechanics from modalities through which these can be accessed by, and presented to, the player. Overall, a positive point is that, very often, a single accessibility design solution may be applicable to several situations beyond the ones originally foreseen.

The designing challenges faced in this chapter had to do with designing for accessibility [C4] (and more specifically for Universal Access) and also designing based on incomplete knowledge [C5] in the context of creating universally accessible games, which are proactively designed to optimally fit and adapt to different individual gamer characteristics and (dis)abilities without the need of further adjustments or developments. The outcomes of the work were a fully-functional universally accessible Web-based board game, the design of which was based on the principles of Design for All, and a detailed account of how accessibility is supported for different user categories through the game's interface, its adaptation capabilities and the available alternative input and output modalities. The game is publicly available on the Web and was nominated for the final jury decision of the European Design for All Awards set by the European Commission, in the category "AT/Culture, Leisure and Sport".

In conclusion, the benefits of developing games accessible to all users are self-evident, since such games strongly cater for the needs and actively support the right of all people for social interaction and play, irrespective of their individual differences, thus providing a steppingstone towards a more inclusive (and fun) Information Society.

Chapter V

ENVIRONMENT SENSING FOR SOFTWARE AGENTS

5.1. INTRODUCTION

This chapter is concerned with extending the capabilities of an existing interaction paradigm to better support interaction designers and also designing for openness and extensibility. For the needs of this part of the thesis work, a synthetic sensory system was modelled and developed. In order to assess, refine and test the system's utility and capabilities, alternative variations of it have been "implanted" to simple embodied agents (or "intelligent" game creatures) and were used to control their interaction with each other and their surrounding virtual environment.

As described in section 2.4.4, existing efforts in agent sensing tackle the task of agent sensing on a "case by case" basis. Furthermore, the supported synthetic senses are tightly coupled to the characteristics of the particular agents and virtual environments. As a result, several limitations can be identified:

1. *Openness/Extensibility*: The sensors and their characteristics are hard-coded in each application. There is no easy way to view or alter their parameters, e.g., change the range, position, or sensitivity. Furthermore, it is not possible to add / remove sensors or define alternative sensors, e.g., for other real (e.g., smell) or fictitious (e.g., telepathy) senses.
2. *Scalability*: The sensory system's sophistication cannot be modified, to accommodate for example the sensing needs of more / less complex agents.
3. *Reusability*: The sensory system can not be directly "transplanted" to other projects.
4. *Flexibility*: When the application is running, the sensory system cannot be dynamically reconfigured, e.g., change what a specific sensor is detecting, alter the system's components, etc.

In this context, the motivation for the work presented in this chapter was to develop a generic approach for simulating a synthetic sensory system overcoming the above limitations. Thus, the main design goal was to create a sensory system model which would be generic, scalable, open and extensible, exhibiting the following characteristics:

(a) Applicability to any type of agent, independently of its intrinsic characteristics, or of the senses and stimuli being simulated;

(b) Support for developing sensory systems of variable sophistication;

(c) Possibility of editing and modification at any time;

(d) Possibility of ease enrichment with new (real or imaginary) senses.

Additional concerns were that the resulting model should be easy to understand, concise and computationally efficient, and that it should allow game designers to equip game creatures with alternative sensing capabilities, with a minimum (ideally with no) need for programming.

The rest of the chapter is structured as follows: section 5.2 introduces the basic concepts. Section 5.3 describes how the modules comprising the sensory system are modeled, and section 5.4 presents the process and algorithms used by the developed system. Section 5.5 contains a number of attributes and characteristics identified and adopted, on the one hand for providing higher design flexibility and more options to game developers, and on the other hand for improving system performance. Section 5.6 presents how the sensory system can be fully specified using XML. Section 5.7 describes how the suggested sensory system can be integrated into an intelligent creature, and section 5.8 discusses indicative uses of the system. Finally, section 5.9 concludes the chapter with a critical review of the current implementation, planned improvements and directions for future work.

5.2. CONCEPTS OVERVIEW

This section introduces the basic concepts that are used for modeling the synthetic sensory system. First of all, a *creature* is defined as "any game entity that has the ability to sense and / or emit one or more stimuli" (see Figure 45). In this sense, even inanimate objects (such as a security camera that can *see* images, or a computer screen that *emits* an image) are also deemed as creatures. The overall approach builds upon three simple basic concepts:

a. *Sensors*. i.e., the means through which a creature collects information about the environment, e.g., other creatures and their actions. A creature can have an unlimited number of sensors. Some indicative examples are: eye, hand, ear, mouth, body, soul, mind reader, heat / motion detector, infra-red, etc.
b. *Stimuli* emitted from a creature to the game world. These can be sensed by the *sensors* of other creatures. Each creature can emit multiple stimuli. Some examples include: image, voice, skin, aura, sound, taste, thoughts, heat, heartbeat, smell, etc.
c. *Sensory Modules Manager* (SMM). This is a mechanism responsible for controlling and coordinating the interaction between all the active creatures' sensors and stimuli.

Both *sensors* and *stimuli* are generically termed *Sensory Modules* (SMods).

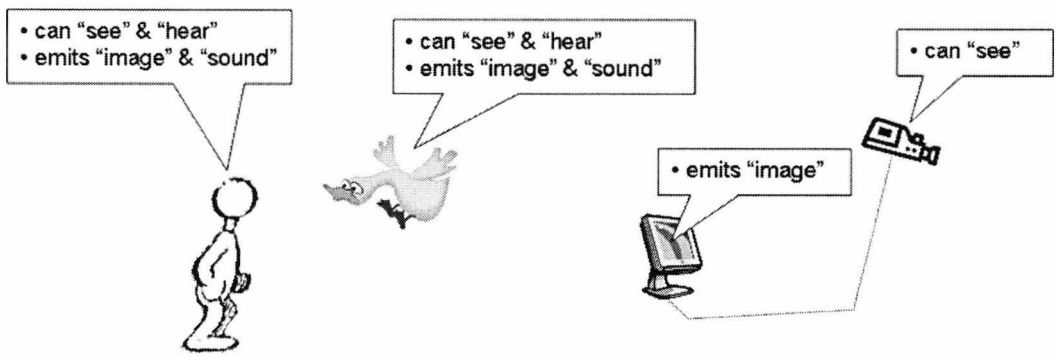


Figure 45: Examples of game creatures

5.3. MODELLING THE SENSORY MODULES

Each sensory module has an *id* (a descriptive name, e.g., eye, ear, lens), a related *sense* (e.g., vision, hearing, 6th, X-ray) and a *type*, which can be either "sensor" or "stimulus". In the pilot implementation, sensors and stimuli are physically modelled as two-dimensional geometrical shapes (samples are illustrated in Figure 46).

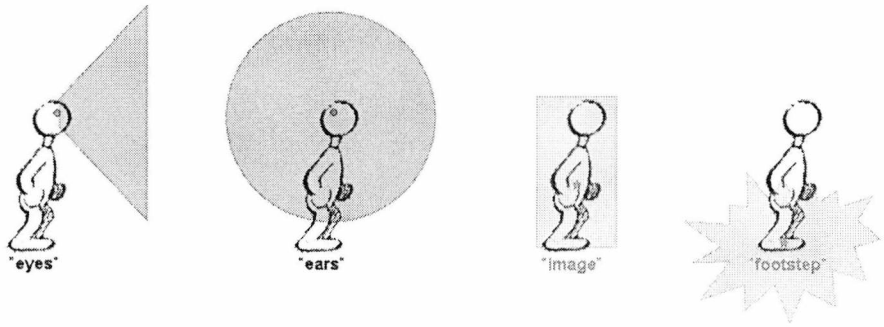


Figure 46: Sample sensors and stimuli models[37]

A sensor is considered to have sensed a particular stimulus if the corresponding shapes overlap. For example, in Figure 47 the 2D area of the "walker's" "eyes" sensor overlaps with the "duck's" "image" stimulus, and thus it is considered that the "walker" can *see* the "duck". On the contrary, in Figure 48 the "walker" cannot *hear* the "duck", since the "ears" area does not intersect the "quack" sound.

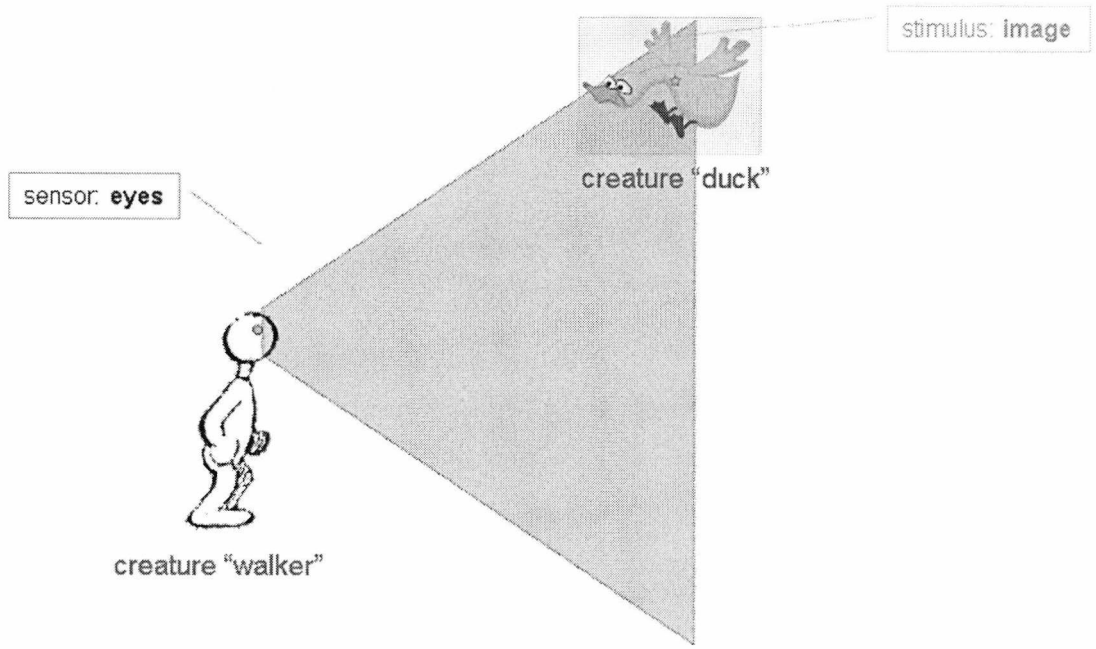


Figure 47: Example of a sensor able to sense a particular stimulus (i.e., the walker *sees* the duck)

[37] The character presented in the illustration is from the book "Cartoon Animation" by Preston Blair (Walter Foster Publishing, 1995).

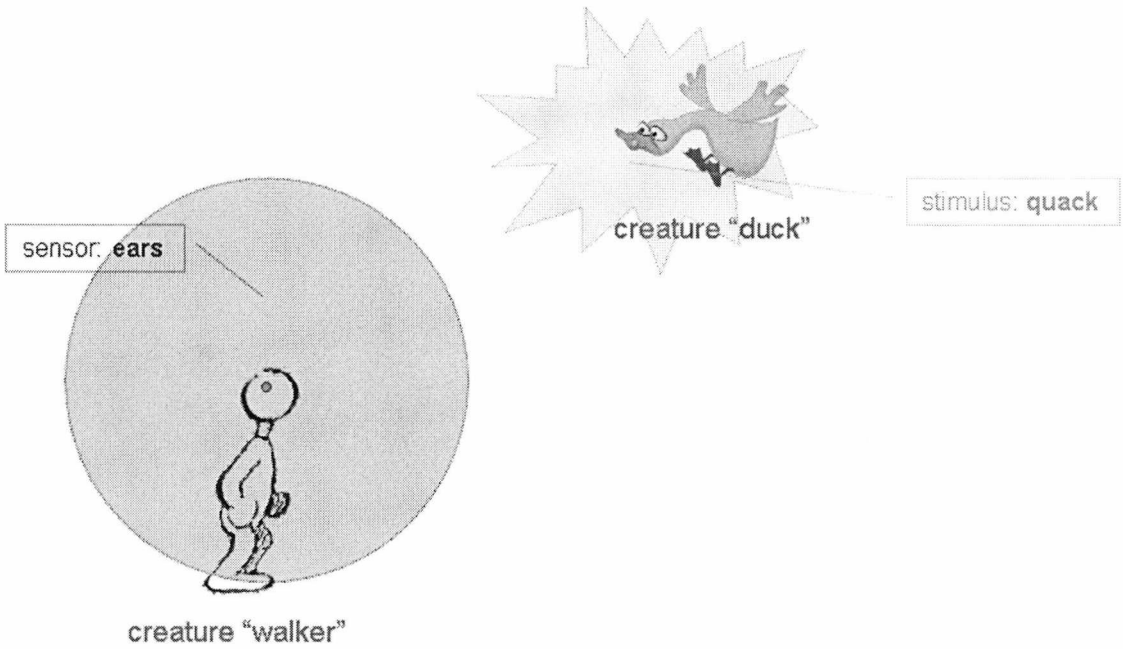


Figure 48: Example of a sensor not able to sense a stimulus (i.e., the walker does not *hear* the duck)

Currently, six shape types are supported (Figure 49): lines, polylines, viewcones, polygons, circles and rectangles.

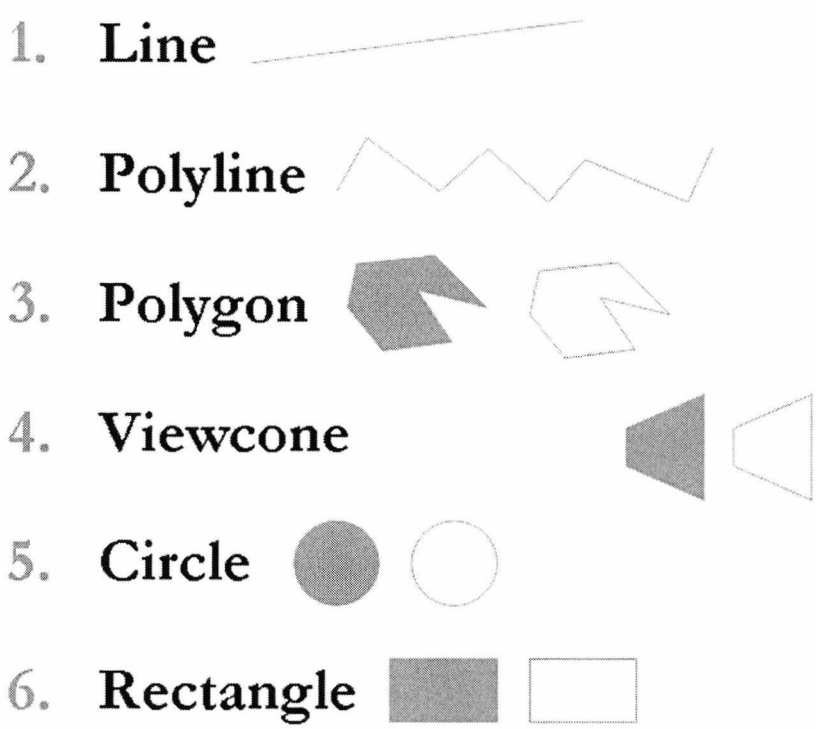


Figure 49: Supported geometrical shapes

A geometrical shape associated to an SMod can have the following attributes:

- *Position* (Figure 50): The shape's absolute location with respect to the virtual body of the SMods' owner creature. A SMod can also be located "outside" a creature's body, e.g., ahead or behind it.
- *Alignment* (Figure 51): The shape's relative position. Both horizontal (i.e., left, centre, right) and vertical (i.e., top, middle, bottom) alignment is supported.
- *Rotation* (Figure 52): The shape can be rotated to any angle, e.g., to follow the creature's head movement.
- *Flip* (Figure 53): This parameter allows mirroring the shape. This has proved to be a handy option for 2D games, since it can accommodate the creatures' change of facing direction.
- *Opaque* (Figure 58): Indicates if the whole area of the shape should be considered for detecting possible overlaps, or just its outline.

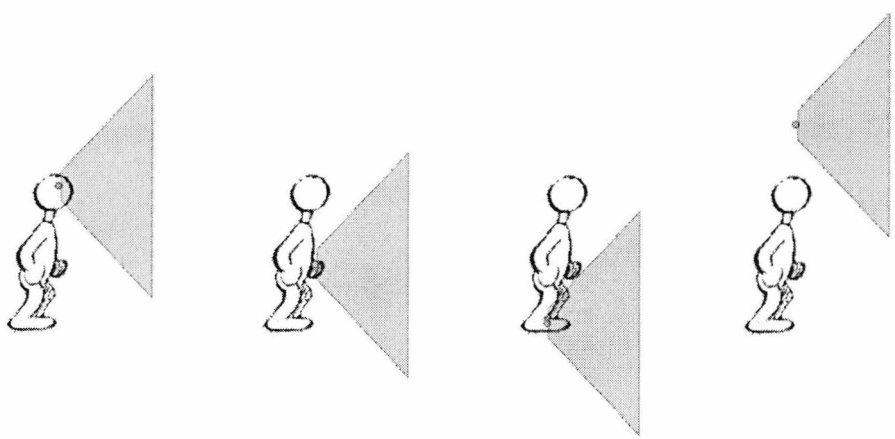


Figure 50: Alternative SMod positions

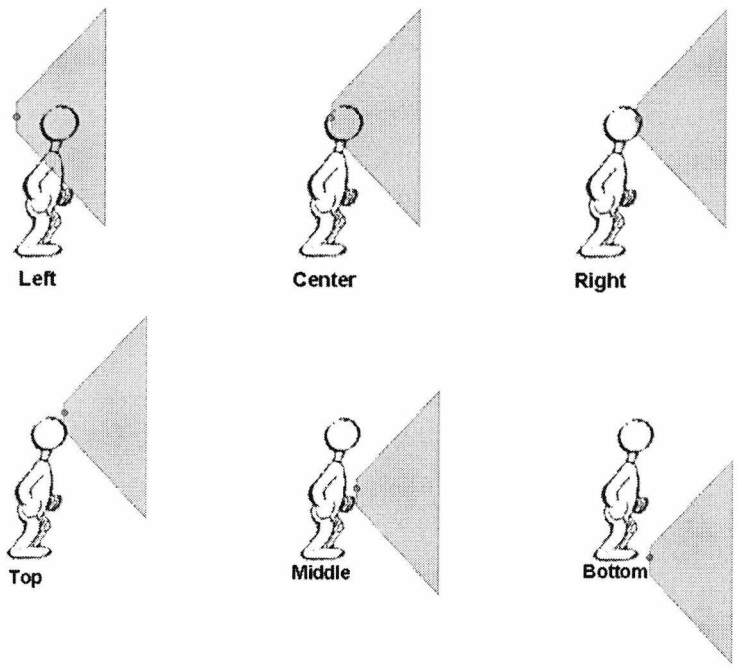


Figure 51: Alternative SMod alignments

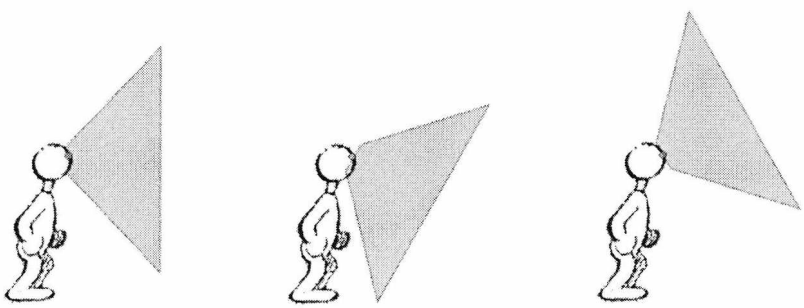


Figure 52: Alternative SMod rotations

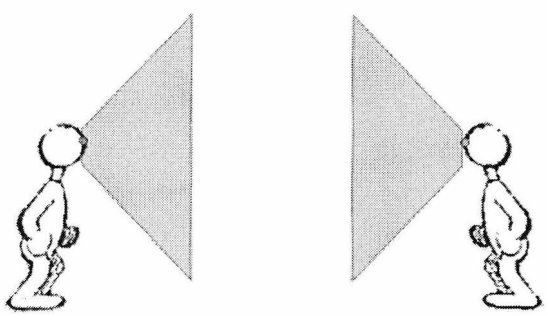


Figure 53: SMod flipping

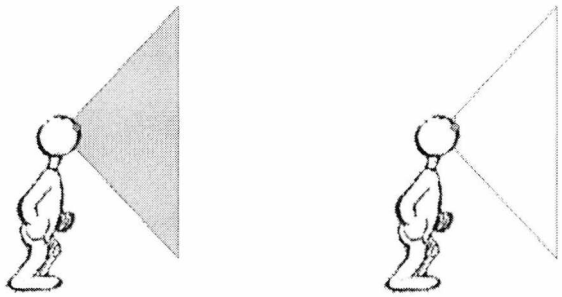


Figure 54: SMod opaque attribute

5.4. MODELLING THE ACT OF SENSING

The Sensory Modules are employed to simulate the act of sensing in a two-step process:

Step1 - SMods Registration: When a sensor or stimulus is added to a creature, it is also registered to the Sensory Modules Manager. For example, in Figure 55, a creature named "walker" registers an "eyes" sensor for the sense of "vision", while the "duck" registers a "vision" stimulus entitled "image".

