Intelligent Interfaces for Digital Games

Daniel Livingstone

School of Computing, University of Paisley
Paisley, PA1 2BE, UK
daniel.livingstone@paisley.ac.uk

Darryl Charles

School of Computing & Information Engineering
University of Ulster, Coleraine, BT52 1SA
dk.charles@ulster.ac.uk

Abstract

Computer game design and technology continues to evolve at an incredible rate and the digital game systems which players must learn to the use and interact with are often far more complex now than in games of even a few years ago. There is an increasing need for interfaces within games that perform more advanced tasks than simply acting as a means for players to make choices, enter commands, and navigate the game world. Rather, intelligent interfaces between players and game worlds will enable games to reason about the needs, desires and motivations of players and to react accordingly. In this paper we outline the requirement for more intelligent interfaces in digital games and describe the benefits of such interfaces while also discussing some of the challenges that need to be overcome to make them a reality in mainstream game production.

Introduction

The main focus of current activity on artificial intelligence for computer games is on creating increasingly intelligent characters or opponents for players to interact with (Barnes and Hutchens 2002). The focus for this paper is a quite different application of artificial intelligence – the design and implementation of intelligent interfaces (Maybury and Wahlster 1997). We present a brief outline of the ideas behind, and current approaches to, intelligent interfaces. Then we present some of the possible uses they might be put to in modern digital games. In doing so, we will outline the potential benefits of intelligent interfaces in games, and end with a look at some of the significant challenges ahead. We believe that this area of research and development is important for the future of games as they attempt to become more mainstream and attract new audiences. (Beal et. al. 2002) notes that games are still struggling to reach beyond the core young male audience and, by comparing game playing preferences to learning preferences, argues that games that adapt to players might help win new audiences. Accordingly, we feel it is worth considering the different ways in which intelligent interfaces might adapt games to enhance the gameplay experiences of individual players and the challenges inherent in making this work.

Intelligent Interfaces Overview

Intelligent interfaces perform a wide range of tasks and are developed using an equally wide range of approaches. Referral agents act as automated intermediaries, matching up and introducing users to one another for business or even for romance (Foner 1997); memory agents observe what a user is doing and maintain lists of other, relevant, items which might contain information of use to the user (Rhodes and Starner 1996); natural language interfaces attempt to let the user access and manipulate data through normal speech instead of through a traditional GUI; the well known Microsoft paperclip tries to second-guess what the user of an application is trying to do and pops up with the offer of helpful advice at regular intervals. Intelligent interfaces can be as general as agents to assist with general word processing, or as specific as assistants for designers developing safety-critical systems (Jenkins et al. 1997).

What these all have in common is the goal of building interfaces which assist the user in some way – whether the assistance is in the form of helping the user navigate a complex application, carry out a complex task in an application or navigate through an over-bearing quantity of information to find that which is most relevant or useful.

Computer games differ from office and information management applications in a great many ways, and the use and application of intelligent interfaces in games will likewise differ in many ways.

Intelligent Interfaces in Games: Helping the Player

So, to what uses might intelligent interfaces be put in games? Some of the more obvious possible applications are outlined here.

1. Assistance with Micro-Management.

Many strategy games, whether the task is simply to build an army and defeat an enemy or more complex involving the

development of an empire to span millennia, require that a player perform both macro and micro-management of resources. As armies and empires grow, players can find themselves spending more and more time on micro-management and correspondingly less on macro-management. This separates players from the overall objective and can lead to too much time being spent on relatively tedious and onerous tasks instead of the fun ones – crushing opposing armies or winning the space race.

Some games already make use of agents to help in these tasks. For example, the "Civilization" series of games provides advisors who offer advice on city-building. Intelligent user-agents might include advisors who learn from players during the early stages of the game, and in later stages are able to take over the micro-management tasks – with the ability to make decisions similar to those the players would make themselves.

A further example can be found in the role-playing genre. Tasks such as redistributing items amongst team characters or selling off recently acquired loot can take minutes of play time and many dozens of mouse clicks, and may be performed dozens of times in a single session of play. Interfaces that can learn how a given player likes to distribute equipment or which can propose lists of items to try to sell could drastically reduce the time spent by players on mundane 'housekeeping' tasks.

2. Adapting the UI to the User

There exists a range of games – principally strategy, role-playing and adventure – which must solve the problems of presenting large amounts of information to the player and simultaneously allow the user to choose from a wide range of possible actions and interactions. A current research direction in solving these problems in more typical software applications is the development of Adaptive User Interfaces (Rogers and Iba 2001).