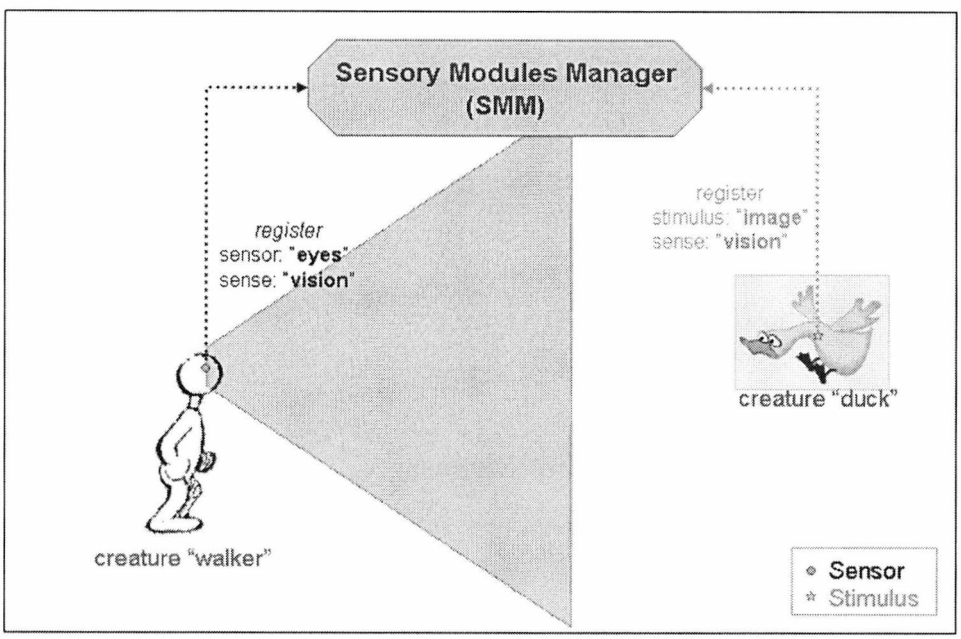


Figure 55: Modelling Sensing: Step 1, registering sensory modules

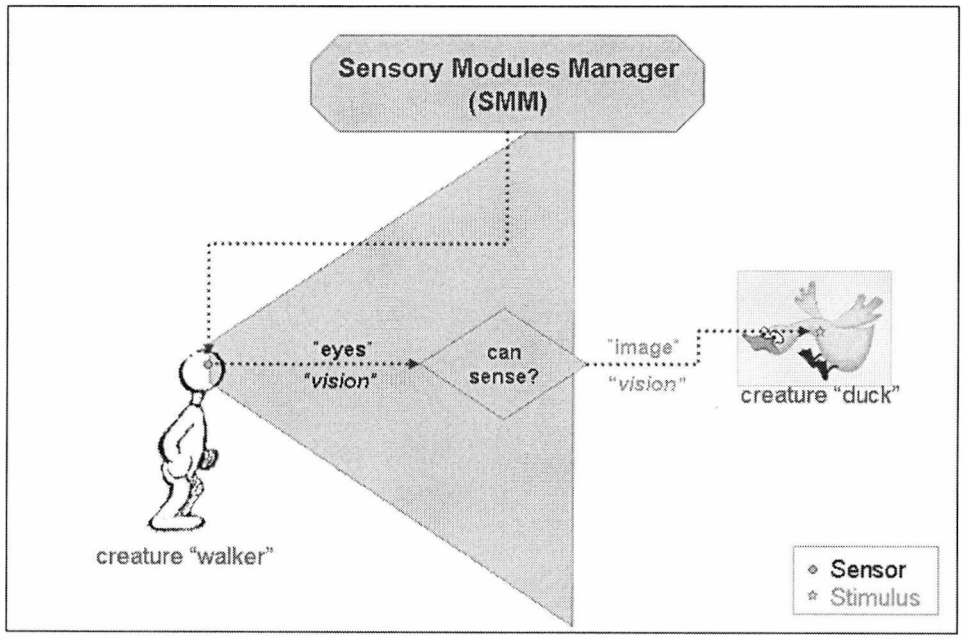


Figure 56: Modelling Sensing: Step 2, matching sensory modules

Step2 - SMods Matching: During game play, at regular intervals, the SMM requests from every registered sensor to test if it can sense any of the available stimuli targeting the same "sense". This procedure is defined as a *sensing round.* For example, in Figure 56, the SMM requests the walker's eyes to test if they can sense the duck's image. The SMods matching algorithm can be described using the following pseudo code:

```
for each Sensor Sens_i
     for each Stimulus Stim_j
          if Sens_i.Sense == Stim_j.Sense
               if Sens_i.2DShape.overlaps(Stim_j.2DShape)
                    Sens_i.canSense(Stim_j)
               else
                    Sens_i.canNotSense(Stim_j)
```

A drawback of the code presented above is that every sensor is matched against every stimulus (sharing the same *sense*). This means that in an environment containing C number of creatures, with S sensors and M stimuli each, the computational complexity can rise up to CxSxM. Typically, in computer games, as opposed to the

real world, synthetic creatures are not interested in every other creature, or in every stimulus of a particular creature. Thus, the matching routine can be considerably speeded up by adding descriptions (in the form of rules) about which creature types, specific creatures or stimuli a sensor is "interested in". For example, the "eyes" sensor of a "hunter" creature may be only interested in any "duck" creature, and the "ears" of a "fireman", in "alarm" stimuli with a "fire" id. In order to implement this improvement both the registration and the matching phases need to change as follows:

***Step1 - SMods Registration*:** Whenever a new sensor is registered to the SMM, it is checked if there are any registered stimuli that match its set of "interest" rules. Any matches found are added to a list associated with the sensor (the "interests list"). Whenever a stimulus is registered, if there are any registered sensors "interested in" it, then the stimulus is added to their "interests list".

***Step2 - SMods Matching*:** During this phase, each registered sensor is tested only against the stimuli in its "interests list", instead of every registered stimulus. Thus, the related pseudo code changes to:

```
for each Sensor Sens_i
      for each Stimulus Stim_j in Sens_i.stimuliInterestsList()
            if Sens_i.Sense == Stim_j.Sense
                  if Sens_i.2DShape.overlaps(Stim_j.2DShape)
                        Sens_i.canSense(Stim_j)
                  else
                        Sens_i.canNotSense(Stim_j)
```

5.5. ADDITIONAL ATTRIBUTES OF THE SENSORY MODULES

While developing the pilot system, several use case scenarios were used to test the range of its possible applications, as well as its functional capabilities and limitations. These tests resulted in the extension of the basic model of the sensory modules with several additional attributes which can provide higher flexibility and more design options to game developers, but also improve system performance. These include:

- *Active attribute*: A Boolean value (i.e., true/false) that indicates whether the SMod should participate or just be ignored during the matching phase.
- *Events*: At the end of each sensing round, every SMod raises an event notifying the creature if it was triggered (or not). For example, if a sensor found at least one stimulus that it could sense, then it would raise an "onTriggered" event. Correspondingly, a stimulus that was not sensed by any sensor would raise an "onNotTriggered" event.
- *Event-related actions*: Events can be used to automatically trigger actions that affect the state (i.e., activate / deactivate) of any other SMod of the same creature. Thus, for example, the fact that a sensor failed to sense may be used to activate one, or more, different kinds of sensors, e.g., when the "eyes" sensor of a "hunter" can not sense a "duck", a "binoculars" sensor is activated that offers longer-range vision. Event actions can also be used for implementing level-of-detail (LOD) sensing. In Figure 57, a simple rectangular sensor (part a) is initially used since the related calculations are very efficient, until a creature of interest is detected. Then, (part b) a viewcone is activated that better approximates the final simulation of vision. Finally (part c), since an "interesting" creature is still in sight, a complex vision model is used, employing two sensors.
- *Interest rules and related events*: These are the rules mentioned in the previous section which are used to improve the efficiency of the sensory system and describe categories or specific creatures and / or stimuli. Each sensor may

have several interest rules. Just like SMods, interest rules can be activated or deactivated at any time, and raise events at the end of each sensing round, denoting whether they have been triggered or not. These events can be used to affect the state of other interest rules or SMods belonging to the creature. For example, when the "look for duck" rule of a "hunter" is not triggered, a "desperately look for any prey" rule is activated along with a "prey radar" sensor.

- *Priority*: Both SMods and interest rules can be attributed a priority number. The matching algorithm evaluates items with higher priority first. So for example, it can be ensured that the rule for sensing a "duck" will be always triggered before the rule for sensing a "grouse".

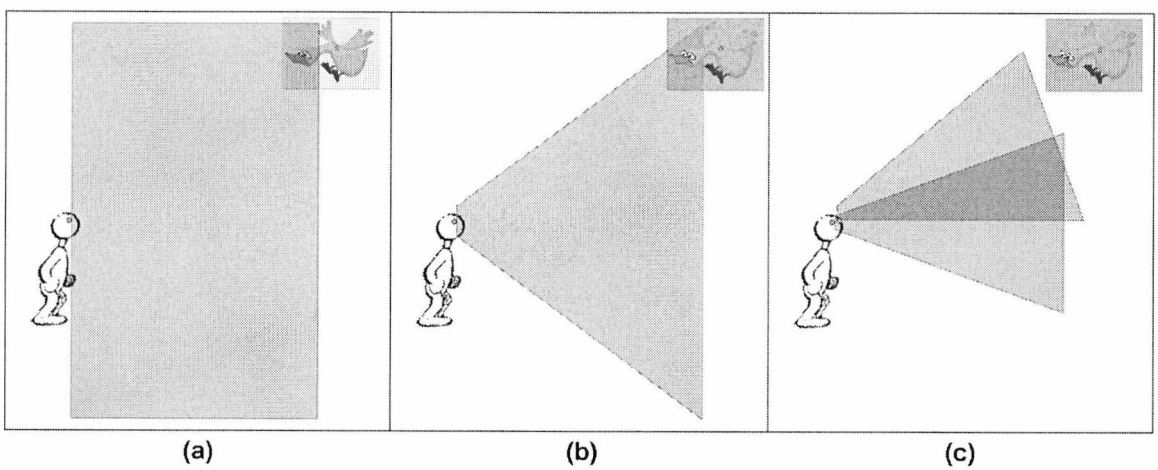


Figure 57: Example of sensing at different levels of detail

5.5.1. Timing attributes

The Sensory Modules Manager has an internal logical clock (i.e., a counter) that advances by one unit (a *tick*) each time it runs. SMM time can be different than "game time", since, usually, in order to increase system performance, the AI code does not run in every game loop. This clock is used for tagging sensing events, so that it is possible to differentiate the older from the newer ones, but also to perform scheduling of the required computations. With regards to timing, SMods can have the following attributes:

Lifespan: A number denoting for how many ticks of the counter the SMod will be active. When this time expires, SMM automatically deactivates the SMod.
Period: Specific time intervals at which the SMod will be active.
Activate after: How many "ticks" after its registration the SMod will be activated.

The *period* and *activate after* attributes can be used in combination for load balancing sensor execution. For example, if there are 3 sensors, namely A, B, and C that have a common period of 3 ticks, but are activated sequentially, then on each tick SMM will have to evaluate only one of them (Figure 58).

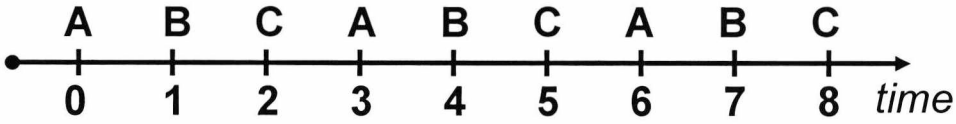


Figure 58: Example of load balancing sensor execution

5.6. SPECIFYING THE SENSORY SYSTEM USING XML

To minimise the need for programming and maximise the ease of use of the system, the sensory system can be fully specified using Extensible Markup Language[38] (XML). XML is a simple, very flexible industry-standard protocol administered by the World Wide Web Consortium (W3C) that has derived from ISO Standard SGML (Standard Generalized Markup Language). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

This option allows game designers to use a descriptive approach for keeping track of their design at a conceptual level, and also easily experiment with alternative configurations and ideas. Furthermore, it contributes to reusability and to communication and sharing of sensory systems.

[38] http://www.w3.org/XML/

The content of a sensory system XML file has the following abstract form:

```
<sensorySystem>

        <sensor id="sensor1" atrribute1 atrribute2 ... atrributeX>
                <sensor elements>
        </sensor>

        <stimulus id="stimulus1" atrribute1 atrribute2 ... atrributeY>
                <stimulus elements>
        </stimulus>

        ...

        <stimulus id="stimulusm" atrribute1 atrribute2 ... atrributeA>
                <stimulus elements>
        </stimulus>

        <sensor id="sensorn" atrribute1 atrribute2 ... atrributeZ>
                <sensor elements>
        </sensor>

</sensorySystem>
```

The `sensorySystem` can contain an unlimited number of `sensor` and `stimulus` elements (SMods), in any order. A `sensor` or `stimulus` can have the following attributes (Table 4):

Attribute	Value	Required	Default value	Description
id	string	yes	-	A unique name used for referring to the SMod.
sense	string	yes	-	A sense to which it is related (can be any user-defined string).
x	integer	no	0	Horizontal position in relation to its owner (creature).
y	integer	no	0	Vertical position in relation to its owner (creature).
active	boolean	no	true	Indicates whether it will participate or just be ignored during the matching phase.
lifespan	unsigned integer	no	∞	For how many ticks it will be active.
period	unsigned integer	no	1	Specific time intervals at which it will be active.
activateAfter	unsigned integer	no	0	How many "ticks" after its registration it will be activated
priority	unsigned integer	no	0	The matching algorithm evaluates items with higher priority first.

Table 4: Attributes of the sensor and stimulus elements

To define a new sensor or stimulus only the two required attributes are needed. All the other are optional and can be totally omitted. Some examples of sensors and stimuli definitions are:

```
<sensor id="ears" sense="hearing">
</sensor>
```

```
<sensor id="eyes" sense="vision" x="38" y="-98" active="false" lifespan="10"
period="100" activateAfter="2" priority="100">
</sensor>
```

```
<stimulus id="image" sense="vision" >
</stimulus>

<stimulus id="sound" sense="hearing" y="12" activateAfter="25" priority="1">
</stimulus>
```

To describe the shape that models the SMod, one or more XML elements are used. If none of these elements is included in the definition of the SMod, then the creature's bounding box is used to model the SMod. These elements, depending on the shape, are presented in Table 5.

Shape	Element(s)	Value
Line	range	unsigned integer
Polyline	points	$[(x_1, y_1)(x_2, y_2)...(x_n, y_n)]$
Polygon	points	$[(x_1, y_1)(x_2, y_2)...(x_n, y_n)(x_1, y_1)]$
Viewcone	range angle	unsigned integer integer
Circle	radius	unsigned integer
Rectangle	rangeH rangeV	unsigned integer unsigned integer

Table 5: Elements for defining the SMod's shape

Examples:

```
//circular sensor
<sensor id="ears" sense="hearing">
      <radius value="200"/>
</sensor>

//polygon sensor
<sensor id="eyes" sense="vision>
      <points value="(0,0)(100,100)(100,-100)(150,-100)(150,-50)(0,0)"/>
</sensor>

//rectangular stimulus
<stimulus id="image" sense="vision" >
```

```
        <rangeH value="400"/>
        <rangeV value="400"/>
</stimulus>
```

As mentioned in section 5.3, there are several transformations that can be applied to an SMod's shape. These are defined through the following elements (Table 6):

Element(s)	Values	Description
hAlign	L, C, R, LEFT, CENTER, RIGHT	The SMod's horizontal position in relation to the bounding box of its owner creature.
vAlign	T, M, B, TOP, MIDDLE, BOTTOM	The SMod's vertical position in relation to the bounding box of its owner creature.
rectAlign	L, C, R, LEFT, CENTER, RIGHT	If the SMod is modeled as a rectangle, an additional alignment parameter is supported which sets the position of the shape in relation to its starting point.
rotation	integer	An angle to which the SMod will rotate.
flip	boolean	If set to true, the shape will be mirrored.
opaque	Boolean	Indicates if the whole area of the shape should be considered for detecting possible overlaps, or just its outline.

Table 6: Elements for transforming the SMod's shape

For obvious reasons, the rotation and flip elements have no meaning when the SMod is modelled as a circle, and the opaque element does not affect lines and polylines.
Both sensors and stimuli support event-based actions (see section 5.5) which affect the state (i.e., activate / deactivate) of any other SMod of the same creature. Such actions are defined through the onEvent element, which has the form:

```
<onEvent name action type sense id />
```

The attributes used for describing an onEvent element are described in Table 7. All attributes are required.

Attribute	Value(s)	Description
name	onTriggered, onNotTriggered	Specifies the event that will trigger the action.
action	activate, deactivate	What will happen to the SMod specified by the type, sense and id attributes.
type	sensor, stimulus	These three attributes must uniquely identify an SMod of the same creature.
sense	string	
id	string	

Table 7: onEvent attributes

Examples of using the onEvent element:

(a) If the "ears" sensor is not triggered by a stimulus, then a "hearing aid" sensor is activated.

```
<sensor id="ears" sense="hearing">
       <onEvent name="onNotTrigerred" action="activate" type="sensor"
                sense="hearing" id="hearingAid" />
</sensor>
```

(b) When the "heat" stimulus is sensed by a sensor, a "heat detection counter measures" stimulus is activated, and a "radar" sensor is deactivated.

```
<stimulus id="heat" sense="heatDetection">
       <onEvent name="onTrigerred" action="activate" type="stimulus"
                sense="heatDetection" id="heatDetectionCountermeasures" />
       <onEvent name="onTrigerred" action="deactivate" type="sensor"
                sense="radioDetection" id="radar" />
</sensor>
```

Finally, sensors can have interest rules (see section 5.5) which describe categories or specific creatures and / or stimuli that the sensor is interested in. Interest rules are specified using the interestedIn element, which has the following form:

```
<interestedIn ruleId stimulusId creatureType creatureId onSensed onNotSensed
active priority>
</interestedIn>
```

The attributes of the interestedIn element are described in Table 8.

Attribute	Value(s)	Required	Description
ruleId	string	yes	A unique name used for referring to the rule.
stimulusId	string	no	These three attributes are used to describe the stimuli and / or creatures that the sensor is interested in. If all three attributes are specified, then only a unique stimulus of a specific creature will match. If any of them is omitted then groups of creatures and / or stimuli will match.
creatureType	string	no	
creatureId	string	no	
onSensed	string	no	User-defined callback functions. If any of them is specified, then when the sensors has (or not) sensed a stimulus that matches this rule, the corresponding function will be called.
onNotSensed	string	no	
active	boolean	no	Indicates whether it will participate or just be ignored during the matching phase.
priority	unsigned integer	no	The matching algorithm evaluates rules with higher priority first.

Table 8: onEvent attributes

In correspondence to SMods, interest rules can also have associated event-related actions that can be used to activate / deactivate other rules of the same sensor, or SMods of the same creature. These actions are defined as follows:

(a) For activating / deactivating other rules:

```
<onRuleEvent name action ruleId />
```

(b) For activating / deactivating other SMods

```
<onRuleEvent name action type sense id />
```

The attributes used for describing an onRuleEvent element are described in Table 9. In both the above cases, all the related attributes are required.

Attribute	Value(s)	Description
name	onTriggered, onNotTriggered	Specifies the event that will trigger the action.
action	activate, deactivate	What will happen to the SMod specified by the type, sense and id attributes
ruleId	string	These attributes must identify an interest rule of the same sensor.
type	sensor, stimulus	These three attributes must uniquely identify an SMod of the same creature.
sense	string	
id	string	

Table 9: onRuleEvent attributes

An example of interest rules and related events is provided below, where, initially, the "eyes" sensor is interested in seeing any duck (using "eyesrule1"). If a duck is seen, then "eyesrule2" is activated (which was inactive), and the sensor gets interested in the "image" stimulus of a particular "duck" creature named "Donald". When this rule is triggered, a user-defined function named "hasSeenDonaldDuck" will be called and also a sensor named "eyes2" will be deactivated.

```
<sensor id="eyes" sense="vision">

    <interestedIn ruleId="eyesrule1" creatureType="duck"
            onSensed="hasSeenDuck" onNotSensed="hasNotSeenDuck"
            priority="1" >

            <onRuleEvent name="onTrigerred" action="activate"
                        id="eyesrule2" />

    </interestedIn>
```

```
    <interestedIn ruleId="eyesrule2" stimulusId="image"
            creatureType="duck" creatureId="Donald" active="false"
            onSensed="hasSeenDonaldDuck">
            <onRuleEvent name="onTrigerred" action="deactivate"
                        type="sensor" sense="vision" id="eyes2" />
    </interestedIn>
</sensor>
```

Below, an example of the definition of a simple sensory system comprising two sensors ("eyes" and "ears") and two stimuli (an "image" and a "walking" sound) is provided:

```
<sensorySystem>

    <sensor id="eyes" sense="vision" x="38" y="-98" active="true"
    lifespan="10" period="100" activateAfter="100" priority="10">
        <range value="400"/>
        <angle value="120"/>
        <hAlign value="CENTER"/>
        <rotation value="-10"/>
        <onEvent name="onTrigerred" action="activate" type="sensor"
                sense="vision" id="eyes2" />
        <onEvent name="onNotTrigerred" action="deactivate"
                type="stimulus" sense="hearing" id="walking" />
        <interestedIn ruleId="eyesrule1" creatureType="duck" priority="1"
                onSensed="hasSeenDuck" onNotSensed="hasNotSeenDuck">
                 <onRuleEvent name="onTrigerred" action="activate"
                           id="eyesrule2" />
        </interestedIn>
        <interestedIn ruleId="eyesrule2" stimulusId="image"
                creatureType="duck" creatureId="Donald" active="false"
                onSensed="hasSeenDonaldDuck">
                 <onRuleEvent name="onTrigerred" action="deactivate"
                           type="sensor" sense="vision" id="eyes2" />
        </interestedIn>
    </sensor>

    <sensor id="ears" sense="hearing" x="38" y="-98">
        <radius value="20"/>
        <interestedIn ruleId="earsrule1" creatureType="duck"
                onSensed="hasHeardDuck" onNotSensed="hasNotHeardDuck">
```

```
            </interestedIn>
        </sensor>

        <stimulus id="image" sense="vision" >
            <hAlign value="CENTER"/>
            <vAlign value="MIDDLE"/>
        </stimulus>

         <stimulus id="walking" sense="hearing" y="0" active="false"
                  lifespan="10" period="5" activateAfter="3" priority = "0">
            <points value="(0,0)(100,100)(100, -100)(150, -100)(150, -50)
                        (400, -200)(400, -50)(0,0)"/>
            <hAlign value="CENTER"/>
            <onEvent name="onTrigerred" action="deactivate" type="sensor"
                     sense="vision" id="eyes" />
            <onEvent name="onNotTrigerred" action="activate" type="stimulus"
                     sense="hearing" id="walking" />
        </stimulus>

</sensorySystem>
```

5.7. LINKING THE SENSORY SYSTEM TO THE "BRAIN" OF INTELLIGENT CREATURES

In general, at the end of each sensing round, all sensors report to the "brain" of the creature they belong to (using predefined callback functions) whether they have sensed any stimuli or not. In the first case, a prioritised list of stimuli is provided, along with information about the creature(s) they are emitted from. Similarly, stimuli can report if they have been sensed (and by whom) or not. Moreover, interest rules can also notify the creature if something was sensed (or not) according to them, but in this case this is optional and it is achieved through callback functions defined the developer. This combination of system-driven and user-defined events provides a very flexible and efficient programming model that can accommodate different implementation approaches and architectures, and supports the creation of creatures with both spontaneous and long-term reactions.

Another interesting feature is that creatures, apart from using their own sensors to get information about their environment, can also "snoop" other creatures' sensors, the

latter remaining intact. This is achieved through *listener sensors*. This feature allows the creation of disembodied sensors, i.e., sensors that are placed anywhere in the environment but are linked to one or more creatures. Some indicative examples include a security camera sensor that is linked to a monitor in an observation room, a microphone attached to remote speakers, or even communication through telepathy where a creature can "see" through another creature's eyes. Another use of listener sensors is centralised control. For example, a group of guards can share an alarm system, or aircrafts can use a common watch tower sensor to avoid collision.

Of course, to create intelligent creatures, sensory information alone is not enough. This information has to be fed into a "brain" (the agent's logic), be processed and result into related (re)actions. In the pilot implementation, relatively simple 2D creatures were created, the behaviour of which was driven by a group of hierarchical finite state machines (hFSMs). In short, an hFSM (Figure 59) is a collection of discrete *states* that describe a condition in which the creature is (e.g., walking, fleeing from enemy, attacking) and *transitions*, which are used to change the creature's state based on one or more events (e.g., the sensory information).

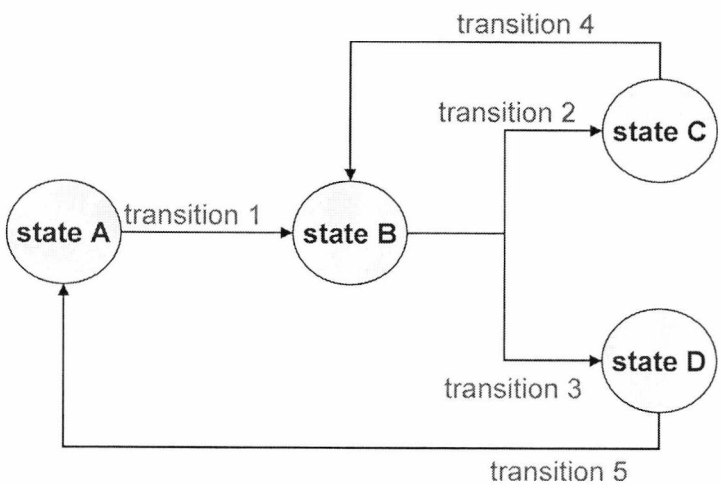


Figure 59: Abstract representation of a hierarchical finite state machine (hFSM)

Figure 60 illustrates a simple hFSM used to drive the behaviour of a "walker" creature. The creature walks towards one direction. If it reaches the end of the screen, then it turns and starts walking to the opposite direction. If at any time the creature

sees a "duck", it stops walking and starts watching it. When the duck is out of sight, the "walker" continues walking to the direction it was facing.

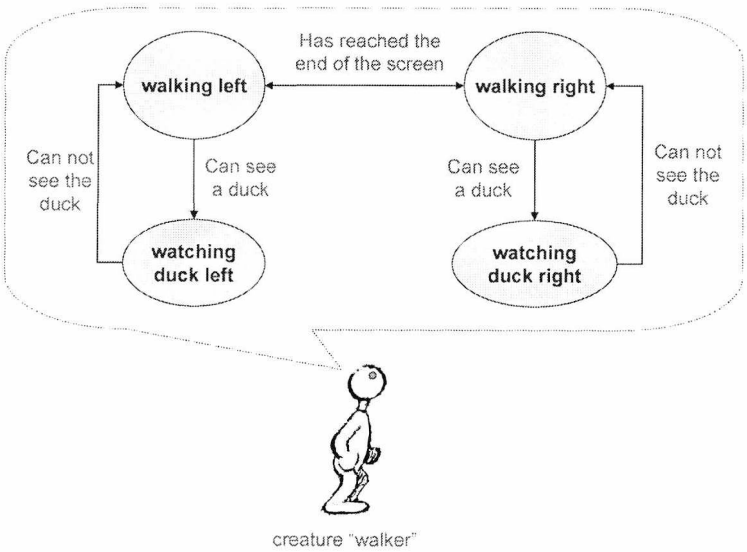


Figure 60: Abstract representation of a hierarchical finite state machine (hFSM)

In the presented approach hFSMs are used to group sets of related states and transitions into simple behaviours (e.g., walking, shooting, fleeing). A creature may have several associated hFSMs (i.e., behaviours). Complex behaviours can emerge through the combination of simpler ones (e.g., Figure 61). A creature has always an associated active hFSM (i.e., *behaviour*) and a related current state. At the end of each sensory round, the sensory system, using the callback functions presented in section 5.5, reports its findings to the creature, which "behaves" accordingly.

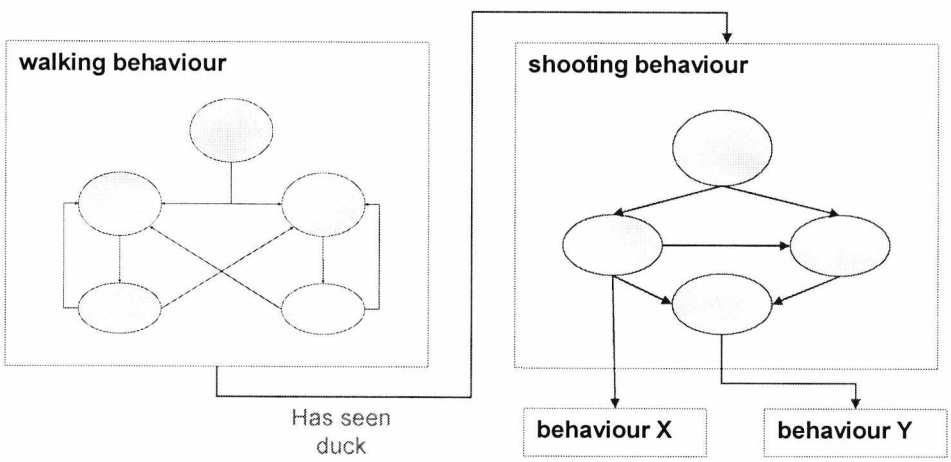


Figure 61: Example of combination of different behaviours

5.8. PROPERTIES AND USES OF THE SUGGESTED SENSORY SYSTEM

In summary, the suggested sensory system is:

1. *Open, extensible and flexible*: Sensors and stimuli are discrete, self-contained components, the properties of which can be fully manipulated both at run-time and off-line. At any time, sensors and stimuli can be added, removed, activated or de-activated.
2. *Scalable*: Depending on the sensory needs of a creature, the sensory system may range from trivial (e.g., a rectangular box used for collision detection), to an extremely complex multi-sensorial structure (e.g., imagine a species of creatures with five heads, each equipped with four eyes, 3 ears and a nose, that also have an extra eye in their tail, the ability to detect fear and they can communicate using telepathy).
3. *Reusable*: An object-oriented approach was followed for the development of the sensory system which allows the reuse of the whole framework "as such" (i.e., as a "black box"), in any other project. The only programming requirement is the implementation by the new project of a specific application programming interface (API) which is needed for the communication between the sensory system and the creature's brain.

To illustrate some of the capabilities of the suggested approach, an indicative supported scenario is presented in Figure 62:

(1) a creature (soldier) crawls into the scene; its "image" stimulus is captured by a "mirror" sensor;

(2) the "image" in the "mirror" is detected by a "camera" sensor (2);

(3) a monitor is "listening" to the camera;

(4) the camera's screen is reflected in a hand-held "mirror" that a guard is holding;

(5) the guard's "eyes" see the "image" and thus the crawling enemy is detected and intercepted!

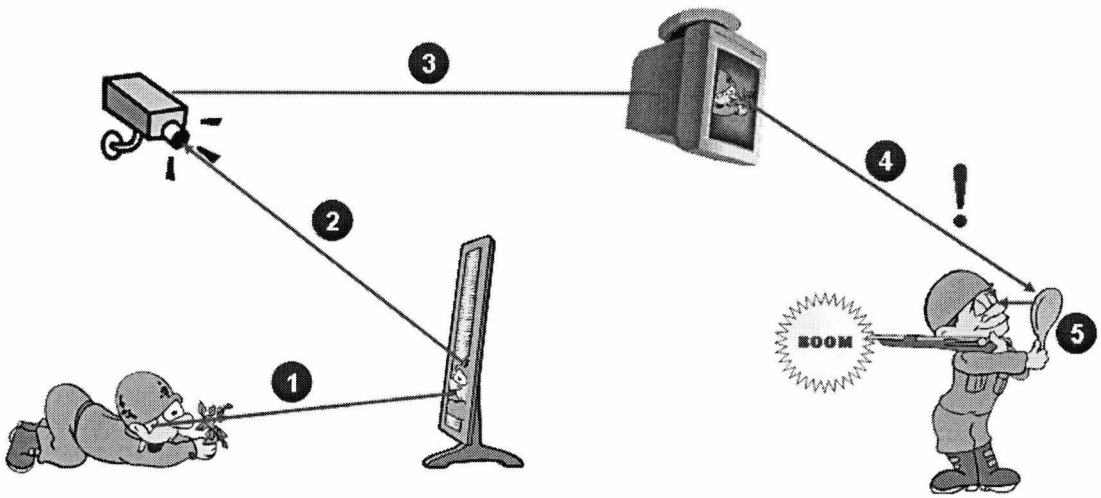


Figure 62: Example of a supported scenario

Except "typical" sensing of the environment, the suggested system can have several additional applications. Some indicative examples include, but are not limited to:

- *Conditional (de)activation*: In most games all creatures do not have to be active all of the time, spending valuable CPU time. Thus, sensors can be plugged in the environment to (de)activate selected creatures based on the player's position, or any other condition, e.g., if the player fires an alarm sensor, then X guards are activated and attack, if not, the guards will never arrive.
- *Simulating injury and damage*: Simply by deactivating related sensors, e.g., when a creature's eyes are hit or covered, or a security camera destroyed.
- *Implementing countermeasures*: By sending false stimuli to a sensor, e.g., fooling a radar or heat detector.
- *Simulating malfunction*: By randomly triggering a sensor, e.g., when battery level is low, or there is a short circuit.
- *Supporting multiple accuracy levels for a sense*: Using complementary sensors (like in the case of the multiple viewcones used in Thief - see section 2.4.4.2) that have different characteristics, e.g., simulate peripheral and direct vision, impairments as short-sightedness where vision may work very well at a close distance, but at a longer distance becomes extremely blurry.

5.9. DISCUSSION

This chapter presented a structured approach for simulating sensing in virtual worlds. The goal was not to accurately re-create the sensory system of humans, but to develop an efficient, generic, approach which would support the creation of intelligent creatures for computer games and entertainment applications. The resulting theoretical model was instantiated and tested in a 2D game world, but the concept can be easily transferred to 3D environments. One of the basic limitations of the implemented model is that it does not automatically handle occlusion. This means that in order for a stimulus to be blocked by another one (e.g., when a creature hides behind an object) it has to be explicitly programmed by the game developer. Another limitation is speed. Although the environment of Macromedia Flash 7 served as a convenient application testbed and the programming language ActionScript 2.0 was a very efficient tool for quickly completing the required developments, it is not possible to perform the level of code optimisation and fine-tuning that is required to support in real-time a large number of concurrently active sensors and stimuli.

Therefore, future work aims on the one hand to develop and provide alternative methods for supporting occlusion, and, on the other hand, to port the system to a development platform that can provide appropriate means for maximising speed and efficiency, such as, for example, C++. Finally, except the XML schema that has been created for specifying the sensory system, ongoing work includes the development of a direct manipulation tool for visually defining and attaching sensors and stimuli to creatures and also managing all the related configuration parameters.

The main challenges faced in this chapter were related to extending the capabilities of a novel interaction paradigm [C1] providing higher design flexibility and more options to designers and also designing for openness and extensibility [C5]. In this context, a generic approach to creating a synthetic sensory system was developed, applicable to any type of embodied agent in any type of virtual world, independent of the nature or the intrinsic characteristics of the senses or, stimuli simulated. The suggested model exhibits openness, scalability, reusability, extensibility, simplicity and computational efficiency.

Chapter VI

SOCIAL MULTI-PERSPECTIVE INTERACTIVE TOOLS FOR YOUNG CHILDREN

6.1. INTRODUCTION

This chapter is concerned with the design of a non-typical application, which also required the development of new metaphors, for a non-typical user group. For accomplishing the work related to this part of the thesis a large number of different constraints and requirements (related to the users, the technology and the context of use) needed to be taken into account, close cooperation with different project stakeholders was required, and every design decision had to be meticulously documented and used to drive related consensus building activities.

The "Today's Stories" project aimed to develop the social, communicative and emotional skills of children through a collaborative reflective activity based on the interesting events that take place during a day. This was to be accomplished by empowering children to create a diary of interesting events through capturing events with a wearable camera (called the KidsCam) and then viewing, annotating and editing the collected material with an appropriate software tool (called the Diary Composer), thus constructing their *Today's Stories*. The target users of the project were children aged 4 to 8, from two different countries (Denmark and Israel), as well as their teachers.

During the early phases of the project, the project partners had compiled and agreed upon a set of requirements and constraints that the user interface of the Diary Composer should meet (Today's Stories Consortium, 1999; Koutra et al., 2000). These included constraints stemming from the end-users (children), the domain experts (educators) and the technology used (hardware and software limitations), at different levels of abstraction, ranging from high-level and abstract requirements (e.g., the notion of timelines should be employed) to very specific, low-level constraints (e.g., only a specific number of videos could be supported). It has to be noted that several requirements, preferences and concepts about the user interface resulted through the employment of cooperative inquiry techniques (Druin, 1999a) with children. The design task was to create and propose a design solution for the user

interface of the Diary Composer satisfying the pre-set requirements and preferences and complying with the specified constraints.

After studying and analysing the related work and bibliography, as well as the project goals and requirements, the following high-level design goals were identified:

- Convey available functionality in a highly visual form, and avoid relying on textual representations so that the system could be language independent, as well as usable by younger children who cannot read, since "*highly visual menus and icons appear to be appealing to children and easy for them to understand and use*" (Wilson, 1988).
- Create an *open learning system*[39] (Jonassen et al., 1993) easy to adapt to suit the children preferences, cultural background (e.g., by adding self-made annotation symbols) and skills (e.g., by customising the functionality).
- Incorporate interactive elements and provide intuitive metaphors (Erickson, 1990) both in terms of the represented function, and in terms of how they could be operated upon (i.e., provide *affordances*; Norman, 1988).
- Render the presentation and interaction appealing to children by making all the components of the user interface (inter)active and providing feedback to indicate 'successful' interaction steps (Norman, 1988; Cooper, 1995), through animation (Baecker & Small, 1990) and audio effects (Mountford & Gaver, 1990) that on the one hand facilitate the comprehension of the concepts, and on the other hand promote and support exploratory styles of interaction.
- Create a 'forgiving' environment with no 'incorrect' or 'wrong' input, offering active support and guidance whenever needed (Cooper's "*Don't make the user look stupid*" and "*Make errors impossible*" design principles; Cooper, 1995).
- Be gender-neutral (but also gender-customizable) and avoid the pitfalls of gender-oriented (and usually male-oriented) design (Furger, 1998).
- Avoid the use of cumbersome input devices (e.g., keyboard, mouse) and interaction techniques (e.g., double-click).

[39] According to (Jonassen et al., 1993) *open learning systems* are those that are: (a) need driven; (b) learner-initiated; and (c) conceptually and intellectually engaging.

The ultimate goal was to provide a *transparent* user interface, enabling children to focus their energies on their activities and not on the interface (Norman, 1990), thus avoiding the interface to get in the way of exploration and knowledge construction (Winn, 1993).

The rest of the chapter is structured as follows: section 6.2 provides an overview of the overall system concept on which the interaction design was based. Section 6.3 outlines the process followed for performing the interaction design while Section 6.4 describes the outcomes of this process, along with issues and problems tackled during the design phase, adopted solutions and followed assumptions and conventions. Section 5.5 reports on the outcomes of the evaluation of an interactive prototype of the designed user interface and presents the resulting re-design suggestions. Finally, Section 6.6 concludes the chapter with a brief discussion.

The work reported in this Chapter has been carried out under a subcontract of the Institute of Computer Science of the Foundation for Research and Technology – Hellas commissioned by the Lambrakis Research Foundation, in the context of the project "Today's Stories" (ESPRIT-i3-ESE Project No. 29312) funded by the European Commission in the framework of the Intelligent Information Interfaces (i^3), Experimental School Environment Programme.

6.2. CONCEPT OVERVIEW

The main concept of the "Today's Stories" is illustrated in Figure 63. Young children are equipped with wearable cameras (the KidsCams) which allow them to capture videos of their everyday activities. KidsCams can be triggered either explicitly, by the children wearing them, or, automatically, through infrared (IR) beacons residing on objects (piggy back trigger) or even biometric sensory data that indicate "interestingness". Additionally, when a single KidsCam is activated, other KidsCam in close range may be also activated.

All the different episodes that are captured by a collection of KidsCams comprise a form of "hyper video", i.e., a collection of synchronised and interrelated perspectives. All the videos are transferred wirelessly to an episode management system that takes care of storing them, and of keeping track of their relationships.

Children can access the video episodes through a touch screen terminal, entitled the Magic Mirror. This allows them to collaboratively explore the video sequences and also annotate selected episodes with signs and symbols that make explicit the negotiated meanings and interpretations of what they see and experience. These annotations are used in turn to detect related episodes, or to link to any other kind of material for pedagogical purposes.

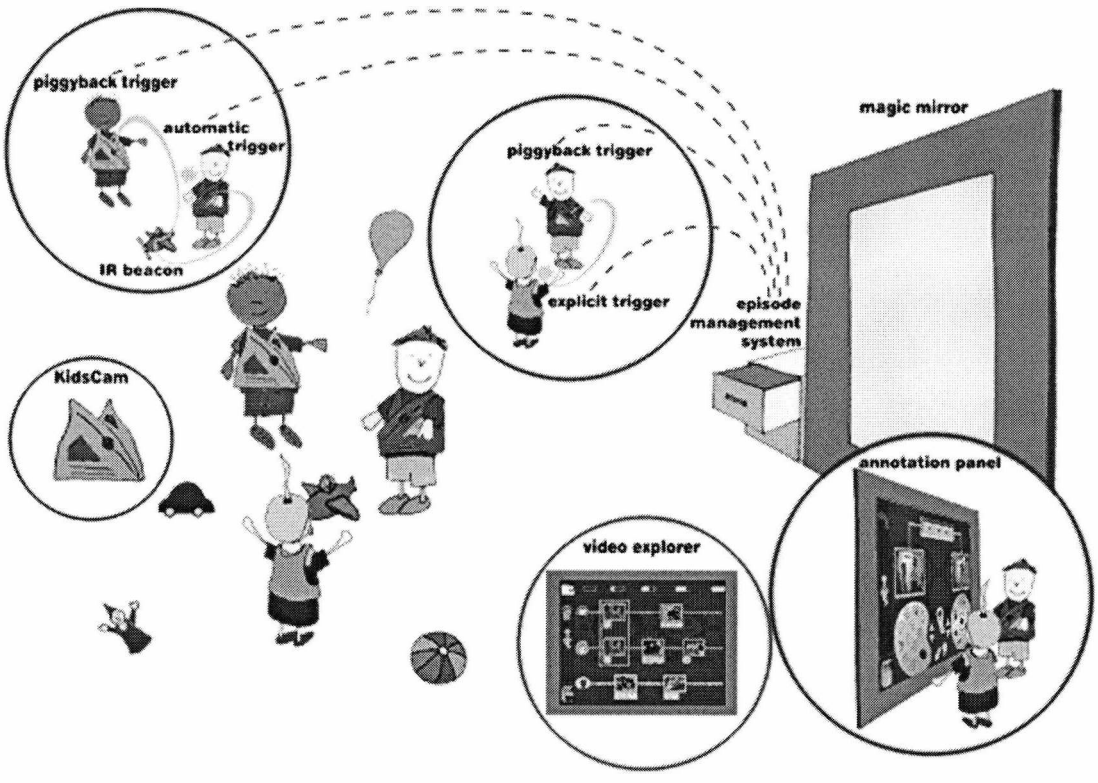


Figure 63: Today's Stories project concept overview[40]

[40] Source: http://www.stories.i3net.org/project.html

6.3. THE INTERACTION DESIGN PROCESS

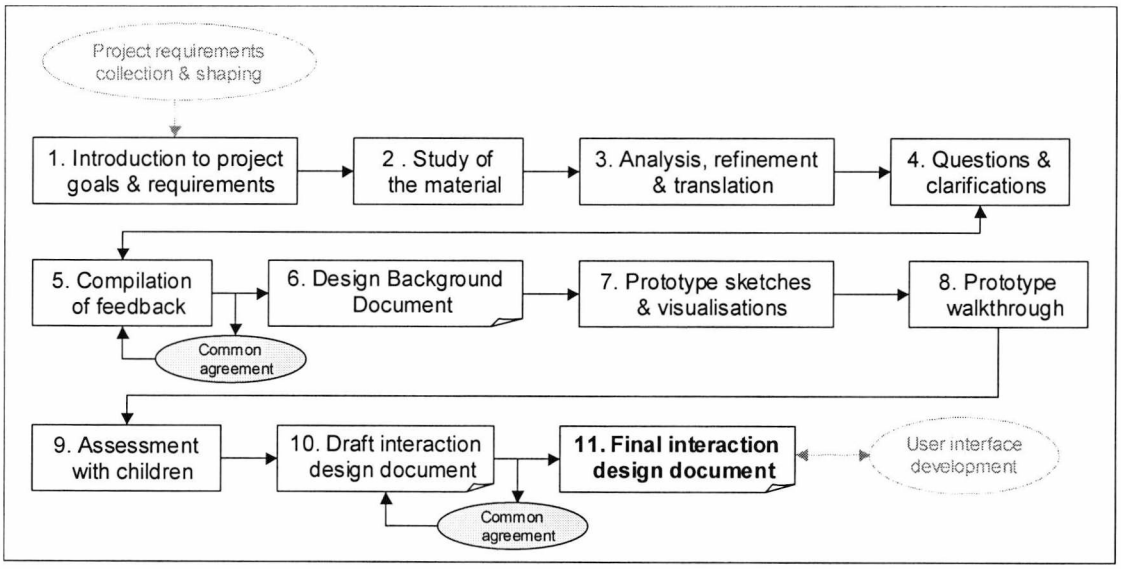


Figure 64. The interaction design process

The general process described below (Figure 64) was elaborated and followed in order to create an interaction design document describing the 'look and feel', as well as the design rationale, of the user interface to be developed:

1. **Introduction** of the design team, not directly involved in the early phases of the project (collection and shaping of requirements and constraints), to the project's concepts and goals, as well as to the requirements documents. This can be accomplished through a face-to-face meeting with some of the project members.

2. **Study of the material** related to the design task by the design team. This material can include, for example, the project's technical annex (containing additional details about its goals and the time schedule), the requirements documents, and other relevant written piece of information such as excerpts of the minutes of project meetings, e-mails, etc.

3. **Requirements analysis, refinement and translation:** This step aims at ensuring that the design team and the project members (i.e., the stakeholders) share a common (and clear) understanding of what needs to be designed (and what not). To achieve this, first of all the design team has to study and fully understand the

actual meaning of the related requirements documents, as well as the rationale with which they were created. Then, the requirements have to be "processed". This includes refining, rephrasing, 'translating' and spotting conflicting requirements but also identifying additional and missing ones, as well as requirements that are implied in, or resulting from, any of the related documents.

4. **Requirements questions and clarifications:** The result of the previous step is a long list of requirements along with related questions and points that need further clarifications. This document, which is communicated to all the stakeholders, can include three different types of questions: (a) single-answer questions, the answer to which can be found in, or deduced from, any of the related documents and are used to make the specific point clear to all, and allow the stakeholders to express their agreement of disagreement with it; (b) multiple-choice questions, for expressing preference (or indifference) towards each one of the suggested alternatives; (c) open-ended questions, to which no obvious answer can be found or suggested.

5. **Compilation of feedback:** The answers and comments received through the previous step need to be compiled and analysed. The resulting document is sent back to all the stakeholders for them to resolve any conflicts and give their approval.

6. **Design Background Document:** The previous step usually requires a few iterations between the design team and the stakeholders until common agreement is achieved. Then the "Design Background Document" can be created to include all the commonly agreed requirements and constraints.

7. **Prototype sketches & visualisations:** During this step, alternative sketches of the user interface are created, based on the "Design Background" document. For the overall approach as well as for each separate sketch, the related design rationale is recorded. The sketches can be produced in paper and / or electronically, using for example PowerPoint. Additionally, sample visualisations of these sketches are graphically designed.

8. **Prototype walkthrough:** The prototype sketches and visualisations along with their design rationale and potential interactive features are presented through a

walkthrough session to all the stakeholders. Since the stakeholders have different roles and backgrounds, several discussion topics at different levels can arise, ranging from usability issues to technical feasibility. This session results to the refinement and improvement of the prototypes.

9. **Assessment of prototypes with children:** In addition to the walkthroughs with the stakeholders, colour printouts of the sample visualisations are tested with children. The goals are to find out their attitude towards the visual characteristics of the suggested interface, assess the comprehension of the employed metaphors and identify potential usability problems.

10. **Draft interaction design document:** The previous two steps contribute towards the compilation of a draft version of the "Interaction Design Document" which includes the "Design Background Document", and a detailed description of the 'look and feel' of the user interface of the system, along with its design rationale, sketches and sample visualisations. This document is sent to all the stakeholders for comments and approval.