In a game with multiple menus and control options, an adaptive user interface could work simply by presenting the most frequently selection options before those which a particular player rarely uses – something like this is already seen in more recent versions of Microsoft software, where rarely used items 'disappear' from menus – but which are always reachable. Alternatively, a more intelligent approach might be where a game detects that a player consistently has difficulties in executing particular commands or strings of commands and then offers help.

3. Assistance in Task Execution

Where the interface can detect what a player is trying to do, it can then try to offer help in completing the task. While it would not be desirable to have the computer play the game for the player, there is scope for built-in assistance that removes the need for players to carry out all tasks by themselves – similar to application 1 above.

An example of this might be in a squad-based game. A number of first-person squad based games exist where players are able to ask other characters to carry out tasks. With an intelligent interface analyzing the players' actions and intent, squad members would be able to pro-actively offer to carry out tasks. This would decrease the need for the player to manage other characters – some element of control would be reduced to accepting or rejecting offers of help. If implemented well it would also increase the perceived intelligence of the computer controlled squad members, and increase the degree of immersion in the game overall.

4. From Tutorials to Mentors

A lot of work on intelligent interfaces has been focused on help systems in particular – on making them more pro-active and genuinely helpful. For advice and information on how to actually play a game – instruction on the basic input commands that can be given, and actions that can be taken directly under player control – many games feature an interactive tutorial. These sometimes are built into the first stages or levels of a game, cycling through a range of actions and activities.

However, it is possible for players to forget how to perform certain actions when the opportunities to carry out the particular actions are infrequent or when they have not played the game for some time. Pro-active help systems offer to explain to users how tasks may be carried out – where appropriate these could be embodied in game as mentor characters or sidekicks, but could alternatively be as simple as pop-up dialogs.

Two game types that have strong potential to benefit from mentors are educational and massively multiplayer games. In both, mentor characters can exist in game and offer advice and information to those playing the game. MMORPG titles typically present very large and diverse worlds with expansive possible paths for players to explore, both in exploring the worlds themselves and in exploring character development possibilities. Mentor agents which can offer guidance targeted to individual users have the potential to make introductory experiences much more pleasant than they might otherwise be.

5. Frustration detection

Many games bought are never completed, and there are a variety of reasons for this. One common cause is that players get stuck at some point in a game. With no progress apparently possible, a player may try to continue playing for some time, becoming increasingly frustrated before giving up. Detecting when a player is stuck might rely on detecting certain patterns of play that might be characteristic of a frustrated player – perhaps play that is increasing erratic or prone to rapid switching between locations.

When the player is truly stuck, help can be offered. A simple scheme for this is demonstrated in "Crash

Bandicoot". If the player repeatedly fails at a particular point, their character is awarded a magic mask – then, no matter how badly the player does on the next attempt, their character will successfully negotiate the offending obstacle. Detecting when a player is stuck in a non-linear game will be less simple, requiring some amount of AI. Detecting what the actual problem is and offering appropriate help may require even more sophistication. As a player wanders around a large and open game world how can a game decide what help to offer, and when to offer it, unless the game has built up some idea of what the player is trying to achieve?

Example: Hint Systems

As an example of the possibilities, consider the non-player character Yorda in the game "Ico". Unless the player very rapidly solves the puzzle in a location, Yorda will often walk around for a short while and then notice something. She will then point to whatever has attracted her attention while calling to the player. This is scripted for a number of locations, but beyond pointing Yorda offers no clues as to how the player is to solve the puzzle.

A natural extension of this approach would be for a non-player character to give additional clues if the player is still unable to solve the puzzle – and to eventually tell the player how to solve the puzzle. Recognizing the possibility of players getting stuck, the puzzle/adventure game "The 7th Guest" featured an in-game hint system that would offer hints and, if that were insufficient, would solve puzzles for the player. The hint system was not proactive however, and required the player to deliberately ask for help – intelligent interfaces can offer help before the player gets frustrated and turns to in-game hint systems or online walkthroughs.

Enhancing and Adapting Gameplay

If we assume that the overall goal of most game AI is to provide an enjoyable challenge for a player, we can propose alternative applications which run directly counter to, or at least orthogonal to, the normal application of intelligent interfaces. Instead of helping the player, an intelligent interface can try to make the game more difficult, deciding when the player is finding the current challenge unsatisfactory and increasing the challenge accordingly. Going beyond this idea of adaptive difficulty, we have the idea of changing not the difficulty but the *gameplay* to suit the player.