11. **Final interaction design document:** After a few iterations for clarifying and resolving any pending comments and issues, the final version of the "Interaction Design Document" is produced, which can be handed over to the developers of the user interface. At this point, it has to be clarified that although the interaction design document is called "final", it is by no means a fixed document. On the contrary, as the interface is constructed, it needs to be continuously refined and updated.

Although the above process might seem a little long or tedious, it has been proved to be very efficient and effective in practice. Two factors have catalytic role in this respect: *mutual understanding* and *agreement*. *Mutual understanding* means sharing a common view of what is needed or wanted, and most importantly why. This common view can be described through narratives, questions and answers, or sketches and pictures. *Mutual agreement* has to do with achieving a general consensus prior to making major steps after which any backtracking will result in considerable loss of resources.

The following sections describe the outcomes of the aforementioned interaction design process, along with issues and problems tackled during the design phase, adopted solutions and followed assumptions and conventions.

6.4. DESIGNING THE USER INTERFACE

The project's specifications (Today's Stories Consortium, 1999) required that more than one child can use collaboratively (on the same computer) the application, which should provide three distinct modes of operation:

(a) the Magic Mirror;

(b) the Video Explorer; and

(c) the Video Composer.

6.4.1. The Magic Mirror

6.4.1.1. Requirements, constraints and functional specification

The role of the Magic Mirror metaphor, according to the specifications, is to familiarise children with the system. When the system is inactive, it should emulate a mirror by displaying on its screen the video input captured through a camera positioned on top of it.

6.4.1.2. Interaction design

In the Magic Mirror, the input from a camera positioned on top of the workstation is displayed (inverted to simulate a mirror) on the screen. When one or more children approach within a predetermined distance from the Magic Mirror (this would be detected through their camera signals) the user interface should automatically switch to the 'Video Explorer' mode through a 'dissolve' effect.

6.4.2. The Video Explorer

6.4.2.1. Requirements, constraints and functional specification

During the day, children can record short video clips using wearable cameras (called the KidsCams). The KidsCams also record the time and place of the events, thus allowing the linking of video clips that are related to the same event. When one or more children holding KidsCams approach the Magic Mirror, all the video clips captured by the KidsCams in a short range should be presented. The children should be able to select, delete or send (to a friend) a video clip, or review the clips of a previous day. If more than one video clips are related to the same episode, they should be somehow linked to each other, so that a multiple view of the event is provided.

A constraint set by the project specifications (Today's Stories Consortium, 1999) is that a 'timeline' metaphor should be adopted for representing the functionality related to reviewing the video clips stored in the KidsCam. A timeline is defined as a horizontal line, spanning from the left to the right side of the screen, on which thumbnails of the video clips can be placed according to the point in time at which they were captured (Figure 65).

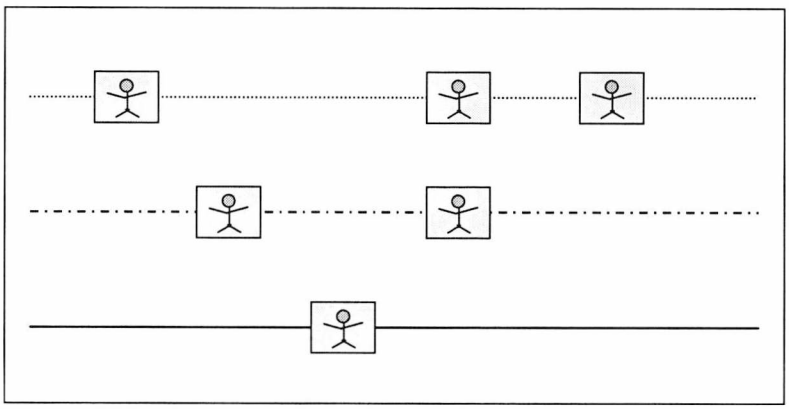


Figure 65: A sample timeline

A number of additional constraints and requirements are associated with the representation and use of the timeline metaphor (Koutra et al., 2000):

- Due to technical and complexity reasons, the maximum number of concurrently visible timelines is limited to 3. All the available video clips from a single day should be visible in the timeline window, so as to avoid back and forth movement (i.e., scrolling) in the timelines.
- Children should be allowed to delete video clips from the timeline.
- An upper limit of 4 video clips per timeline is assumed, but it is also required to have the possibility of accommodating a considerably larger number of video clips.
- If a video clip is annotated, the annotated version should be presented on the timeline instead of the original one.
- Video clips referring to the same episode should be explicitly linked and should act as a group.
- Because of the nature of the system and the particular characteristics of the target user group, a touch screen is considered as the preferable input device, but mouse (or equivalent pointing device) input should also supported, in order to enable use of the system on ordinary PCs (e.g., for home use).

6.4.2.2. Interaction design

In the light of the above-mentioned requirements and constraints, a major issue regarding the design of the Video Explorer's user interface has been identified during design, concerning the content and the meaning of the timelines. Since timelines in the 'Video Explorer' are intended to contain video clips captured within a single day, they represent a time frame of *today*. Thus, two different design options were considered:

(a) timelines always representing the same (fixed) time frame (e.g., from 8 a.m. to 2 p.m.); or

(b) timelines representing a variable time frame proportional to the period during which the separate video clips were captured.

Option (a) implies that, in the case that all the video clips were captured in a short period (e.g., between 9 a.m. and 11 a.m.) more than half of the available space is left blank (Figure 66). Since the available screen estate is quite limited and the size of video thumbnails should be maximised, the part of the timelines that have no thumbnails (the 'white space') should be minimised. Thus, option (b) has been considered as more appropriate.

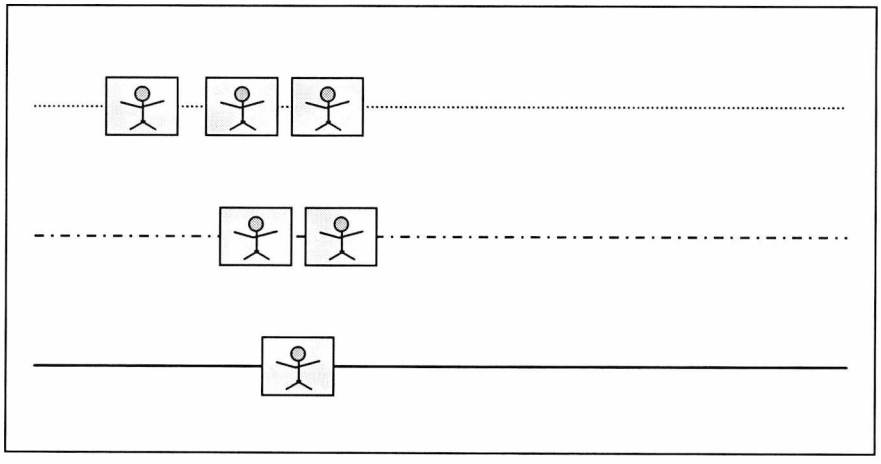


Figure 66. Fixed time-frame timelines window

An additional problem (inherent in both of the above options) stemming from the non-scrollable timelines approach is that in case the video clips of different children concentrated on different time periods (e.g., child X shot 4 video clips between 9 a.m. and 10 a.m., while child Y shot 4 video clips between 1 p.m. and 2 p.m.), it is impossible for the timelines to accommodate both the large time range (from 9 a.m. to 2 p.m.) and the large concentration of thumbnails in little screen space without overlapping of the thumbnails (Figure 67). This problem had been addressed by the Isis Story builder (Kim, 1995) through focusing on groups (*cliques*) of temporally close elements and letting the user work with only one of them at a time – but this solution required some kind of scrolling or switching between groups of thumbnails, and in the case of Today's Stories it was decided (following the project's requirements) not to support scrolling.

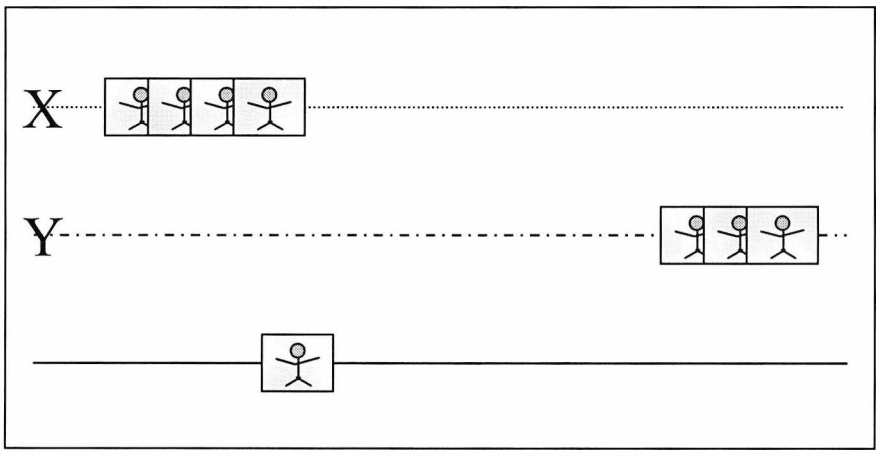


Figure 67. A problem with non-scrollable timelines

However, since the activity of video capturing is expected to be performed by children at scheduled (i.e., pre-determined) points in time (e.g., during the break, or during a visit to a museum), it is likely that the various video clips are generally concentrated over specific time intervals. Thus, it was decided that the timelines should represent a variable time frame varying according to the period in which the separate video clips were shot.

To calculate this time frame the following formula was adopted:

- Start of the timelines = a quarter of an hour before the first video clip (the time of the earliest video clip minus a quarter of an hour) – e.g., if the first video clip was shot at 8.35 a.m., then the start of the timelines would be the quarter of an hour that is before (8.35 – 15 = 8.20) which is 8.15 a.m.
- End of the timelines = a quarter of an hour after the last video clip (the time of the latest video clip plus a quarter of an hour) – e.g., if the last video clip was shot at 1.35 p.m., then the end of the timelines would be the quarter of an hour that is after (1.35 + 15 = 1.50) which is 2.00 p.m.

An analogue time-scale is then adjusted on the timelines. The reason why quarters of an hour are used is that they can be conveniently represented. Furthermore, a quarter of an hour is subtracted and added to the start and end of timelines, respectively, in order to avoid having thumbnails at any one of the ends of the timelines.

In order for the start and end time of the timelines to be easily understandable, a small clock (along with a time indication below it) was added over the start, medium and end of the timelines (Figure 68). Although this information is probably of no use for very young children, it can be quite useful to older children (or to the teachers) in order to be aware of what the timelines represent, since they change dynamically. Pre-readers can hear the time simply by touching on the clock.

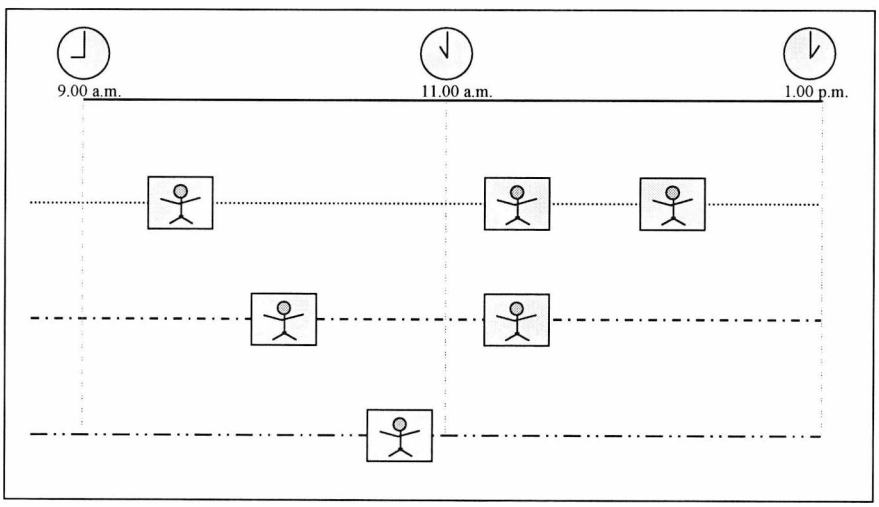


Figure 68. Representation of a time-frame

In addition to the above, it has been decided to use different colours and patterns for the three timelines, so as to allow the children to distinguish their own timeline. The same colours and patterns are used to frame the thumbnails, in order to incorporate them on the timelines and provide a coherent image, but also make it possible to distinguish the 'author' of each video clip during the annotation phase. Furthermore, a small picture of the child's face (or another picture selected by the child) is presented at the beginning of each timeline (Figure 69).

An inherent problem of the thumbnails is that although they represented a single point in time (i.e., they corresponded to a single pixel of a timeline), they have considerable size and span over several pixels. This means that if two consecutive thumbnails correspond to two close points in time, they probably overlap. A solution considered for minimising this problem is to minimise the size of the thumbnails. On the other hand, as the size of the thumbnails decreases, so does their clarity, and their ease of

use. Additionally, the size of a thumbnail can not be less than about 64 x 48 pixels, in order for it to be large enough for being selected using a finger on a touch screen. Furthermore, the screen resolution for which the application is designed is 1024x768, and a part of the screen is devoted to a 'toolbar' (see below). As a consequence of these constraints, it was not possible to place more than 10 thumbnails on the same timeline. Furthermore, the need to link together thumbnails belonging to different timelines but corresponding to the same episode (i.e., point in time) imposes an additional constraint on the layout of the thumbnails, as well as to their size, since part of the screen is devoted to visualise the linking.

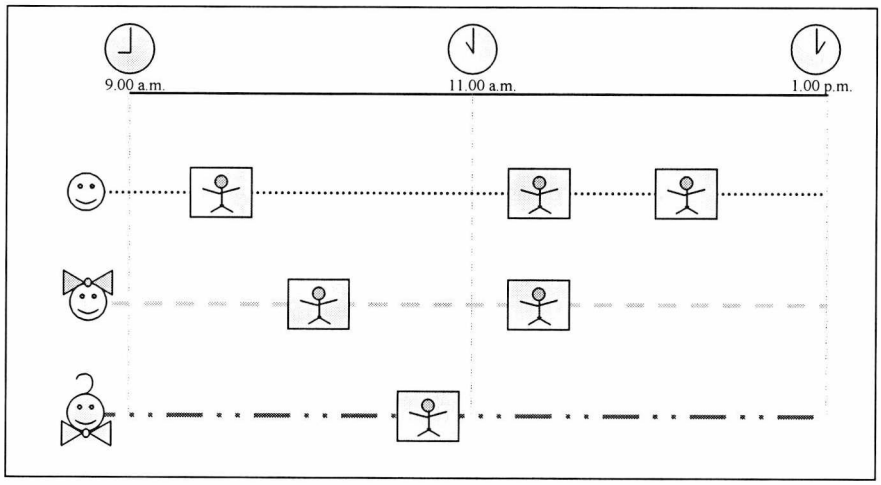


Figure 69. Distinguishing the timelines

Taking into consideration the above constraints, two different design approaches were proposed:

(a) a less general approach which has the advantage of being quite simple and easy to use and understand, and therefore more suitable for younger children;

(b) a more general approach, supporting the display of a larger number of thumbnails, but more complex, and therefore suitable for older children.

Following approach (a) some of the arising constraints concern: (i) the number of the thumbnails; (ii) the time frame during which thumbnails are captured; and (iii) thumbnails' distribution over time. Thus, (i) the maximum number of video clips transferred from a KidsCam to the Video Explorer is 4; (ii) all the video clips should

be captured in the same (limited) time frame; and (iii) all the video clips belonging to the same timeline should be captured in intervals such that their thumbnails do not overlap when positioned on the timeline.

Two or more video clips referring to the same episode are surrounded by a coloured frame with a semi-transparent background. Since there are only 3 timelines, the following 3 combinations arise (Figure 70):

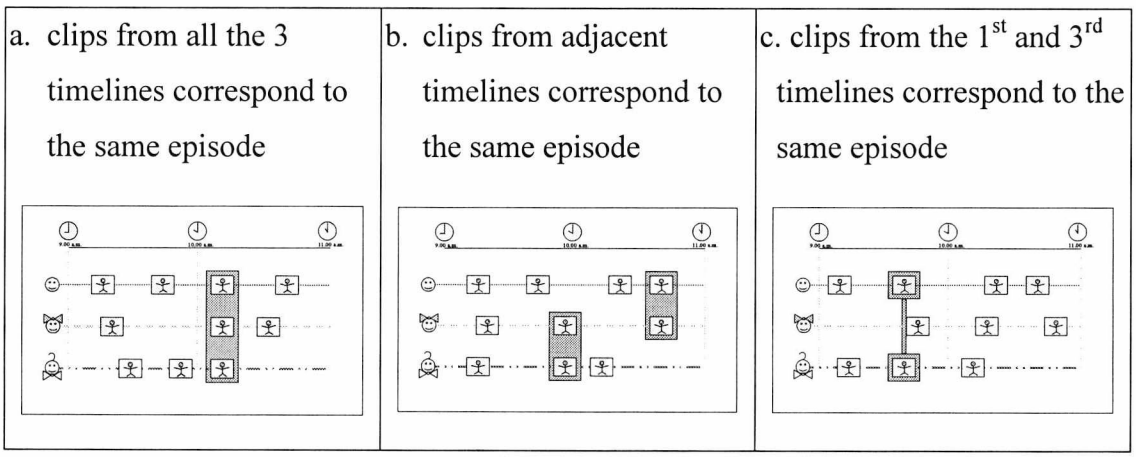


Figure 70. Alternatives for linking related video clips

As mentioned above, placing the thumbnails directly on the timeline leads to overlapping. In approach (b), the solution adopted to overcome this problem is to place the thumbnails alternatively over and under the timeline and link them to it through a line (Figure 71). Using the minimum thumbnail size (64x48 pixels), there is enough space for up to 20 thumbnails on a single timeline, an adequate number for accommodating 'advanced' use of the system. A coloured circle with a semi-transparent background surrounds two or more video clips referring to the same episode. Since there are only up to 3 timelines, the combinations that arise are presented in Figure 72.

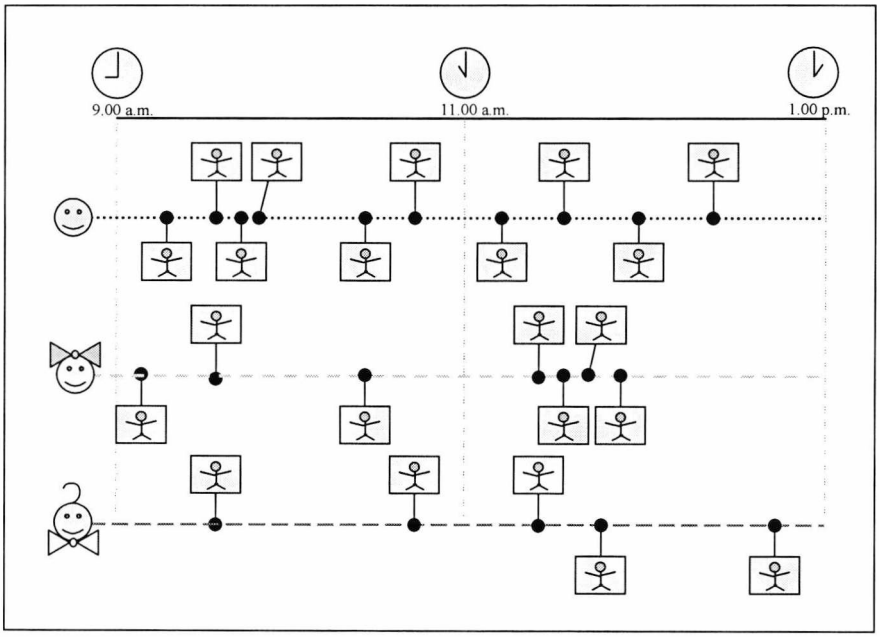


Figure 71. Linking thumbnails to timelines

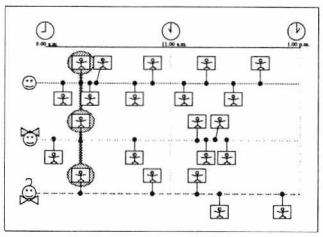
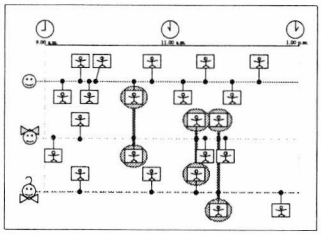
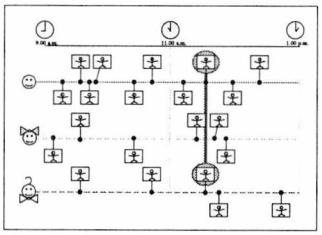
a. video clips from all the 3 timelines correspond to the same episode	b. video clips from adjacent timelines correspond to the same episode	c. video clips from the 1st and 3rd timelines correspond to the same episode
		

Figure 72: Alternatives for linking related video clips

When the minimum thumbnail size is used, a preview (or 'zoom') function is provided to help identifying thumbnails' contents: when one or more video clips are selected, a larger version is presented (Figure 73). Then, if the child presses / clicks on the zoomed video clips, the Video Composer is called, while if the child presses / clicks anywhere else, he / she returns to the Video Explorer. A coloured circle with a semi-transparent background surrounds two or more video clips referring to the same episode.

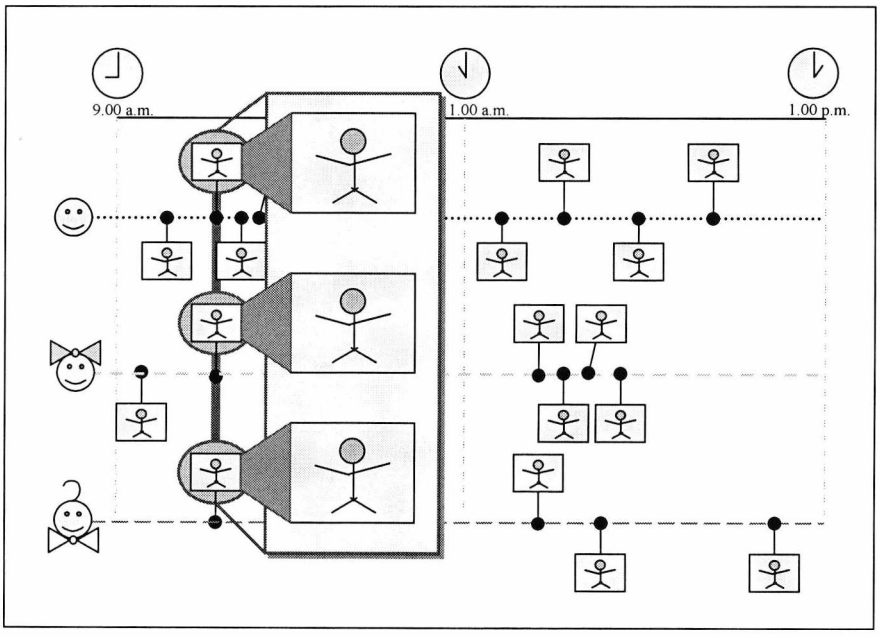


Figure 73: Zooming in the video clips

6.4.2.3. Additional functionality offered by the user interface

In addition to the above functions, children have to be able to delete a video clip (and undo the deletion), send / give it to a friend, view a previous day's 'Today's Story', and get help. All these additional functions are 'horizontal', in the sense that they could also be used in other parts of the system (e.g., in the Video Composer). In a 'traditional' window-based user interface those functions would belong to a toolbar. Because of the overall style of the designed user interface, instead of a typical toolbar, part of the screen was separated and 'populated' by interactive objects providing the above functionality. These objects are described below in Table 10.

Function	Object	Description
Get help	 *character*	An animated cartoon character was selected as a 'responsible' for this task. When pressed, the character moves around the screen presenting the available functions and explaining the interface. When dragged and dropped on a specific control / button, it explains its function.
Delete / undelete a video clip	 *trashcan*	• When a video clip is dragged and dropped on the trashcan, it is deleted. When the video clip is dragged over it, the trashcan's lid opens, the video clip is dropped, a sound effect is heard and the lid closes. Furthermore, the trashcan looks differently when it is empty and when it is full (e.g., fatter than normal). • When pressed, the trashcan 'spits back' the last deleted video clip. Multiple "undos" can also be supported.
Send / give a video clip to a friend	 *box*	When a Memory Box[41] is near the Magic Mirror, an 'open box' icon appears. If a video clip is dragged and dropped on it, it is copied to the box, and a small thumbnail appears on it. If the video clip in the box is dragged and dropped on the trashcan, it is deleted. When no Memory Box is near, an embossed icon of the Memory Box appears.
View previous 'Today's Story'	 *calendar*	When the calendar is pressed, a new screen appears presenting a calendar control, through which it is possible to select a previous day's story. This icon always has the current date written on it.

Table 10: Interaction objects populating the toolbar

[41] A Memory Box is a special I/O device developed by the project. It is a wooden box equipped with infrared that could store in it data and transmit its contents to a computer.

6.4.2.4. Sketches and prototypes of alternative user interface designs

The above steps lead to the creation of the following sketches of the Video Explorer user interface for approach (a) (Figure 74) and (b) (Figure 75) respectively:

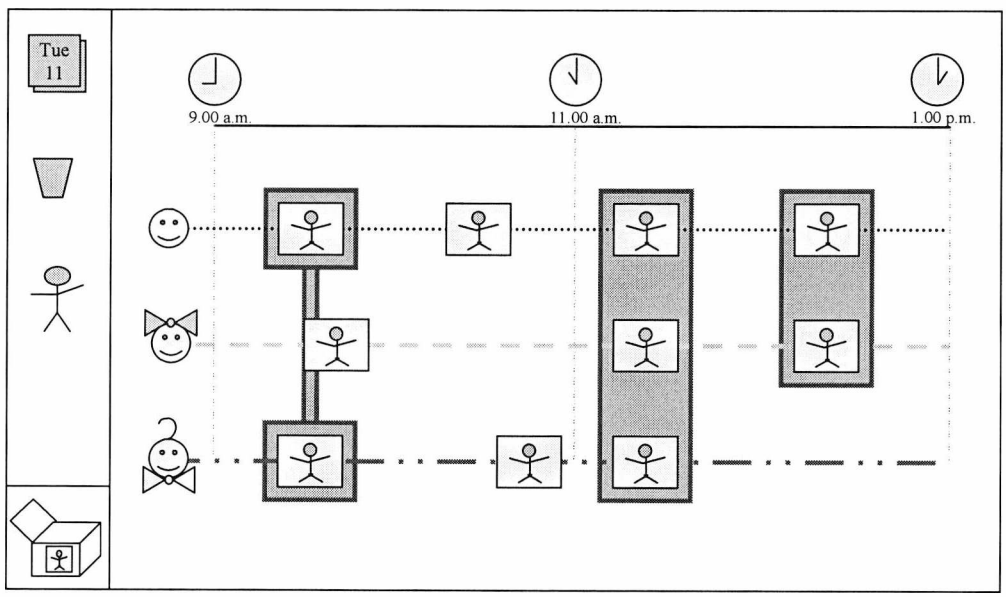


Figure 74: Sketch of the Video Explorer user interface – approach (a)

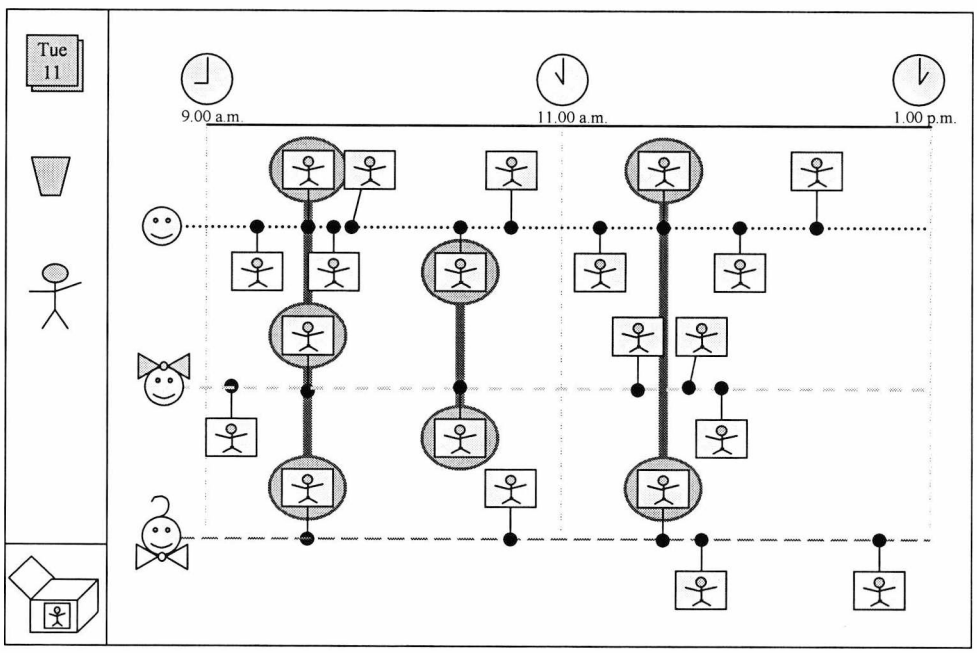


Figure 75: Sketch of the Video Explorer user interface – approach (b)

A sample visualisation of the Video Explorer user interface for approach (a) is illustrated in Figure 76:

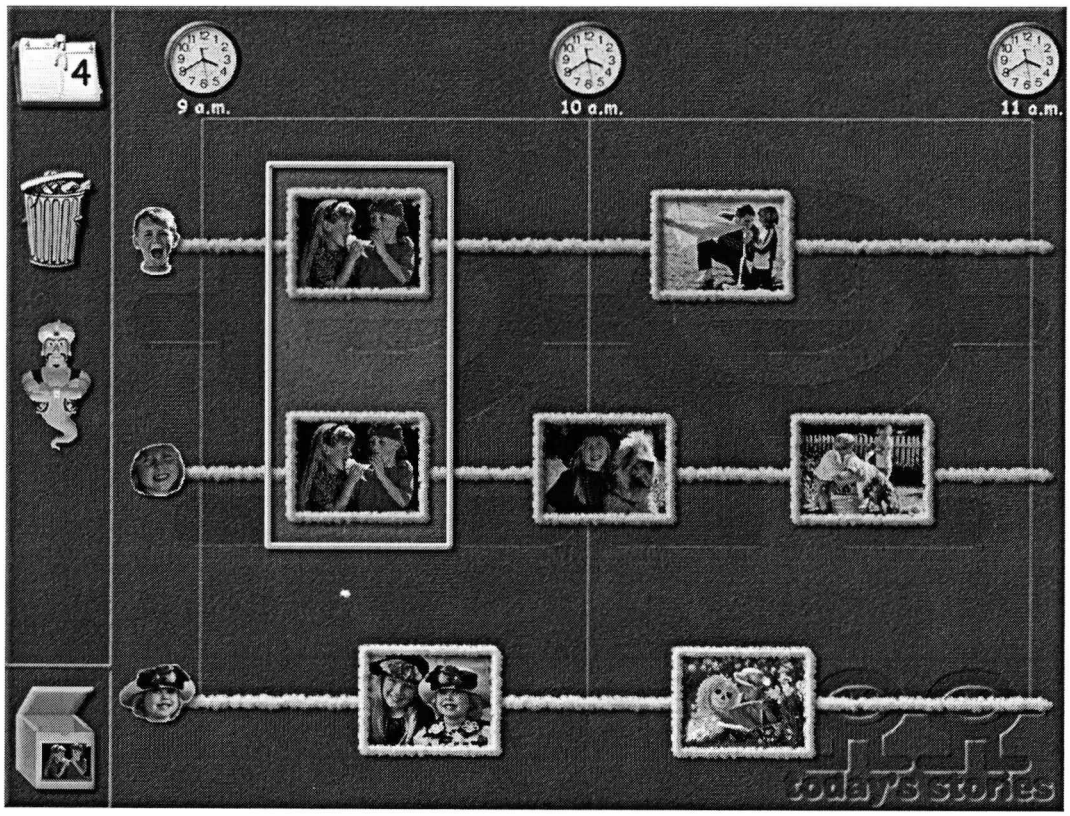


Figure 76: A sample visualisation of the Video Explorer user interface – approach (a)

6.4.3. The Video Composer

6.4.3.1. Requirements, constraints and functional specification

When children select one or more video clips from the Video Explorer, the system should allow to collaboratively control and annotate them. In this context, the first requirement is the provision of standard video control functionality (e.g., Play, Stop, Pause, Rewind, Fast Forward, Slow motion, Volume Control). If more than one video clips are concurrently selected, then they should be treated as a single entity, meaning that there should be no separate controls for each one of them. Additionally, in this case, a way for shutting down any of the video clips should be provided, so that children can focus their attention on specific video clip(s).

Regarding the annotation of video clips, three distinct types of annotations have to be supported: (i) images / symbols, (ii) sound effects and (iii) spoken comments. It was suggested that image and sound annotations are selected through some kind of 'palette'. Palettes group related annotation symbols (e.g., emotions, actions, user-defined), but the grouping should not be made explicit to the children. Each palette should have a flat structure. Furthermore, there should be no restrictions on the number of annotation symbols included in the final system. New annotations should not replace the existing ones. Any previously existing annotations should be explicitly deleted / removed by the children.

Some additional requirements include a function for removing annotation symbols (but not all of them simultaneously), the possibility of viewing the original video clip without any annotations, a mechanism for extending the existing collections annotation symbols / sounds, and the provision of on-line, context-sensitive help for all tasks.

6.4.3.2. Interaction design

First of all, two different options have been considered concerning the video control functions: (i) supporting all the functions available by common video editing programs (e.g., Play, Play Backwards, Stop, Pause, Rewind, Fast Forward, Slow Backward & Forward), and (ii) reducing the set of functions for the sake of simplicity and visual clarity. The minimum functions opted for to facilitate the annotation process are: Play (forward & backward), Stop (in the sense of 'pause'), and Fast (forward and backward). In case of use by very young children, available functions are further reduced to Play (forward & backward) and Stop.

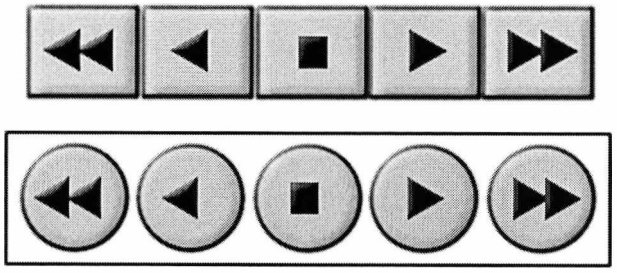


Figure 77: Examples of rectangural and round buttons for video control

It has also been decided to represent the selected video functions by conventional video controls (Figure 77), since most children, even from a very young age, are quite familiar with them. An approach suitable for very young children is to represent the same actions through relevant graphical metaphors, such as using a hare for fast-forward, a turtle for slow-forward, etc. In order to provide redundant ways of interaction with, and allow for the direct manipulation of the user interface interactive objects, it has also been decided to allow starting and stopping a video clip simply by touching or clicking on it, in addition to using the control buttons. With more than one video clips open, only a single manifestation of the video buttons is visible (and not one for each video clip), since, according to the project requirements, they should all be treated as one. The buttons are placed on a 'control box' (Figure 78).

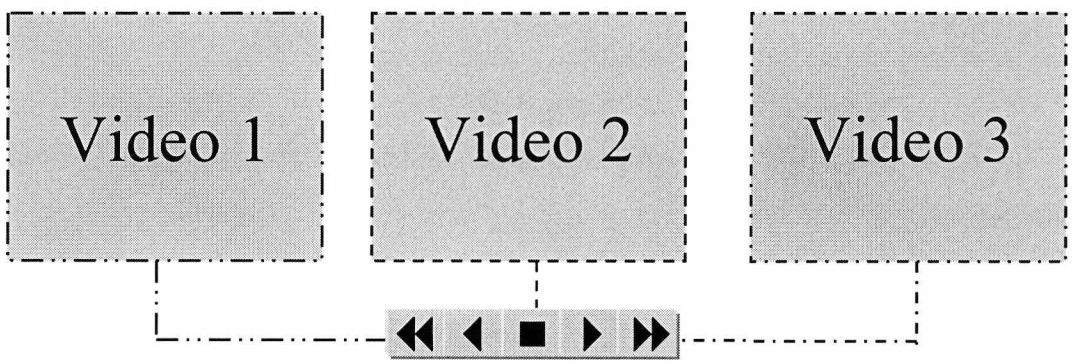


Figure 78: Example of video control box

Volume control can be performed through the interaction object presented in Figure 79. Pressing the '+' and '-' buttons increases or decreases the volume respectively, and plays a sample (short) tone. Pressing the 'speaker' button plays a sample sentence. Additional visual feedback is provided through increasing / decreasing the size of the speaker.

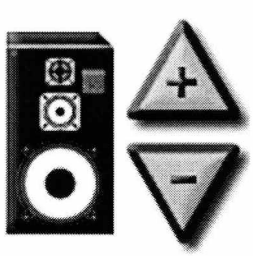


Figure 79: Volume control object

To allow children to focus on just one or two of the video clips, a special 'handle' is attached below each video clip. When this handle is pressed or clicked, the video clip shrinks to a thumbnail, which also includes this special handle (see Figure 80). The thumbnail is just a 'still' picture and thus is not affected by the use of the video controls. When the thumbnail or its 'handle' is pressed or clicked, it 'grows' back to its normal size and functionality. This function is omitted when the system is used by younger children.

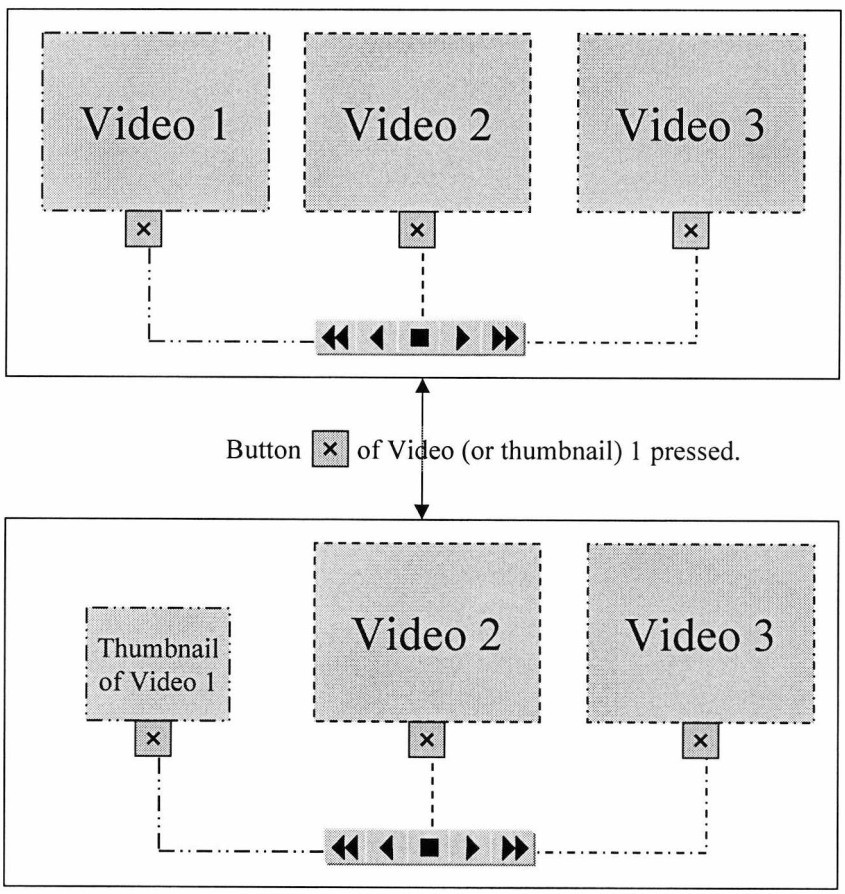


Figure 80: Closing and opening a video clip

The image palette supporting the annotation process (Figure 81) has been designed to have the 'look and 'feel' of a collection of stickers (e.g., when an image is selected an 'un-stick' visual and sound effect is produced), since most of the children are quite familiar with stickers and can intuitively understand their purpose and functionality. When one of these pictures is selected, a short description or sound is played.

Figure 81: Image annotation palette

The sound palette was intentionally designed to look significantly different from the image palette, in order to avoid confusion. Actually, it looks like a CD that has pictures representing sound effects on it. These pictures have a different look than those of the image annotation palette; they all include a small CD icon (Figure 82). When one of these pictures is selected, the respective sound effect is played, along with a visual effect of the picture sticking to the mouse pointer (or finger). Since image and sound annotations should not be added while the video clip is playing, the respective palettes are (and look) inactive during that period and become activated as soon as the video clip are stopped.

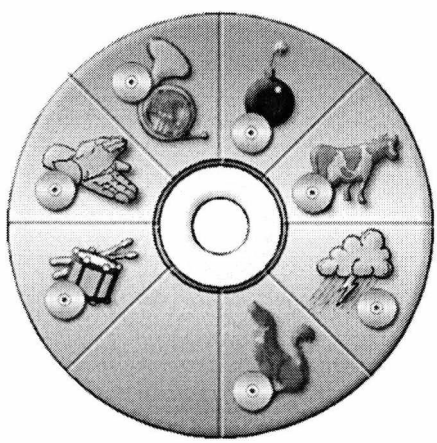


Figure 82: Sound annotation palette

In addition to the above, as a high number of annotation symbols and multiple annotation palettes have to be supported through an easy and direct approach, a set of previous / next buttons has been added that reside close to each palette and enable the users to browse through the available palettes (see Figure 83). When one of these buttons is pressed, a new palette slides on top of the old one. Offering the option of using alternative annotation palettes and symbols can be quite important since "*the use of varieties of sign systems can enhance still further our understanding of the constructedness of knowledge, the value of considering multiple perspectives*" (Knuth & Cunningham, 1993). This functionality is omitted when the system is used by younger children.

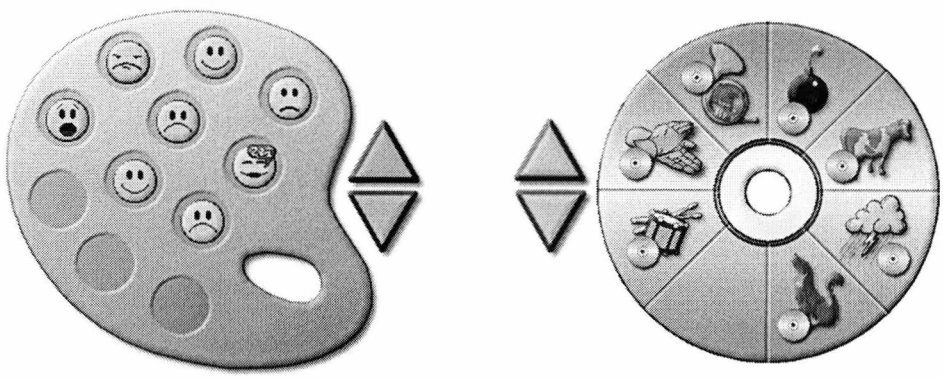


Figure 83: Examples of annotation palettes with previous / next buttons

The annotation palettes can be easily customized by the children, who can fill them with their own items (images and sounds) by using a Memory Box. If the content of a Memory Box is an annotation item, it can be dragged and dropped on an empty position of a (relevant) annotation palette, so as to be added on it. The palettes include empty positions by default, and additionally the children can create more empty positions by deleting existing items. Children can also remove palette items by dragging and dropping them to the trashcan. Un-deleting is supported through pressing / clicking the trashcan.

The final 'look and feel' of the palettes, as well as the annotation symbols presented on them, would be shaped collaboratively with children, to appeal to their individual preferences but also to their cultural background. Some suggestions for annotation

symbols can be found in (Dreyfuss, 1984; Frutiger & Bluhm, 1998; Horton, 1994; Modley et al., 1977).

To add an image or sound annotation, first the video clip has to be stopped. Then:

a) To add an image annotation the child drags an image from the annotation palette and drops it anywhere on one of the video clips. The child can then move the annotation symbol and change its position on the video clip.

b) To remove an image annotation the child drags it from the video clip it resides upon and drops it anywhere outside it. In case it drops it on another video clip, the annotation is transferred to it.

Sound annotations work in a similar way, with the difference that when the annotation symbol is pressed, the sound effect represented by the symbol is played instead of an 'un-stick' sound effect.

Speech annotations are treated differently. Firstly, a microphone control is used instead of a palette. The microphone control is animated (small 'beams' of sound would appear) when active, to provide relevant feedback. Secondly, there are two alternative ways for inserting voice annotations:

a) 'Running' voice annotation (i.e., add to the clip a running commentary, in the same way that a sports caster describes a football game). The annotation starts and ends by pressing the microphone button. When the video clip is played back, the annotation is played along with it, substituting its original sound. This type of annotation is represented on the video clip by means of a microphone symbol.

b) 'Still' voice annotation: First, the video has to be stopped. Then, the microphone button has to be pressed. The annotation ends by either pressing the microphone button again or starting the video clip. When the video clip is played back, it (automatically) stops at the annotated point, the comment is played (along with some feedback that the specific annotation was played – e.g., the annotation symbol flashes) and then the video clip resumes. This type of annotation is represented on the video clip by means of a microphone symbol with a small stop button on it.

In addition to the above, children can switch the presentation of annotations on and off, through a two-state button (see Figure 84). To accommodate any additional functionality a toolbar is used, which is quite similar to that of the Video Explorer. The only difference is that the calendar control is replaced by a control for returning to the Video Explorer.

State 1	When pressed, annotation palettes become inactive (and / or hidden) and video annotations are switched off. The button changes to *State 2*.	*State 2*	When pressed, annotation palettes become active (and visible) and video annotations are switched on. The button changes to *State 1*.