Adaptive Difficulty

A very simple implementation of adaptive difficulty currently exists in many racing games, although there are versions known in other genres (Miller 2004). Known as catch-up/slow-down or rubber-banding, this ties the speed of computer-controlled cars to the speed of the player car. Cars far in front of the player slow down while those far behind speed up. In other game genres, determining the ease with which the player is completing the game may not be so simple and may require the ability to evaluate a number of factors, such as the time taken in different areas, number of attempts taken or other, more subtle, indicators.

While the task of adapting the challenge presented by a game itself is not of concern here, it is clear that intelligent interfaces which monitor and support the players can help provide information to a game to help it decide when to increase or decrease the difficulty. Generally, an intelligent interface can allow a game to learn about the user (as noted below, where we discuss player modeling), or even learn from the user. This is potentially useful for building adaptive difficulty into games where players may have multiple objectives to choose from and multiple ways of adapting the difficulty – are certain objectives missed because of player choice or players being unable to progress?

Adaptive Gameplay

In extremis, the actions taken by an intelligent interface could potentially go beyond helping a player, and result in adapting the gameplay itself.

Considering the examples of help already mentioned, removing the need for characters to balance inventories, or order subordinates to search rooms could dramatically affect the experience of playing a typical role-playing game. This can be taken further. Integral to most role-playing games are reasonably large and regular dialogues and frequent combat encounters. Some players may skip through dialogues, relishing the combat while others might enjoy this aspect of role-playing more than the fighting. Being able to detect such tastes, and respond accordingly would allow the game to adjust and allow the player to spend more time on the aspects they enjoy – such as by extending the combat sequences and trimming the dialogue of redundant lines.

Many games already allow players to choose their own style of play – commonly whether to advance using stealth or by force of arms. If implemented, adaptive gameplay could allow designers even greater power in enabling players to choose their own style of play. In turn, this might help games appeal to the widest possible audiences.

Implementing Intelligent Interfaces: Building and Responding to the Player Model

A requirement for most of the applications discussed in this paper is for the intelligent interface to be able to build, and reason about, a player model. User modeling is itself a focus of a significant amount of study, and user modeling in tutoring systems formed the background of (Beal et. al. 2002).

(Houlette 2004) describes a simple method of building a player model based on recording the actions taken by a

player in game, and keeping count of the frequencies of different actions. Such information, once gained, can be put to a variety of uses including training the computer player on data gleaned from how its human opponent is playing the game (Rabin 2002). An artificial neural-net based approach to training AI on player data has also been demonstrated (McGlinchy 2003). In this case, the player model is encoded by the weights of the neural network, which learns what response the player makes to different conditions in the game. The model built is successful at imitating player behavior, but may be less useful for reasoning about or recognizing players' intents and goals.

A different approach is used by (Fagan and Cunningham 2003). As a basis for a player model, the different possible player states, and actions possible for each state, are determined. Actions may cause the player to change state, or may result in the player maintaining the same state. In the simple "Space Invaders" derived game example presented the three player states are 'safe', 'unsafe' and 'very unsafe'. Actions possible include 'hide' or 'emerge' when entering or leaving cover, and 'dodge' and 'fire'.

Here, by learning from sequences of actions that players typically follow, the game is able to predict with reasonable accuracy what action they will take next, at any given moment. This approach and that of (McGlinchy 2003) seem promising but both have yet to be scaled up to complex games, where there may be very rich sets of states and actions to consider.

Whatever methods are to be used, the first problems to be faced are to decide what data to collect, and what methods should be used to interpret it.

Challenges in Building Intelligent Interfaces for Games

Aside from the practical challenges of how to actually implement an intelligent interface, there are several challenges that must be overcome for intelligent interfaces to become accepted and successful in games.

Some of the challenges become readily apparent if one considers 'Clippy', the user assistant in recent versions of Microsoft Office. While some people may find Clippy useful, the consensus appears to be that this intelligent interface is simply an annoyance to be deactivated as soon as possible (Google 2004). Assistant functionality needs an off switch. To minimize the likelihood of players reaching instantly for the off switch the interface needs to offer help without being too intrusive or irritating. This alone is a very significant challenge.

We also have the challenge of being able to reliably interpret not just players' intentions, but also their emotional state to be able to offer help at just the right moment. While some progress has been made on measuring the emotional state of players without the use of additional non-game peripherals to monitor the player (Sykes and Brown 2003), more work is required here to determine its reliability. It may help to use additional means of observing players (such as "EyeToy" or heart-rate monitors), but for the majority of games the existence of such devices cannot be assumed.