Figure 84: Annotation switching button

6.4.3.3. Sketches and prototypes of alternative user interface designs

The above analysis and discussion lead to the creation of the following (first) sketch (Figure 85) of the Video Composer user interface:

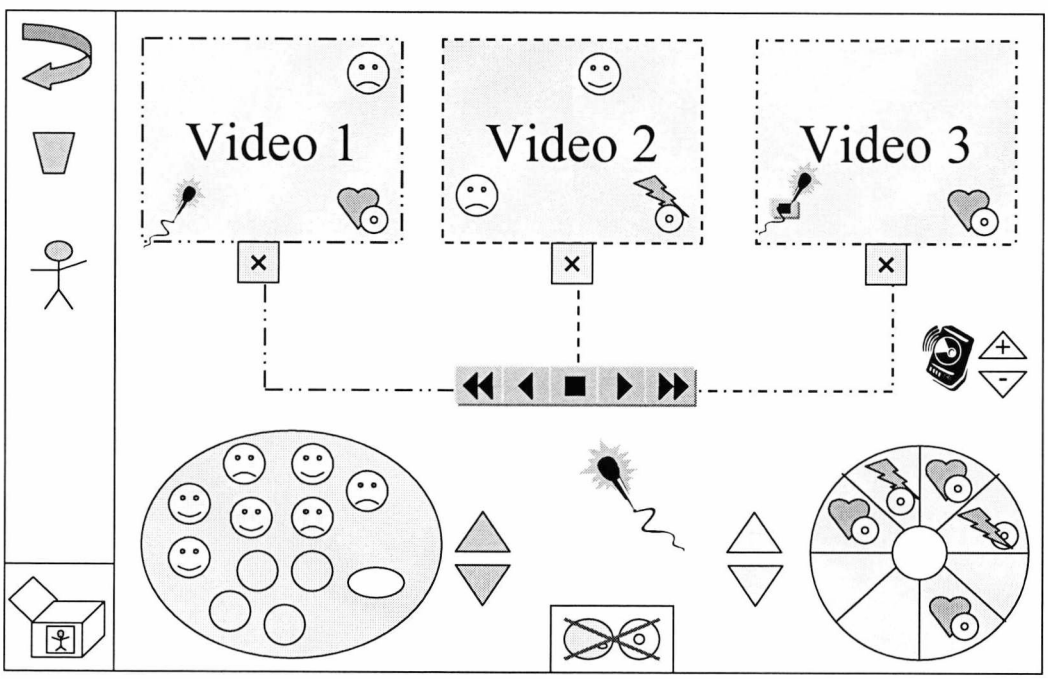


Figure 85: A first sketch of the Video Composer interface

An expert usability evaluation (Nielsen & Mack, 1994) of the above user interface sketch revealed two possible drawbacks:

(a) the annotation palettes are too far from the video clips;

(b) the video control and the open / close buttons are 'mixed' with the palettes.

These drawbacks were addressed by reversing the position of the video clips and their controls (Figure 86). As a result, the video clips provide a physical barrier that separates (and groups) the video control buttons from the annotation functions. A sample visualisation of the revised Video Composer user interface is presented in Figure 87.

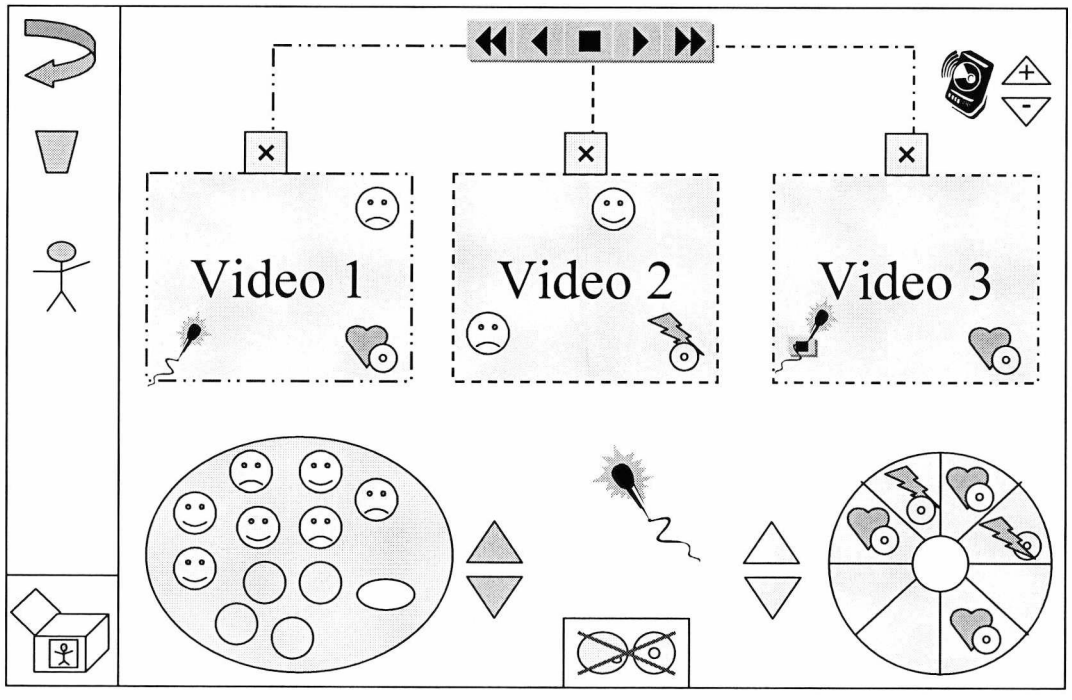


Figure 86: Revised sketch of the interface

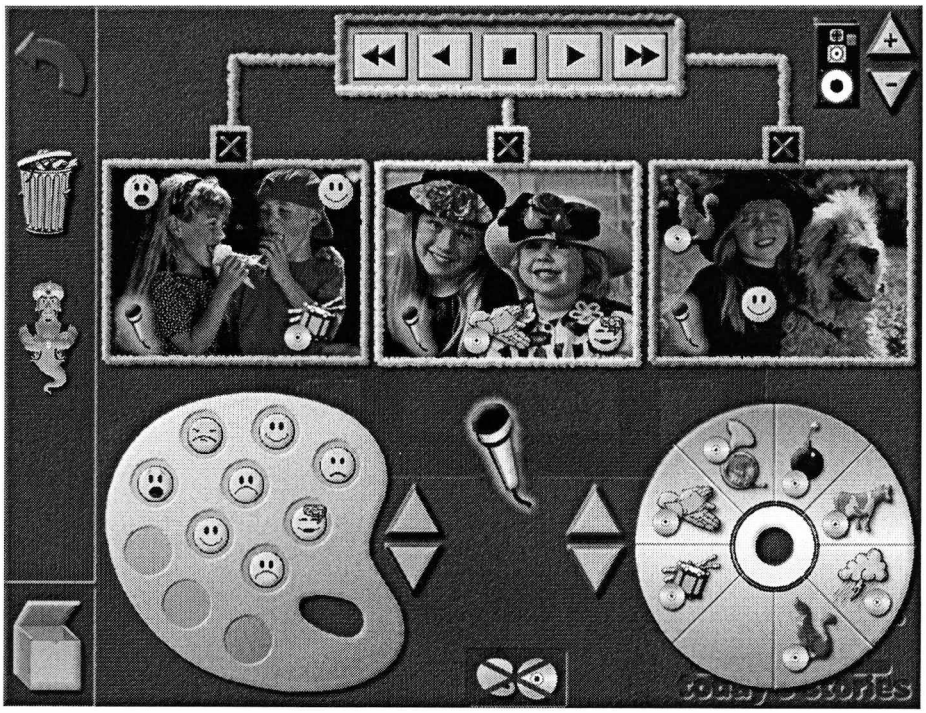


Figure 87: A sample visualisation of the revised Video Composer user interface

Since the Video Composer provides some complex functionality that might overwhelm or perplex very young children during their first contacts with the system, a simpler version of the user interface, containing only the 'bare necessities' for the video annotation task, was also designed (Figure 88).

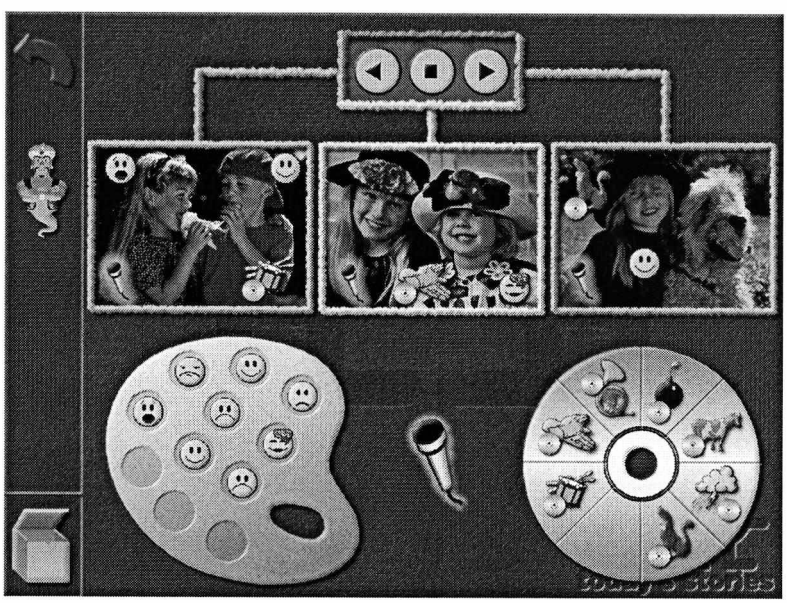


Figure 88: Alternative version of the Video Composer user interface for very young children

6.5. EVALUATION

The outcomes of the work reported in this chapter were used to create prototypes which, following the principles of User-Centred Design (Norman & Draper, 1986), were tested with, and evaluated by, actual users (i.e., children) (Hanna et al., 1997). These activities lead to refinements and elaboration of the design and overall improvement of the user interface. Furthermore, they helped to ensure that the system behaved as it was expected to, and to assess whether the requirements and goals set during the requirements phase have been met. The evaluation of the Diary Composer user interface is reported in, Hansen et al. (2002) and will not be discussed in details here. Overall, it appears that (even very young) children found the interface visually attractive ("smart and fancy") and at the same time very easy and fun to use. Feedback received from the evaluators in the form of discussions, written comments and requests for changes or extensions to the interaction design, lead to re-design of some aspects of the user interface.

First of all, a problem spotted for the Video Explorer interface was that the concept of using actual clocks to represent "real" time over the timelines (see Figure 76) was found to be difficult to understand, or totally ignored by most of the children. So it was decided to replace the clocks with horizontal fill-in bars representing a relative percentage of time (start, a quarter, half, etc.) in relation to the total time elapsed between the first and last video clip (see the top of Figure 89).

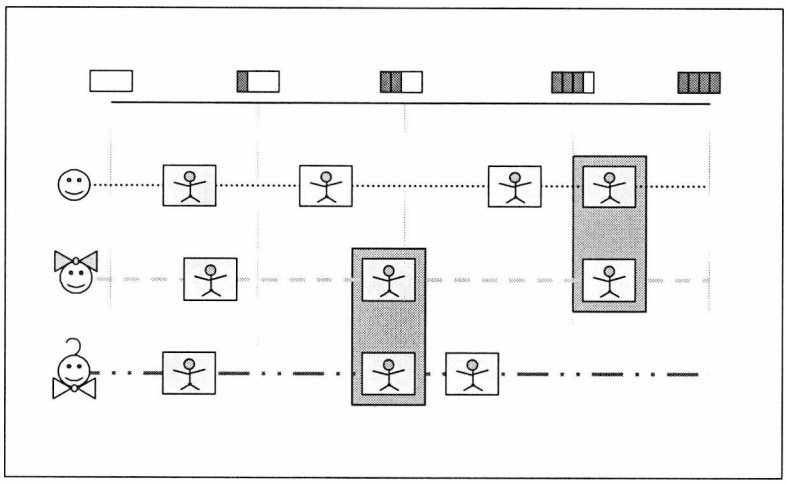


Figure 89: Using fill-in bars to represent time

Another drawback was that children were not able to tell if they had already annotated a video clip or not. To overcome this, three types of icons were created which can be added below a video clip and denote the type of annotations that the video clip includes: a small microphone indicating voice annotation, a small palette indicating image annotation, and a small disc indicating sound annotation (Figure 90). The same approach was also followed for the clips presented in the Video Composer interface.

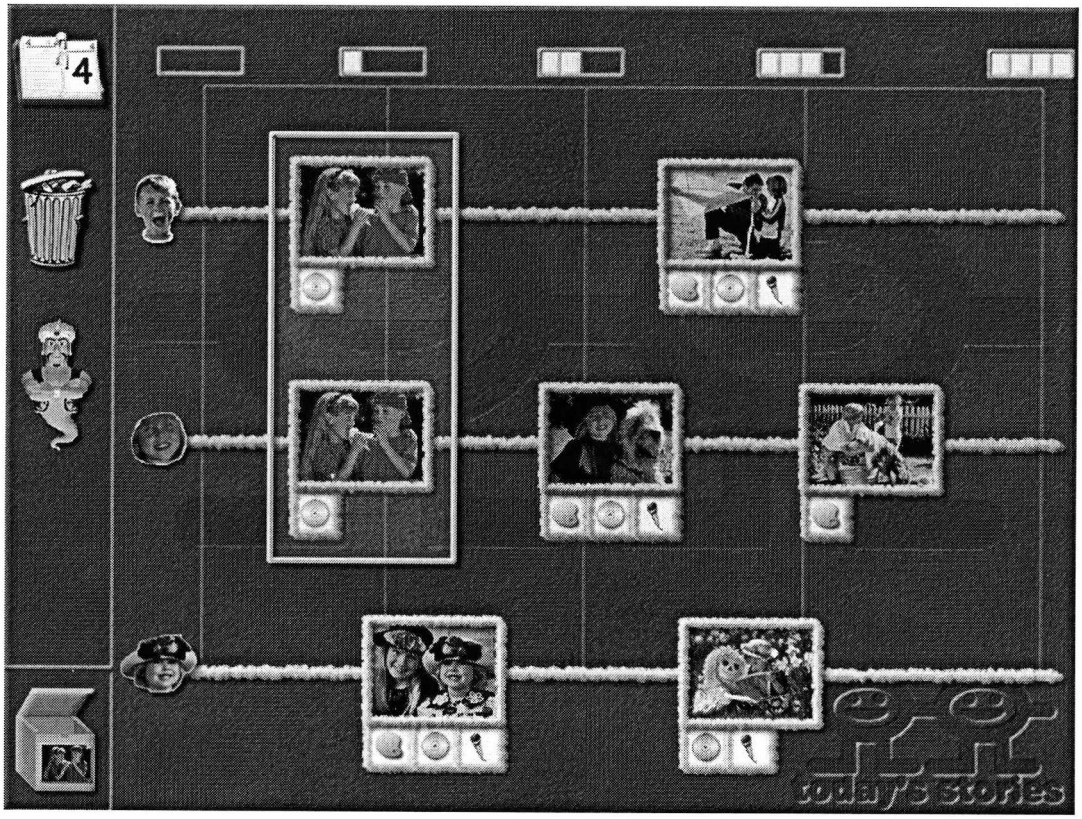


Figure 90: Visualizing the type of annotations made on each video clip

Regarding the Video Composer interface, the following observations and changes were made (Today's Stories Consortium, 2000):

(a) Observations have shown that children often need to use the same annotation more than once. So, when children drag an annotation from the palette, a copy of the annotation remains on it.

(b) Sound feedback is required for all the annotation actions since observation and discussion have shown that it is very important. When an image annotation is

selected, a sound is played, as well as when the annotation is dropped on a video clip. Similarly, when a sound annotation is selected, the relevant sound effect is played and also when the annotation is dropped on a video clip.

(c) Sometimes confusion was caused by trying to use the annotation palette while the video was playing. So, in this case, the palettes should be disabled.

(d) There is a need for visual cues that help the children understand the results of their actions and also notify them when an annotation 'fails'. So, when a child drags an annotation off the palette, it gets highlighted according to the following scheme (see Figure 91): (i) if the annotation is located over a video clip (which means that if dropped it will be added to it) then it has a green highlight; (ii) if the annotation is located over anything else (which means that if dropped it will return to the palette) then it has a red highlight. Also, when the annotation cannot be used, it is greyed out.

(e) A button is required for erasing in one step all the annotations made on a clip (Figure 92), as well as an undo button (named "Oops!") (Figure 93) for compensating the accidental use of such "massively destructive" actions.

(f) The play button must become a two-state button, i.e., while the video is playing this button should be pressed down, since the evaluation showed that confusion is likely to be caused if the play button is pressed again and again by the children.

(g) The tests showed that it is more practical for young children to go back to the beginning and view again a short video clip (not more than 2-3 minutes) instead of trying to find and 'freeze' a certain moment of the video-clip. Thus, the play-backwards button is replaced by a "go-to-the-beginning" button.

State	Visualisation
1. No interaction	
2. Dropped on a video clip	
3. Dropped *anywhere else*	
4. Disabled	

Figure 91: Different states of an annotation symbol

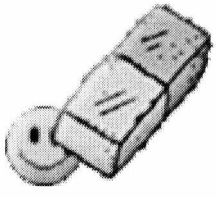


Figure 92: The "erase all annotations" button

Figure 93: The "Oops!" button (i.e., undo)

Finally, in the implemented interactive prototype the time it took to get from the Video Explorer to the Video Composer interface was considerable and sometimes children lost interest or got distracted. A solution to this was the creation of an animated graphic providing feedback that the system is currently loading the video clips (Figure 94). This animated graphic attracts the users' attention for as long as loading takes place and gives them a hint on what is going on while they are waiting.

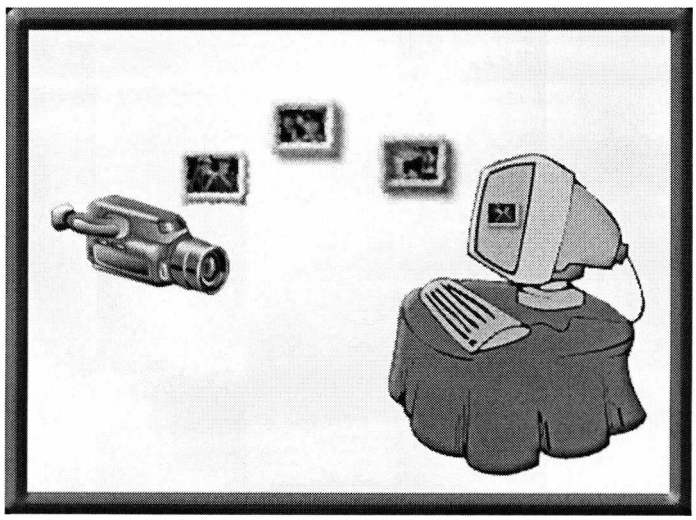


Figure 94: Providing feedback that the system is busy

6.6. DISCUSSION

This chapter reported on the process followed for the interaction design of the user interface of the Diary Composer software of the Today's Stories project. Furthermore, the outcomes of this process were presented. Since the user interface was intended for use by very young children, the design relied highly upon visual rather than textual interaction. The resulting environment was intended to be aesthetically pleasing and appealing to children, while, at the same time, it aimed to support simple, intuitive interaction through the adoption of adequate interaction techniques, the use of suitable metaphors to represent system functions, and the continuous provision of visual and audio feedback. Furthermore, it provided for customisation to the needs, requirements and preferences of the children, promoting self-expression and individuality. The software addresses the involvement of different senses of the child: visual, audio, touch, and trigger a variety of experiences: classifying, spontaneous reaction, search for meaning, creating annotations – sound, image, text.

The challenges faced in this chapter had to do with designing the user interface of a non-typical application [C2] (including the creation of new metaphors [C1]) for a non-typical user group [C3] and more specifically designing an interaction environment, suitable for young children (4-8 years), empowering them to reflect on

recorded events of their daily lives, presented through alternative perspectives and supporting collaborative viewing and annotation. The related outcomes included the account of a design process for the creation of a multi-perspective collaborative application for young children, as well as the resulting design concepts, metaphors and solutions, along with their design rationale and the related issues.

Chapter VII

SUMMARY, CONCLUSIONS AND FUTURE WORK

7.1. SUMMARY AND CONCLUSIONS

This thesis focussed on creating new knowledge and providing concrete solutions for addressing a number of interweaved challenges that interaction designers of emerging everyday computer-based systems (ECS) are facing. These challenges are:

[C1] Creating novel interaction metaphors and techniques.

[C2] Designing for non-typical interaction paradigms.

[C3] Designing for non-typical user groups.

[C4] Designing for accessibility.

[C5] Designing based on incomplete knowledge.

The related research has produced several results that contribute in different interaction domains and at different levels. More specifically, four interaction domains were addressed, namely: (i) Virtual Environments (VEs); (ii) Accessible Computer Games; (iii) Software Agents; and (iv), Interactive Applications for Young Children.

For each of these domains a key research issue reflecting one or more of the aforementioned challenges was selected and addressed. More specifically, for each domain, the related challenges and key research issue were the following:

i) Virtual Environments

Design challenges: Creating a novel, intuitive, interaction metaphor [C1] for a non-typical interaction domain [C2].

Key research issue: Developing a new interaction concept for supporting navigation in Virtual Environments.

ii) Accessible Computer Games

Design challenges: Designing for accessibility [C4] (and more specifically for Universal Access) and also designing based on incomplete knowledge [C5].

Key research issue: Creating universally accessible games, which are proactively designed to optimally fit and adapt to different individual gamer characteristics and (dis)abilities without the need of further adjustments or developments.

iii) Software Agents

Design challenges: Extending the capabilities of a novel interaction paradigm [C1] providing higher design flexibility and more options to designers and also designing for openness and extensibility [C5].

Key research issue: Developing a generic approach to creating a synthetic sensory system, applicable to any type of embodied agent in any type of virtual world, independent of the nature or the intrinsic characteristics of the senses or, stimuli simulated.

iv) Interactive Applications for Young Children:

Design challenges: Designing the user interface of a non-typical application [C2] (including the creation of new metaphors [C1]) for a non-typical user group [C3].

Key research issue: Designing an interaction environment, suitable for young children (4-8 years), empowering them to reflect on recorded events of their daily lives, presented through alternative perspectives and supporting collaborative viewing and annotation.

In this context, the main outcomes of this thesis were the following:

- A novel intuitive interaction metaphor (Virtual Prints) for supporting navigation, orientation, way-finding, as well as a number of additional functions in Virtual Environments and the process followed towards defining and refining it.
- A fully-functional universally accessible Web-based board game, the design of which was based on the principles of Design for All, and a detailed account of how accessibility is supported for different user categories through the game's

interface, its adaptation capabilities and the available alternative input and output modalities. The game is publicly available on the Web and was nominated for the final jury decision of the European Design for All Awards set by the European Commission, in the category "AT/Culture, Leisure and Sport".

- A generic model for creating an artificial sensory system intended to be used for the creation of intelligent software agents which exhibits openness, scalability, reusability, extensibility, simplicity and computational efficiency.
- A design process for the creation of a multi-perspective collaborative application for young children, as well as the resulting design concepts, metaphors and solutions, along with their design rationale and the related issues.

The above outcomes have appeared in several publications and also led to two undergraduate seminars on "Artificial Intelligence, Software Agents and Sensing" at the Department of Computer Science of the University of Crete during the Fall semester 2004 and to a tutorial entitled "Design and Implementation of Universally Accessible Computer Games", which will be delivered in the context of the 11th International Conference on Human-Computer Interaction (HCII 2005) in Las Vegas.

7.1.1. Designing emerging everyday computer-based systems

Beyond the specific contribution in each interaction domain, some more general conclusions can be derived concerning interaction design for ECS in order to address the related challenges. These conclusions are provided below.

[C1] Creating novel interaction metaphors and techniques.

They key attributes when creating a new interaction metaphor or, technique can be summarised as follows:

(a) *Reusability*: a metaphor should be self-contained so that it can be directly reused it in other cases / projects.

(b) *Scalability and flexibility*: a metaphor should be scalable and flexible regarding its sophistication, interaction complexity and capabilities and required resources

(e.g., computational power, memory, screen estate) so that it can be adjusted to different user and system requirements.

(c) *Openness and extensibility*: a metaphor should be open and extensible, so that functionality and interaction characteristics can be added / removed on demand.

(d) *Abstraction and generality*: ideally, a metaphor should be independent of the specific characteristics of the environment it is instantiated, or, the devices used to interact with it but it should be possible to adapt to these.

[C2] Designing for non-typical interaction paradigms.

This case usually also requires the design of new metaphors and concepts. Typically, there are no ready-to-use interface solutions or components and the related technology is rather unstable, creating several interaction barriers. Design heavily resides on rapid prototyping in different forms, ranging from simple sketches to electronic mockups, and quick informal evaluation experiments with users. An intangible but invaluable resource for the interaction design is imagination, since it is very probable that unique situations and problems may arise where no pre-existing solutions or references are available and where users cannot provide any assistance.

[C3] Designing for non-typical user groups.

When designing for user groups that the designer is not part of, the only really safe and adequate methods are user-centred and participatory design, where representatives of the user group are part of the design team. Assumptions about third-party observations are not reliable and may be totally misleading (e.g., about why someone likes, dislikes or has a problem about a particular feature of an interface). Meticulously studying the physical and cognitive characteristics of these groups can provide a helpful insight and act as a good starting point for getting good design ideas. Of course, related work and bibliography should also be consulted.

[C4] Designing for accessibility.

Designing for accessibility requires:

- Designing for *context independence*. This means designing at an abstract level without considering specific interaction modalities, metaphors, techniques or devices. A basic approach for achieving this is to separate the content and the related mechanics from the way that these can be accessed by, and presented to, the user. For example, in the case of a chess game, the outcomes of a context independent design may include: the board and its attributes, the pieces and their moves, the game rules, the possible actions (or tasks) that players may wish to make and the feedback information they should receive about it.
- Being able to map the abstract design elements to coherent, usable and accessible interaction designs based on the users' individual characteristics. For example, in the case of a board game like "Snakes & Ladders", the presentation of the board maybe translated to a highly complex 3D graphic for a sighted player and to a linear series of squares announced orally for a blind player. In the latter case, the blind player should also be provided with access to additional information (e.g., about the contents of a particular square, the opponents position).
- Creating user interfaces that can *support alternative interaction methods and modalities* that can co-exist and co-operate. So, for example, there can be various input methods that can be used to select and move a piece (e.g., with a mouse, the keyboard, voice commands) and also if players start performing an action through any one of them, they should be able to complete it using any other. Of course sometimes there can be cases where two methods may be incompatible, and this should be foreseen and avoided.
- Creating user interfaces that are able to adapt to alternative user profiles, i.e., sets of preferences, requirements and needs and contexts of use. In case two or more people are sharing the same computer in a sequential manner (e.g., a turn-based game), the interface should change accordingly.
- *Inclusive design*, i.e., considering the broadest possible population during the design phase.

[C5] Designing based on incomplete knowledge.

As one the one hand a very large percentage of the software being developed is Web-based, and, on the other hand, computer applications and products target highly diverse audiences, it is very unlikely that designers can have a complete view of all the attributes and details of the design space they should consider. The only way to overcome this problem is *open and extensible interaction design* so that, later on, it will be still possible to expand the design so that it can cater for more user categories and contexts of use, e.g., by supporting additional interaction methods, techniques or devices. A prerequisite for achieving this is designing for *context independence*, that was referred above.

7.2. FUTURE WORK

Future work related to this thesis can be divided in two categories:

(a) Work related to further extending and elaborating the individual contributions made to the selected key research issues and domains.

(b) Work related to the overall theme and goals of the thesis.

Concerning the first category (a), future work shall seek to:

- Further develop the software instantiation of the Virtual Prints metaphor (described in Chapter III), integrate it into diverse VE applications, and assess its impact on the usability of such environments in practice.
- Contribute to the evolving research domain concerned with the evaluation VEs, by testing and evolving related structured processes.
- Apply and experiment with the concept of Virtual Prints in Augmented Reality applications.
- Transfer and apply the know-how, methods and techniques developed for creating universally accessible games (Chapter IV), in game genres that are more interaction-demanding than board games, such as, for example, action games.

- Extend the capabilities of the developed artificial sensory system model (Chapter V), by providing alternative integrated options and methods for supporting sensory occlusion.
- Port the module for creating the sensory system model to a development platform that can provide appropriate means for maximising speed and efficiency, such as, for example, C++, and integrate it in a real-time interactive game.
- Create direct manipulation tools targeted to interaction designers for visually defining and adding sensory capabilities to embodied intelligent agents and also managing all the related configuration parameters.

Concerning the overall goals of this thesis, ongoing and future work aims to integrate the research outcomes towards the development of novel state of the art applications that exhibit a combination of innovative characteristics, such as:

- they are targeted to non-typical interaction domains;
- they are concurrently targeted to several non-typical user groups;
- they employ new metaphors and interaction concepts;
- they are universally accessible.

In this direction, current research efforts focus on two such applications:

(a) A multi-lingual context-aware mobile virtual guide, designed to be used by *anyone*, including children, non-computer users, illiterate, disabled and elderly in indoor (e.g., museum) and outdoor (e.g., archaeological site) settings. The system shall use adequate positioning technologies to infer the user's position and based on this, as well as on the user's profile, will deliver customized information (e.g., about nearby exhibited articles, the user's location and path, suggestions for) in a suitable and usable way for each user.

(b) A universally accessible multi-player action game for both children and adults, the interactive characteristics, presentation and gameplay of which can be adapted to the characteristics, preferences and (dis)abilities of each individual player. The game, based on each user's profile, will employ relevant techniques (including artificial intelligence) to compensate for individual weaknesses (e.g., novice vs.

experienced player, single-switch vs. full control), so that two or more players can co-operate, or even compete, on an equal basis.

Finally, another ambition is to disseminate and share the knowledge that was gained while working on this thesis through further publications, lectures and tutorials.

BIBLIOGRAPHY

Apple (2004). *Apple Human Interface Guidelines*. Retrieved September 3, 2004, from Apple Developer Connection Web Site: http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/OSXHIGuidelines.pdf

Architectural and Transportation Barriers Compliance Board (Access Board) (2000). *Electronic and Information Technology Accessibility Standards*. Retrieved September 14, 2004, from United States Access Board Web Site: http://www.access-board.gov/sec508/508standards.htm

Arrington, M. E., & Staples, S. (2000). *3D Visualisation & Simulation*. Tiburon, CA: Jon Peddie Associates.

Baecker, R., & Small, I. (1990). Animation at the Interface. In B. Laurel (Ed.), *The Art of Human-Computer Interface Design* (pp. 113-122). Reading, MA: Addison-Wesley Publishing Company, Inc.

Bangemann M. et al. (1994). *Europe and the global information society: Recommendations to the European Council Europe and the global information society*. Retrieved October 6, 2004, from Europa - The European Union On-Line Web Site: http://europa.eu.int/ISPO/infosoc/backg/bangeman.html

Basso, V., Enrico, G., Liliana, R., Domenico, T., Davide, L., Andrina, G., Saluaar, D., Letourneur, S., Lorrison, J., Greif, P., Bullinger, A., Stefani, O., Hoffman, & H. (2002). *User Requirements Document*. Internal deliverable of the VIEW of the Future Project, Contract N. IST-2000-26089.

Beavan, C. (2001). *Finger prints: The Origins of Crime Detection and The Murder Case That Launched Forensic Science*. New York, NY: Hyperion.

Berkovitz, J. (1994). Graphical interfaces for young children in a software-based math curriculum. In K. Plaisant (Ed.), *Conference companion on Human factors in computing systems* (pp. 247-248). New York, NY: ACM Press.

Bevan, N., Azuma, M. (1997). Quality in Use: Incorporating Human Factors into Software Engineering Lifecycle. In J. Harauz (Ed.), *Proceedings of the 3rd IEEE International Software Engineering Standards Symposium and Forum* (pp.169-179). Washington, DC: IEEE Computer Society.

Bier E.A., & Freeman, S. (1991). MMM: A user interface architecture for shared editors on a single screen. In *Proceedings of the UIST'91 Symposium on User Interface Software and Technology* (pp. 79-86). New York, NY: ACM Press.

Bouras, C., Kapoulas, V., Konidaris, A., Ramahlo, M., Sevasti, A., & Van de Velde, W. (2000). Diary composer: Supporting reflection on past events for young children. In J. Bordeau, & R. Heller (Eds.), *Proceedings of World Conference on Educational Multimedia, Hypermedia & Telecommunications* (pp. 105-110). Montréal, Canada: Association for the Advancement of Computing in Education.

Bowman, D. A., Gabbard, J., & Hix, D. (2001). *Usability Evaluation in Virtual Environments: Classification and Comparison of Methods*. Technical Report TR-01-17, Computer Science Department, Virginia Tech.

Boyle, R.D., & Thomas, R.C. (1988). *Computer Vision: A First Course*. Oxford: Blackwell Scientific Publications

Bricker, L.J., Baker, M.J., & Tanimoto, S.L. (1997). Support for cooperatively controlled objects in multimedia applications. In S. Pemberton (Ed.), *Proceedings of Human Factors in Computing Systems CHI'97, Extended Abstracts* (pp. 313-314). New York, NY: ACM Press.

Bricker, L. J., Bennett, M. J., Fujioke, E., & Tanimoto, S. L. (1999). Colt: A System for Developing Software that Supports Synchronous Collaborative Activities. In B. Collins & R. Oliver (Eds.), *Proceedings of Conference on Educational Multimedia, Hypermedia & Telecommunications (EdMedia'99)* (pp. 587-592). Charlottesville, VA: AACE.

Champandard, A. J. (2003). *AI Game Development*. Indianapolis, Indiana: New Riders.

Charitos, D. (1997). Designing Space in Virtual Environments for Aiding Wayfinding Behaviour. *Social Science Computing Review*, 10(2), 453-469.

Chen, J. (2003). Effective Interaction Techniques in Information-Rich Virtual Environments. In *Proceedings of the Young VR 2003*. Retrieved June 9, 2004, from the author's personal Web Site: http://csgrad.cs.vt.edu/~jichen8/publications/YVR-2003/finalversion/YVR-JianChen.pdf

Chiueh, T., Mitra, T., Neogi, A., & Yang, CK. (1998). Zodiac: A history interactive video authoring system. In *Proceedings of the 6th ACM International Multimedia Conference (Multimedia'98)* (pp. 435-443). New York, NY: ACM Press.

Colley, A., Hill, F., Hill, J. & Jones, A. (1995). Gender Effects in the Stereotyping of Those with Different Kinds of Computing Experience. *Journal of Educational Computing Research*, 12(1), 19-27.

Cooper, A. (1995). *UI Design About Face: The Essentials of user Interface Design.* Foster City, CA: IDG Books Worldwide.

Crosier, J., Nichols, S., Stedmon, A., Patel, H., Hoffmann, H., Deisinger, J., Rönkkö, J., Grammenos, D., Protogeros, Z., Laukkanen, S., Amditis, A., Panou, M., Basso, V., Saluaar, D. Letourneur, S., Bullinger, A., Mager, R., D'Cruz, M., & Wilson, J. (2000). *D.1.1 State of the Art and Market Analysis.* Public deliverable of the VIEW of the Future Project, Contract N. IST-2000-26089. Retrieved November 2, 2003, from VIEW of the Future Project Web Site: http://www.view.iao.fraunhofer.de/pdf/D1_1.pdf

CyberEdge Information Services (2001). *The Market for Visualisation / Virtual Reality Systems Report.* Oakland, CA: Cyberedge Information Services.

Dalmau, D. (2004). *The Structure of a Game AI System.* Retrieved September 9, 2004, from Informit NetworkWeb Site: http://www.informit.com/articles/article.asp?p=169492

D'Amico, C. (2001). *Gaming With A Disability.* Retrieved September 14, 2004, from 3DAction planet Web Site: http://www.3dactionplanet.com/features/editorials/disabledgamers/index2.shtml

Darken, R. P., & Peterson, B. (2002). Spatial Orientation, Wayfinding, and Representation. In K. M. Stanney (ed.), *Handbook of Virtual Environment Technology* (pp. 493-518). Mahwah, NJ: Lawrence Erlbaum Associates

Darken, R. P., & Goerger, S. R. (1999). The transfer of strategies from virtual to real environments: an explanation for performance differences? In C. Landauer & K. L. Bellman (Eds.), *Proceedings of the Virtual Worlds and Simulation Conference (VWsim '99)* (pp. 159-164). San Diego, CA: Society for Computer Simulation.

Darken, R. P., Cockayne, R., & Carmein, D. (1997). The Omni-Directional Treadmill: A locomotion device for virtual worlds. In *Proceedings of the 10th annual ACM symposium on User interface software and technology (UIST '97)* (pp. 213-221). New York, NY: ACM Press.

Darken, R. P., & Sibert, J. L. (1993). A Toolset for Navigation in Virtual Environments. In *Proceedings of the 6th annual ACM symposium on User interface software and technology (UIST '93)* (pp. 157-165). New York, NY: ACM Press.

Darken, R. P., & Sibert, J. L. (1996). Navigating in Large Virtual Worlds. *The International Journal of Human-Computer Interaction*, 8(1), 49-72.

Delaney, B. (1999). *The Market for Visual Simulation / Virtual Reality Systems*. Oakland, CA: Cyberedge Information Services.

Digiplay (2004). *Some Key Gaming Facts*. Retrieved September 14, 2004, from Digiplay Web Site: http://www.digiplay.org.uk/facts.php

Dreyfuss, H. (1984). *Symbol Sourcebook: An Authoritative Guide to International Graphic Symbols*. New York, NY: John Wiley & Sons.

Druin, A., & Solomon, C. (1996). *Designing Multimedia Environments for Children*. New York, NY: John Wiley & Sons, Inc.

Druin, A. (1999a). Cooperative Inquiry: Developing New Technologies for Children with Children. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit (CHI '99)* (pp. 592-599). New York, NY: ACM Press.

Druin, A. (Ed.). (1999b). *The Design of Children's Technology*. San Francisco, CA: Morgan Kaufmann Publishers.

Druin, A. (2002). The Role of Children in the Design of New Technology. *Behaviour and Information Technology*, 21(1), 1-25.

Durandell, A., Glissov, P. & Siann, G. (1995). Gender and Computing: Persisting Differences. *Educational Research*, 37(3), 219-228.

Edwards V., Monaghan F., Pemberton L., & Knight J. (2002). Fabula: A Bilingual Multimedia Authoring Environment for Children Exploring Minority Languages. *Language Learning and Technology*, 6(2), 59-69.

Edwards, J., & Hand, C. (1997). MaPS: Movement and Planning Support for Navigation in an Immersive VRML Browser. In *Proceedings of the Second Symposium on the Virtual Reality Modeling Language (VRML'97)* (pp. 65-73). New York, NY: ACM.

Ellis, S. R., Smith, S. R., Grunwald, A. J., & McGreevy, M. W. (1991). Direction judgment error in computer generated displays and actual scenes. In S. Ellis (Ed.), *Pictorial Communication in Virtual and Real Environments*. London: Taylor and Francis.

Elvins, T. T., Nadeau, D. R., & Kirsh, D. (1997). Worldlets - 3D thumbnails for wayfinding in virtual environments. In *Proceedings of the 10th annual ACM symposium on User interface software and technology* (pp. 21–30). New York, NY: ACM Press.

Entertainment Software Association (ESA) (2003). *Demographic Information.* Retrieved September 20, 2004, from Entertainment Software Association Web Site: http://www.theesa.com/pressroom.html

Erickson, T. D. (1990). Working with Interface Metaphors. In B. Laurel (Ed.), *The Art of Human-Computer Interface Design* (pp. 65-73). Reading, MA: Addison-Wesley Publishing Company.

European Commission (2002). *eEurope 2005: An information society for all.* Retrieved September 4, 2004, from Europa - The European Union On-Line Web Site: http://europa.eu.int/information_society/eeurope/2005/index_en.htm

Fernandes, T. (1995). *Global Interface Design: A Guide to Designing International User Interfaces*. Boston, MA: AP Professional.

Franklin, S., & Graesser, A. C. (1997). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In J. P. Muller, M. J. Wooldridge, & N. Jennings (Eds.), *Intelligent Agents III, Agent Theories, Architectures, and Languages* (pp. 21-35). Berlin: Springer Verlag.

Frutiger, A., & Bluhm, A. (1998). *Signs and Symbols: Their Design and Meaning.* New York, NY: Watson-Guptill Publications

Furger, R. (1998). *Does Jane Compute? : Preserving Our Daughters' Place in the Cyber Revolution*. New York, NY: Warner Books.

Gabbard, J. L., Hix, D., & Swan, E. J. (1999). User Centered Design and Evaluation of Virtual Environments. *IEEE Computer Graphics and Applications,* 19(6), 51-59.

Galdo, M., & Nielsen, J. (1996). *International User Interfaces*. New York, NY: John Wiley & Sons.

Galitz, O. W. (1997). *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and techniques*. New York, NY: John Wiley & Sons.

GA-SIG (2004). *White Paper: Accessibility in Games: Motivations and Approaches.* Retrieved September 4, 2004, from International Game Developers Association Web Site: http://www.igda.org/accessibility/IGDA_Accessibility_WhitePaper.pdf

Gotts, G., & Cox, T. (1988). *Stress and arousal checklist: A manual for its administration, scoring and interpretation.* Melbourne: Swinburne Press.

Grammenos, D., Savidis, A., & Stephanidis C. (2004). An Accessible Two-player Multi-modal Board Game. *ERCIM News,* 57, 35-36.

Grammenos, D., Savidis, & Stephanidis, C. (2005). The Development of a Sensory System for Intelligent Game Creatures. In C. Stephanidis (Ed.), *Universal Access in HCI: Exploring New Interaction Environments - Volume 7 of the Proceedings of the 11th International Conference on Human-Computer Interaction (HCI International 2005)* [CD-ROM]. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Grammenos, D., Savidis, A., & Stephanidis, C. (2005). UA-Chess: A Universally Accessible Board Game. In C. Stephanidis (Ed.), *Universal Access in HCI: Exploring New Interaction Environments - Volume 7 of the Proceedings of the 11th International Conference on Human-Computer Interaction (HCI International 2005)* [CD-ROM]. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Grammenos, G., Filou, M., Papadakos, P., & Stephanidis, C. (2002). Virtual Prints: Leaving trails in Virtual Environments. In W. Stürzlinger & S. Müller (Eds.), *Proceedings of the Eighth Eurographics Workshop on Virtual Environments* (pp. 131-138). New York, NY: ACM Press.

Grand, S., Cliff, D., & Malhotra, A. (1997). Creatures: Artificial Life Autonomous Software Agents for Home Entertainment. In *Proceedings of the First International Conference on Autonomous Agents* (pp. 22-29). New York, NY: ACM Press.

Hanna, L., Risden, K., & Alexander, K. (1997). Guidelines for Usability Testing with Children. *Interactions Magazine*, 4(5), 9-14.

Hansen, K., Bernsen, N., O., Beach, D., & Koutra, C. (2002). *Deliverable D1.3.3: School Trials*. Internal report of the "Today's Stories" Project (P29312) funded by the European Commission in the framework of the Intelligent Information Interfaces (i3), Experimental School Environment Programme.

Hayes-Roth, B., Sincoff, E., Brownston, L., Huard, R., & Lent, B. (1995). Directed improvisation with animated puppets. In I. Katz, R. Mack, & L. Marks (Eds.), *Conference companion on Human factors in computing systems (CHI '95)* (pp.). New York, NY: ACM Press.

Healy, J. M. (1999). *Failure to connect: how computers affect our children's minds -- and what we can do about it*. New York, NY: Simon & Schuster.

Hewett, T. T. (1992). *ACM SIGCHI curricula for human-computer interaction*. New York, NY: ACM Press.

Hix, D., & Hartson, H. R. (1993). *Developing User Interfaces: Ensuring Usability through Product & Process*. New York, NY: John Willey and Sons.

Horton, W. K. (1994). *The Icon Book: Visual Symbols for Computer Systems and Documentation*. New York, NY: John Wiley & Sons.

Howlett, V. (1996). *Visual Interface Design for Windows*. New York, NY: John Wiley & Sons.

Huang, Z., Boulic, R., Magnenat-Thalmann, N., & Thalmann, D. (1995). A Multi-sensor Approach for Grasping and 3D Interaction. In R. Earnshaw & J. Vince (Eds.) Computer graphics: developments in virtual environments (pp.235-253). London, UK: Academic Press.

Hunt, E., & Waller, D. (1999). *Orientation and Wayfinding: A Review* (ONR technical report N00014-96-0380). Arlington, VA: Office of Naval Research.

IBM (2002). *IBM software accessibility checklist*. Retrieved September 3, 2004, from IBM Accessibility Center Web Site: http://www-3.ibm.com/able/guidelines/software/accesssoftware.html

Ieronutti L., Ranon R., & Chittaro L. (2004). Automatic Derivation of Electronic Maps from X3D/VRML Worlds. In *Proceedings of Web3D 2004: 9th International Conference on 3D Web Technology* (pp. 61-70). New York, NY: ACM Press.

Inkpen, K. (1997). Three Important Research Agendas for Educational Multimedia: Learning, Children and Gender. In T. Müldner & T. Reeves (Eds.) *Proceedings of Conference on Educational Multimedia, Hypermedia & Telecommunications (EdMedia'97)* (pp. 521-526). Charlottesville, VA: AACE.

Inkpen, K., Booth, K. S., & Klawe, M. (1996). *Interaction styles for Educational Computer Environments: A Comparison of Drag-and-Drop vs. Point-and-Click* (Technical Report 96-17). Vancouver, Canada: Department of Computer Science, University of British Columbia.

Inkpen, K., Booth, K. S., Gribble, S. D., & Klawe, M. (1995). Give and Take: Children Collaborative on One Computer. In I. Katz, R. Mack & L. Marks (Eds.), Companion Proceedings of the Conference on Human Factors and Computing (CHI '95) (pp. 258-259). New York, NY: ACM press.

Irish National Disability Authority (2001). *IT Accessibility Guidelines*. Retrieved September 3, 2004, from Irish National Disability Authority IT Accessibility Guidelines Web Site: http://accessit.nda.ie

ISO (1997). *ISO 9241: Ergonomic Requirements for Office Work with Visual Display Terminals*. Geneva, Switzerland: International Organisation for Standardisation

ISO (1998). *ISO/IEC 14598-1: Information Technology - Evaluation of Software Products - Part 1 General guide*. Geneva, Switzerland: International Organisation for Standardisation

ISO (1999). *ISO 13407: Human-centred design processes for interactive systems.* Geneva, Switzerland: International Organisation for Standardisation

ISO (2002): *ISO 14915: Software ergonomics for multimedia user interfaces*. Geneva, Switzerland: International Organisation for Standardisation

ISPO (1995). *Introduction to the Information Society - The European Way*. Brussels, Luxembourg: Information Society Project Office.

Iwata, H., & Yoshida, Y. (1999). *Path reproduction tests using a Torus Treadmill. Presence*, 8, 587-597.

Jennings, N. R., & Wooldridge, M., (2001). Agent-Oriented Software Engineering. In J. Bradshaw (Ed.), *Handbook of Agent Technology*. Cambridge, MA: AAAI Press/MIT Press.

Jonassen, D., Mayes, T., & McAleese, R. (1993). A Manifesto for a Constructivist Approach to Uses of Technology in Higher Education. In T.M. Duffy, J. Lowyck, & D.H. Jonassen (Eds.), *Designing Environments for Constructive Learning* (vol. 105, pp. 231-247). NATO ASI Series, Series F: Computer and Systems sciences. Berlin: Springer-Verlag

Jones, T. (1990). Children and animated computer icons. *Journal of Research on Computing in Education,* Spring 1990, 300-309.