In adapting difficulty based on players' current performance, developers need to ensure that their games don't remove all challenge. Our own survey of gamers' attitudes indicates, unsurprisingly, that players dislike games that are too easy as much as games that are too hard. When increasing the difficulty there is also, clearly, a risk of increasing the difficulty too much – and achieving the opposite of the intended goal of maintaining interest in the game.

Answering these challenges require careful balancing; just enough help and just enough intervention. And ultimately, careful consideration must be made of player preferences and motivations.

With the benefits of intelligent interfaces being unproven, and the effort and difficulties in developing them being significant, it is unlikely that any game developer will be keen to integrate them in current projects. Building prototypes and conducting thorough evaluations to determine whether they do actually provide the benefits that we hope they might is another important challenge – and is the challenge on which we are currently embarking.

Conclusions

In this paper we have argued that there are clear applications of intelligent interfaces to digital games. Such interfaces can be used to assist the user in a number of ways and can be applied to a variety of game genres.

Despite there being many applications and possible benefits, there are also clear challenges to be overcome before intelligent interfaces can be used in mainstream game development. It will most likely fall to the academic community to prototype such interfaces in order to more clearly illustrate the benefits and to find solutions to the challenges before their commercial adoption can become a reality.

Acknowledgements

Daniel Livingstone would like to thank the Carnegie Trust for the Universities of Scotland for supporting this work. The authors would also like to thank the anonymous reviewers for their comments on an earlier version of this paper.

References

Barnes, J & Hutchens, J, 2002, Testing Undefined Behaviour as a Result of Learning, in *AI Game Programming Wisdom*, S. Rabin (ed), pp 615-623, Charles River Media

Beal, C., Beck, J., Westbrook, D., Atkin, M., and Cohen, P., 2002, Intelligent Modeling of the User in Interactive Entertainment. In AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment. Stanford, CA

Charles, D., 2003, Enhancing Gameplay: Challenges for Artificial Intelligence in Digital Games. in Level Up: Digital Games Research Conference. Utrecht

Jenkins, D., Livingstone, D., Maclean, D., Reglinski, A., 1997, Supporting Safety-Related Projects with a Designer's Assistant, in Proceedings of the 1st International Conference on Autonomous Agents, Marina Del Rey, CA, USA, Feb 5-8.

Fagan, M. and P. Cunningham, 2003, Case-Based Plan Recognition in Computer Games in ICCBR 2003, 5th International Conference on Case-Based Reasoning. Trondheim, Norway: Lecture Notes in Computer Science 2689, Springer, Berlin.

Foner, L.N., 1997, Yenta : A Multi-Agent, Referral Based Matchmaking System, presented at The First International Conference on Autonomous Agents (Agents '97), Marina del Rey, CA

Houlette, R., 2004, Player Modeling for Adaptive Games, in *AI Game Programming Wisdom 2*, S. Rabin, Editor. Charles River Media, Inc.: Hingham, MA.

McGlinchy, S., 2003, Learning of AI Players from Game Observation Data. In Q. Mehdi, N. Gough and S. Natkin (Eds.), Game-On 2003, 4th International Conference on Intelligent Games and Simulation, London: 197-200.

McNamee, B. and P. Cunningham, 2003, Creating socially interactive non-player characters: The µ-SIV system. International Journal of Intelligent Games & Simulation, 2(1): http://www.scit.wlv.ac.uk/~cm1822/ijigs21.htm.

Maybury, M., Wahlster, W. (Eds), 1997, *Readings in Intelligent User Interfaces*. Morgan Kaufmann

Miller, S., 2004. Auto-Dynamic Difficulty. In *Game Matters*, January 19th 2004, http://dukenukem.typepad.com/

Rabin, S., 2002, *AI Game Programming Wisdom*, Charles River Media

Rhodes, B. J. and Starner, T., 1996, Remembrance Agent: A continuously running automated information retrieval system. In proceedings of The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 96), London, UK

Rogers, S. and Iba, W. (Eds), 2001, Adaptive User Interfaces, Papers from 2000 AAAI Spring Symposium, Technical Report SS-00-01. Menlo Park, Calif: AAAI Press.

Sykes, J. and S. Brown, 2003, Affective Gaming: Measuring emotion Through the Gamepad. In Proceedings of CHI 2003.

Google Web Search, 2004, http://www.google.co.uk/search?q=office+assistant+paperclip