Kearney, J. (1999). *Tracking: A Blueprint for Learning How*. El Cajon, CA: Pathways Press.

Kennedy, R. S., Lane, N. E., Berbaum, K. S., & Lilienthal, M. G. (1993). Simulator Sickness Questionnaire: An enhanced method for quantifying simulator sickness. *The International Journal of Aviation Psychology*, 3(3), 203-220.

Kim M., & Song, J. (1995). Multimedia Documents with Elastic Time. In *Proceedings of the third ACM international conference on Multimedia* (pp. 143-154). New York, NY: ACM press.

Kim, M. Y. (1995). Creative Multimedia for Children: Isis Story Builder. In I. Katz, R. Mack & L. Marks (Eds.), Companion Proceedings of the Conference on

Human Factors and Computing (CHI '95) (pp.37-38). New York, NY: ACM Press.

Knuth, R. A., & Cunningham, D. J. (1993). Tools for Constructivism. In T. M. Duffy, J. Lowyck, & D.H. Jonassen (Eds.), *Designing Environments for Constructive Learning* (vol. 105, pp. 163-188). NATO ASI Series, Series F: Computer and Systems sciences. Berlin: Springer-Verlag.

Koutra, C., Kastis, N., Neofotistos, G., van de Velde, W., Ramalho, M., & Panayi, M. (2000). Interactive Learning Environments for Children: User Interface Requirements for a Magic Mirror and Diary Composer Environment. In C. Stephanidis (Ed.), *Electronic Proceedings of the ERCIM WG UI4ALL one-day joint workshop with i3 Spring Days 2000 on Interactive Learning Environments for Children.* Retrieved December 7, 2001, from UI4ALL Web Site: http://ui4all.ics.forth.gr/i3SD2000/Koutra.PDF

Laird, J.E., & van Lent, M. (2001). Interactive computer games: Human-level AI's killer application. *AI Magazine*, 22(2), 15-25.

Langley, P. (1996). *Elements of Machine Learning.* San Francisco, CA: Morgan Kaufmann.

Leonard, T. (2003). *GDC 2003: Building an AI Sensory System: Examining The Design of Thief: The Dark Project.* Retrieved September 9, 2004, from Gamasutra - The Art & Science of Making Games Web Site: http://www.gamasutra.com/gdc2003/features/20030307/leonard_01.htm

Lewis, R. J. (1995). IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction*, 7(1), 57-78.

Loeffler, C. E., & Anderson, T. (Eds.) (1994). *The Virtual Reality Casebook.* New York, NY: Van Nostrand Rheinhold.

Maes, P., Darrell, T., Blumberg, B., & Pentland, A (1995). The ALIVE system: Full-body interaction with autonomous agents. In N. Magnenat-Thalmann & D. Thalmann (Eds.), *Proceedings of the Computer Animation '95 Conference* (pp.11-18). Los Alamitos, CA: IEEE Computer Society Press.

Magnenat-Thalman N., & Thalmann, D. (Eds.) (1994). *Artificial Life and Virtual Reality.* New York, NY: John Wiley & Sons

Marsh, T., & Wright, P. (1999). Co-operative Evaluation of a Desktop Virtual Reality System. In S. Smith & M. Harrison (Eds.), *Proceedings of Workshop on User-*

Centred Design and Implementation of Virtual Environments (pp. 99-108). York, UK: University of York.

McGovern, D. E. (1993). Experience and results in teleoperation of land vehicles. In S. Ellis (Ed.), *Pictorial Communication in Virtual and Real Environments* (pp. 182-195). London: Taylor and Francis.

Microsoft (2004). *Microsoft Official Guidelines for User Interface Developers and Designers*. Retrieved August 28, 2004, from MSDN Web Site: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwue/html/welcome.asp

Microsoft (2003). *Accessible Technology Market Research*. Retrieved September 4, 2004, from Accessibility at Microsoft Web Site: http://www.microsoft.com/enable/research/default.aspx

Modley, R., Myers, W. R., & Comer, D. G. (1977). *Handbook of Pictorial Symbols: 3250 Examples from International Sources*. New York, NY: Dover Publications.

Moran, T. (1980). A framework for studying human-computer interaction. in R. A. Guedij, P. J. W. ten Hagen, F. R. A. Hopgood, H. A. Tucker & D. A. Duce (Eds.), *Methodology of Interaction* (pp. 293-301). Amsterdam: North-Holland.

Mountford, S. J., & Gaver, W. W. (1990). Talking and Listening to Computers. In B. Laurel (Ed.), *The Art of Human-Computer Interface Design* (pp. 113-122). Reading, MA: Addison-Wesley.

Mourouzis, A., Grammenos, D., Filou, M., Papadakos, P., & Stephanidis, C. (2004). Case Study: Sequential Evaluation of the Virtual Prints Concept and Pilot Implementation. In *Proceedings of the Workshop on Virtual Reality Design and Evaluation*, Nottingham, U.K., 22-23 January [CD-ROM - ISBN 0 85358 123 1]. University of Nottingham, United Kingdom.

Mourouzis, A., Grammenos, D., Filou, M., Papadakos, P., & Stephanidis, C. (2003). Virtual Prints: an empowering tool for virtual environments. In D. Harris, V. Duffy, M. Smith, & C. Stephanidis (Eds.), Human - Centred Computing: Cognitive, Social and Ergonomic Aspects - Volume 3 of the Proceedings of the 10th International Conference on Human-Computer Interaction (HCI International 2003), Crete, Greece, 22-27 June (pp. 1426 - 1430). Mahwah, NJ: Lawrence Erlbaum Associates.

Munro, A.J., Höök, K., & Benyon, D.R. (1999). Footprints in the Snow. In A.J. Munro, K. Höök, & D.R. Benyon (Eds.), *Social Navigation of Information Space* (pp. 1-14). London: Springer-Verlag.

Nicol, A. (1990). Interfaces for Learning: What Do Good Teachers Know That We Don't? In B. Laurel (Ed.), *The Art of Human-Computer Interface Design* (pp. 113-122). Reading, MA: Addison-Wesley.

Nielsen, J. (1990). *Designing User Interfaces for International Use*. Amsterdam: Elsevier Science Publishers.

Nielsen, J. (1993). *Usability Engineering*. Boston, MA: Academic Press.

Nielsen J.. & Mack, R. (Eds.) (1994). *Usability Inspection Methods*. New York, NY: John Wiley & Sons.

Nielsen, J. (1994). Heuristic Evaluation. In J. Nielsen & R. Mack (Eds.), *Usability inspection methods* (pp. 25-62). New York, NY: John Wiley & Sons.

Norman, D., & Draper, P. (1986). *User-centred Design: New perspective in Human Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Norman, D. (1988). *The Psychology of Everyday Things*. New York, NY: Basic Books.

Norman, D. (1990). Why Interfaces Don't Work. In B. Laurel (Ed.), *The Art of Human-Computer Interface Design* (pp. 209-219). Reading, MA: Addison-Wesley

Noser, H., & Thalmann, D. (1995). Synthetic Vision and Audition for Digital Actors. *Computer Graphics Forum*, 14(3), 325-336.

O'Modhrain, S. (2004). *Accessible gaming*. Retrieved September 14, 2004, from BBC Ouch! Web Site: http://www.bbc.co.uk/ouch/closeup/gaming.shtml

Oosterholt, R., Kusano, M., & de Vries, G. (1996). Interaction design and human factors support in the development of a personal communicator for children. In M. J. Tauber (Ed.), *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground* (pp. 450 – 457). New York, NY: ACM Press

Passig, D., & Levin, H. (2000). Gender preferences for multimedia interfaces. *Journal of Computer Assisted Learning*, 16(1), 64-71.

Patel, H., Stedmon, A., Nichols, S. C., D'Cruz, M. D., Mager, R., Stoermer, R., Schaerli, H., Estoppey, K. H., & Bullinger, A. H. (2003). *D3.2: Usability test-battery manual*. Public deliverable of the VIEW of the Future Project, Contract N. IST-2000-26089.

Payne, E. C., & McArthur, R.C. (1990). *Developing Expert Systems: A Knowledge Engineer's Handbook for Rules and Objects*. New York, NY: John Wiley & Sons.

Pereira, F. C., & Grosz, B. J. (Eds.) (1994). *Natural Language Processing*. Cambridge, MA: MIT Press.

Peterson, B., Wells, M., Furness III, T. A., & Hunt, E. (1998). The Effects of the Interface on Navigation in Virtual Environments. In *Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting* (pp. 1496-1505). Santa Monica, CA: HFES.

Piaget, J. (1988). Extracts from Piaget's theory (G. Gellerier & J. Langer, Trans.). In K. Richardson & S. Sheldon (Eds.), *Cognitive development to adolescence: A reader* (pp. 3-18). Hillsdale, NJ: Erlbaum. (Reprinted from Manual of child psychology, pp. 703-732, by P. H. Mussen, Ed., 1970, New York, NY: John Wiley & Sons).

Ponceleon, D., Srinivasan, S., Amir, A., Petkovic, D., & Diklic, D. (1998). Key to effective video retrieval: Effective cataloging and browsing. In *Proceedings of the 6th ACM International Multimedia Conference (Multimedia'98)* (pp. 99-107). New York, NY: ACM Press.

Prensky, M. (2000). *Digital Game-Based Learning*. San Francisco, CA: McGraw-Hill

Razzaque, S., Swapp, D., Slater, M., Whitton, M. C., & Steed, A. (2002). Redirected walking in place. In W. Stürzlinger & S. Müller (Eds), *Proceedings of the Workshop on Virtual Environments 2002* (pp. 123-130). ACM International Conference Proceeding Series, vol. 23. New York, NY: ACM Press.

Renault, O., Magnenat-Thalmann, N., & Thalmann, D. (1990). A Vision-based Approach to Behavioural Animation. *Journal of Visualization and Computer Animation*, 1(1), 18-21.

Reynolds, C.W. (1993). An Evolved, Vision-Based Behavioral Model of Coordinated Group Motion. In J.A. Meyer et al. (Eds.) *From Animals to Animats, Proc. 2nd International Conf. on Simulation of Adaptive Behavior* (pp.384-392). Cambridge, MA: MIT Press.

Satalich, G.A. (1995). Navigation and Wayfinding in Virtual Reality: Finding the Proper Tools and Cues to Enhance Navigational Awareness. Master's Thesis, University of Washington, Seattle. Retrieved April 12, 1999, from Human Interface Technology Laboratory Web Site: http://www.hitl.washington.edu/publications/satalich

Screendigest (2004). *Press Release: Video Games Market Demonstrates Over 100 Per Cent Growth In Six Years, London.* Retrieved September 14, 2004, from Screendigest Web Site: http://www.screendigest.com/reports/ils04/press_releases_31_08_2004n/view

Scriven, M. (1967). The methodology of evaluation. In R. E. Stake (Ed.), *Perspectives of curriculum evaluation. American Educational Research Association monograph series on evaluation, no. 1.* Chicago, IL: Rand McNally.

Seeman, L. (2002) Inclusion of Cognitive Disabilities in the Web Accessibility Movement. In Proceedings of the Eleventh International World Wide Web Conference. Honolulu, Hawaii, USA [CD-ROM ISBN 1-880672-20-0]. Retrieved March 6, 2003, from WWW2002 Conference Web Site: http://www2002.org/CDROM/alternate/689/

Shneiderman, B. (1998). *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Reading, MA: Addison-Wesley.

Steiner, K., & Moher, T. (1992). Graphic StoryWriter: an interactive environment for emergent storytelling. In P. Bauersfeld, J. Bennett & G. Lynch (Eds.), *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '92)* (pp. 357-364). New York, NY: ACM Press.

Stephanidis, C. (2001a). User Interfaces for All: New perspectives into Human-Computer Interaction. In C. Stephanidis (Ed.), *User Interfaces for All - Concepts, Methods, and Tools* (pp. 3-17). Mahwah, NJ: Lawrence Erlbaum Associates.

Stephanidis, C. (2001b). The concept of Unified User Interfaces. In C. Stephanidis (Ed.), *User Interfaces for All - Concepts, Methods, and Tools* (pp. 371-388). Mahwah, NJ: Lawrence Erlbaum Associates.

Stephanidis, C., Salvendy, G., Akoumianakis, D., Bevan, N., Brewer, J., Emiliani, P. L., Galetsas, A., Haataja, S., Iakovidis, I., Jacko, J., Jenkins, P., Karshmer, A., Korn, P., Marcus, A., Murphy, H., Stary, C., Vanderheiden, G., Weber, G., & Ziegler, J. (1988). Towards an Information Society for All: An International R&D Agenda. *International Journal of Human-Computer Interaction*, 10(2), 107-134.

Stewart, J., Bederson, B. B., & Druin, A. (1999). Single display groupware: A model for copresent collaboration. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit (CHI '99)* (pp. 286-293). New York, NY: ACM Press.

Stoakley, R., Conway, M.J., & Pausch, R. (1995). Virtual reality on a WIM: Interactive worlds in miniature. . In I. Katz, R. Mack, L. Marks, M. B. Rosson & J. Nielsen (Eds.), *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI '95)* (pp. 265–272). New York, NY: ACM Press.

Stoev, S., Schmalstieg, D., & Straßer, W. (2001). Through-The-Lens Techniques for Remote Object Manipulation, Motion, and Navigation in Virtual Environments. In B. Froehlich, J. Deisinger & H.-J. Bullinger (Eds.), *Proceedings of the Joint Immersive Projection Technology / EUROGRAPHICS Workshop on Virtual Environments (IPT/EGVE 2001)* (pp. 51-60). Berlin: Springer Verlag.

Story, M. F. (1998). Maximising Usability: The Principles of Universal Design. *Assistive Technology*, 10(1), 4-12.

Tattersall, I. (1995). *The Fossil Trail : How We Know What We Think We Know About Human Evolution*. New York, NY: Oxford University Press.

Temple University (1999). *Guidelines for the Design of Educational Software*. Retrieved July 4, 2000, from Temple University Web Site: http://www.temple.edu/dentistry/di/Eds.wstd/title.htm

Templeman, J. N., Denbrook, P. S., & Sibert, L. E. (1999). Virtual Locomotion: Walking in Place through Virtual Environments. *Presence: Teleoperators and Virtual Environments*, 8(6), 598-617.

Today's Stories Consortium. (1999). *Deliverable D 3.3.1: Diary Composer Requirements and Concept Prototypes*. Internal report of the "Today's Stories" Project (P29312) funded by the European Commission in the framework of the Intelligent Information Interfaces (i3), Experimental School Environment Programme.

Today's Stories Consortium. (2000). *Deliverable D 3.3.2: Diary Composer Prototypes*. Public technical report of the "Today's Stories" Project (P29312) funded by the European Commission in the framework of the Intelligent Information Interfaces (i3), Experimental School Environment Programme.

Tognazzini, B. (1996). *TOG on Interface*. Reading, MA: Addison-Wesley.

Tromp, J., & Nichols, S. (2003). VIEW-IT: A VR/CAD Inspection Tool for use in Industry. In D. Harris, V. Duffy, M. Smith, & C. Stephanidis (Eds.), Human - Centred Computing: Cognitive, Social and Ergonomic Aspects - Volume 3 of the Proceedings of the 10th International Conference on Human-Computer

Interaction (HCI International 2003), Crete, Greece, 22-27 June (pp. 1451 - 1455). Mahwah, NJ: Lawrence Erlbaum Associates.

Tu, X., & Terzopoulos, D. (1994). Artificial Fishes: Physics, Locomotion, Perception, Behavior. In *Proceedings of the 21st Annual Conference on Computer Graphics and interactive Techniques SIGGRAPH '94* (pp.43-50). New York, NY: ACM Press

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59, 433-460.

United Nations (UN) (2004). *World Programme of Action Concerning Disabled Persons: Current situation.* Retrieved September 2, 2004, from The United Nations Web Site: http://www.un.org/esa/socdev/enable/diswpa04.htm

van Ballegooij, A., & Eliëns, A. (2001). Navigation by Query in Virtual Worlds. In *Proceedings of the sixth international conference on 3D Web technology* (pp. 77-83). New York, NY: ACM Press.

van Lent, M., Laird, J., Buckman, J., Hartford, J., Houchard, S., Steinkraus, K., & Tedrake, R. (1999). Intelligent agents in computer games. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence* (pp. 929-930). Cambridge, MA: MIT Press.

Vinson, N. G. (1999). Design Guidelines for Landmarks to Support Navigation in Virtual Environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit* (CHI '99) (pp. 278-285). New York, NY: ACM Press.

Ware, C., & Osborne, S. (1990). Exploration and virtual camera control in virtual three dimensional environments. Proceedings of the 1990 Symposium on Interactive 3D Graphics. *Special Issue of Computer Graphics*, 24, 175–183.

Weinschenk, S., Jamar, P., & Yeo, C. S. (1997). *GUI Design Essentials*. New York, NY: John Wiley & Sons.

Weizenbaum, J. (1966). ELIZA - A computer program for the study of natural language communication between man and machine. *Communications of the ACM,* 9(1), 36-45.

Wharton, C., Rieman, J., Lewis, C., & Polson, P. (1994). The cognitive walkthrough method: A practitioner's guide. . In J. Nielsen & R. Mack (Eds.), *Usability inspection methods* (pp. 105-140). New York, NY: John Wiley & Sons.

Williamson, B. (2003). *Accessing games through sound, motion and emotion*, EducationGuardian.co.uk, Tuesday December 23, 2003. Retrieved September 14, 2004, from EducationGuardion.co.uk Web Site: http://education.guardian.co.uk/elearning/story/0,10577,1112191,00.html

Wilson, K.S. (1988). Palenque: An Interactive Multimedia Digital Video Interactive Prototype for Children In J. J. O'Hare (Ed.), *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '88) (pp. 275-279). New York, NY: ACM Press.

Winn, W. (1993). A Constructivist Critique of the Assumptions of Instructional Design. In T.M. Duffy, J. Lowyck, & D.H. Jonassen (1993). *Designing Environments for Constructive Learning* (vol. 105, pp. 189-212). NATO ASI Series, Series F: Computer and Systems sciences. Berlin: Springer-Verlag.

Witmer, B. G., & Singer, M. J. (1998). Measuring Presence in Virtual Environments: A presence Questionnaire. *Presence: Teleoperators and Virtual Environments*, 7(3), 225-240.

Wood, E. L. (1998). *User Interface Design: Bridging the Gap from User Requirements to Design*. Boca Raton, FL: CRC Press.

Wooldridge, M. J., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2), 115-152.

World Health Organisation (WHO) (2004). *About Ageing and Life Course*. Retrieved September 2, 2004, from World Health Organisation Web Site: http://www.who.int/hpr/ageing/index.htm

World-Wide Web Consortium (W3C) (2004). W3C-WAI resources: Guidelines. Retrieved September 14, 2004, from Web Accessibility Initiative (WAI) Web Site: http://www.w3.org/WAI/Resources/#gl

Wright, P., & Monk, A. (1992). *Co-operative Evaluation: The York Manual*. Tutorial notes, SIGCHI conference on Human factors in computing systems (CHI '92).

Yang, T., Gross, M., D., & Do, E., Y. (2002). Sketching annotations in a 3D Web environment. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves* (CHI '02) (pp. 618-619). New York, NY: ACM Press.

APPENDIX A

Questionnaire used for evaluating the usability of the Virtual Prints mechanism interactive prototype (Section 3.6.3)

Usability Questionnaire

Please answer the following questions about the virtual environment you have just viewed. Most of the questions consist of statements and some ask for your own views. Please indicate how much you agree with each of the statements by circling the appropriate point on the scale. Your answers should be based on your experience with the display technology and virtual environment you have just used. Please provide written answers where appropriate.

1. The Virtual Prints mechanism is easy to learn

   Strongly Agree *Agree* *Neutral* *Disagree* *Strongly Disagree*

2. The Virtual Prints mechanism is easy to use

   Strongly Agree *Agree* *Neutral* *Disagree* *Strongly Disagree*

3. I felt comfortable using the Virtual Prints mechanism

   Strongly Agree *Agree* *Neutral* *Disagree* *Strongly Disagree*

4. The Virtual Prints mechanism helped me in performing my tasks easier

   Strongly Agree *Agree* *Neutral* *Disagree* *Strongly Disagree*

5. The Virtual Prints mechanism helped me in performing my tasks more quickly

   Strongly Agree *Agree* *Neutral* *Disagree* *Strongly Disagree*

6. The Virtual Prints mechanism is a handy tool for orientation

   Strongly Agree *Agree* *Neutral* *Disagree* *Strongly Disagree*

7. The Virtual Prints mechanism is a handy tool for navigation

   Strongly Agree *Agree* *Neutral* *Disagree* *Strongly Disagree*

8. The Virtual Prints mechanism is a handy tool for wayfinding

   Strongly Agree *Agree* *Neutral* *Disagree* *Strongly Disagree*

9. The Virtual Prints mechanism is a handy tool for annotations

   Strongly Agree *Agree* *Neutral* *Disagree* *Strongly Disagree*

10. The Virtual Prints mechanism is a handy tool overall

Strongly Agree	*Agree*	*Neutral*	*Disagree*	*Strongly Disagree*

11. The interaction with the Virtual Prints mechanism is intuitive

Strongly Agree	*Agree*	*Neutral*	*Disagree*	*Strongly Disagree*

12. I enjoyed using the Virtual Prints mechanism

Strongly Agree	*Agree*	*Neutral*	*Disagree*	*Strongly Disagree*

APPENDIX B

Questionnaires used for the evaluation of UA-Chess (Section 4.4)

The following questionnaires are an adaptation of the "IBM Computer Usability Satisfaction Questionnaires" included in:

- Lewis, R. J. (1995). IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction*, 7(1), 57-78.

After-Scenario Questionnaire (ASQ)

1. Overall, I am satisfied with the ease of completing this game round.

strongly agree ⟷ strongly disagree N/A[42]

1 2 3 4 5 6 7

Comments:

2. Overall, I am satisfied with the amount of time it took me to complete all my actions during this game round.

strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

3. Overall, I am satisfied with the support information (messages, documentation) I received during this game round.

strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

[42] N/A = *Not Applicable*

Computer System Usability Questionnaire (CSUQ)

1. Overall, I am satisfied with how easy it is to play this computer game.

strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

2. It is simple to play this computer game.

strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

3. I can play this computer game effectively.

strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

4. I am able to play this computer game quickly.

strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

5. I am able to efficiently play this computer game.

strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

6. I feel comfortable playing this computer game.

strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

7. It was easy to learn to play this computer game.

strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

8. I believe I became efficient quickly playing this computer game.

strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

9. The computer game gives error messages that clearly tell me how to fix problems.

strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

10. Whenever I make a mistake playing this computer game, I recover easily and quickly.

strongly agree ⟷ strongly disagree N/A

1 2 3 4 5 6 7

Comments:

11. The information (such as on-screen messages, feedback and other documentation) provided with this computer game is clear.

strongly agree ← → strongly disagree N/A

1 2 3 4 5 6 7

Comments:

12. It is easy to find the information I need.

strongly agree ← → strongly disagree N/A

1 2 3 4 5 6 7

Comments:

13. The information provided with this computer game is easy to understand.

strongly agree ← → strongly disagree N/A

1 2 3 4 5 6 7

Comments:

14. The information is effective in helping me playing this game.

strongly agree ← → strongly disagree N/A

1 2 3 4 5 6 7

Comments:

15. The organisation of information provided by this game is clear.

strongly agree ← → strongly disagree N/A

1 2 3 4 5 6 7

Comments: