A Framework for the Semi-Automatic Testing of Video Games

Alfredo Nantes, Ross Brown and Frederic Maire

Faculty of Information Technology
Queensland University of Technology
a.nantes@qut.edu.au, r.brown@qut.edu.au, f.maire@qut.edu.au

Abstract

Game environments are complex interactive systems that require extensive analysis and testing to ensure that they are at a high enough quality to be released commercially. In particular, the last build of the product needs an additional and extensive beta test carried out by people that play the game in order to establish its robustness and playability. This entails additional costs from the viewpoint of a company as it requires the hiring of play testers. In the present work we propose a general software framework that integrates Artificial Intelligence (AI) Agents and Computer Vision (CV) technologies to support the test team and help to improve and accelerate the test process. We also present a prototype shadow alias detection algorithm that illustrates the effectiveness of the framework in developing automated visual debugging technology that will ease the heavy cost of beta testing games.

Keywords: Video Game Testing, Synthetic Image Debugging, Virtual Environment Testing, Artificial Intelligence (AI), Computer Vision (CV).

1. Introduction

Video Games cover a large component of the entertainment industry running into billions of dollars spent annually (Riley, 2007). Present games differ from early generations where the graphics consisted of very few polygons and the user-game interaction was restricted to few commands entered through the keyboard. Today games provide remarkably realistic graphics with highly interactive scenarios held together by complex software that requires large development and testing teams. The more complex the software, the more crucial the effort of the companies in testing their product in order to ensure a high enough quality in terms of functionality, stability and robustness in general. Furthermore, a computer game is not only expected to work properly but it has to be, amongst other things, fun, challenging, realistic and well animated.

In the light of all these aspects we can then split the testing problem for a game into three sub problems which we can name: *Entertainment Inspection*, *Environment Integrity Inspection* and *Software Inspection*. The former aims, at a high level, to check playability issues against playability heuristics such as story progression, gameplay functionality, game usability and game mechanics (Desurvire, Caplan, & Toth, 2004). The Environment Integrity Inspection aims to look for perceptual anomalies such as sound, textures, meshes, lighting and shadowing malfunctions. Finally, the Software Inspection validates the software interfaces amongst components and the integrity of the components itself. This includes code inspection, data analysis and algorithm specific testing. Entertainment and Environment Integrity inspection is usually performed by the Quality Assurance (QA) personnel, whereas the Software inspection is responsibility of the development team (Irish, 2005).

Irish points out how QA and game testing are two very important steps in bringing a video game to completion before its commercial release and that rushing these steps could affect the functionality and the playability of the product. On the other hand, testing is also a time consuming and frustrating activity especially with early software builds when the game cannot run for more than a few minutes before crashing. In addition, making testers play the same game over and over again could push them to overlook defects for the haste of getting their job done. This entails an additional cost from the viewpoint of a company as it requires the hiring of a sufficient number of play testers during the test-cycle in order to test the game thoroughly for release.

Although the burden of debugging graphical applications is usually lightened by performance analysis and shader debugging tools such as Microsoft PIX®[1] and GLIntercept®[2], to the best of our knowledge there has been little contribution to assisting the higher level Entertainment and Environment game testing. In this paper, we propose a general framework for addressing the automation of the game testing problem by using Artificial Intelligence (AI) and Computer Vision (CV) technologies. The implementation of such architecture can bring important benefits to the game industry. Indeed it will bring costs savings by decreasing the number of play testers required during the test process. Moreover, it will increase the robustness of the product to release by allowing the coverage of a far larger set of test cases currently performed only by human players.

In Section 2 we review the AI technology that has been applied to computer games and human-agent systems.

[1] http://msdn2.microsoft.com/

[2] http://glintercept.nutty.org/index.html/

Section 3 presents a list of visual anomalies that often affect virtual environments. The general framework is presented in Section 0. In Section 5 we show an implementation of a visual bug detection agent.

2. Related Work

Many approaches have been proposed for improving the game design in terms of its entertainment factors. Researchers have also studied human-machine cooperation for solving tedious and difficult tasks or operating in hostile environments. These two research areas are reviewed in the following section.

2.1. Human Entertainment and Computer Games

Abundant literature exists (Laird, 2001; Merrick & Maher, 2006) aiming to provide non-playing characters (NPCs) with a convincing human-like behavior by using Artificial Intelligence to improve the entertainment features of a game.

Also research in AI has investigated human entertainment models for measuring qualitative factors such as challenge and curiosity (Yannakakis & Hallam, 2006) and approaches for analyzing and reasoning about the user experience when interacting with virtual environments (Yang, Marsh, & Shahabi, 2005). Recently *self-playing experiments* have been applied to computer games with the aim of modeling the integrity and fairness in rudimentary computer games, assisting the game design (Macleod, 2005). Denzinger et al. (2004) have applied similar systems to a complex game (FIFA-99) showing how their agents, employing a genetic algorithm (GA), were able to find faults in the game mechanics.

Although all these results could be used for automating the game testing process, they do not address Environment Integrity issues that surely affect playability and gameplay functionality or in general the human experience when interacting with the Environment. In our work we propose to extend the previous research to encompass also lower level, yet important Environment Inspection issues in order to thoroughly address the automation of the game testing problem.

2.2. Human-Agent teaming

In modern AI research, agents are seen as assistants in advisory and decision support roles in critical or high workload situations. An agent is a computer system having the ability to provide simple autonomy, sociability, reactivity or sensing capability and pro-activeness (Wooldridge & Jennings, 1995).

In their work Urlings et al. (2006) showed how human-agent teaming architectures are being explored with growing interest due to their high capability to adapt to unpredictable tasks environment, yet with a commonly agreed need for human supervision. These ideas have been successfully applied in fields like system analysis, design and engineering, information processing and computer games. In particular Urlings et al. applied the Belief-Desire-Intention (BDI) syntax and JACK – a commercial BDI agent development language architecture – to demonstrate their teaming framework of human-agent coordination within the commercial computer game Unreal Tournament (UT).

Seeing the promising results obtained from such architectures it is reasonable to think that they can be adapted or extended to accommodate also *autonomous game inspection* problems, in that autonomous agent systems could be built in a similar manner to cooperate with humans for improving and accelerating the testing phase of video games.

3. Perceptual Anomalies in Virtual Environments

In order to have a more general and comprehensive view of the Environment Inspection problem we briefly list here possible environment issues classified by typology. Such a list has been drawn up thanks to the support of the QA team of the game company THQ® Australia:

- Graphics: textures may become corrupted, may disappear or flicker, they may appear distorted or discolored. Items could appear floating in the air, sticking out or through walls. Shading may also cause effects like *aliasing* or *acne*;
- Camera: a character may be obscured by another object or seen at an awkward angle; the camera may allow *clip through* effects letting the user to view objects from inside;
- Collision: the objects in the game may intersect each other or do not intersect appropriately; barriers may exist in inappropriate places or may not exist where they should (for instance the character would be able to walk through walls and objects);
- Sound: a sound or music may not play when expected or it may contain imperfections; the game may play music or sound effects at inappropriate levels or may not play them at all;
- Framerate: when the framerate drops down the action on the screen will appear in slow motion.

All the above mentioned anomalies surely affect the play experience; hence they need to be fixed early on in the development phase so that the product can be assessed against higher level playability or entertainment issues. In the rest of this document we present an approach for dealing with the automatic detection of environment anomalies and, more generally for making the video game testing process semi-automatic.

4. General Framework

In section 2 we presented an overview of AI techniques that we identified as good candidates for approaching our Game Inspection problem. This section aims to extend

those concepts by combining those techniques with Computer Vision technologies in order to build a unified game inspection framework. To this end we will first show the way such system could fit into a real game production scenario; then we will point out two approaches that can assist the building of such a general architecture. Finally we will show how to combine different Computer Vision techniques for producing a shadow aliasing game inspection agent.

4.1. Game Inspection Agents in a real scenario

In the games production literature one of the most flexible game production processes is referred to as the *iterate-until-you-drop* method (Irish, 2005). In its original conception, the *testing* task of such workflow is the one which steers the fun part of the game towards the required direction.

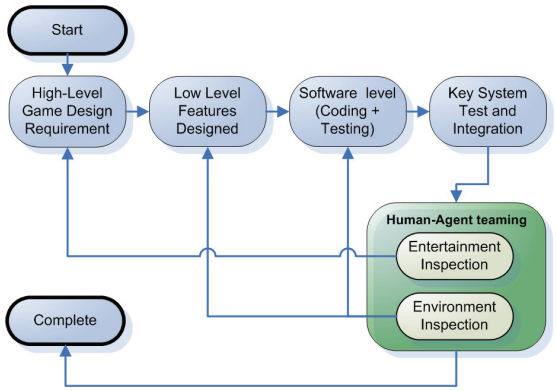


Figure 1: the *iterate-until-you-drop* method modified to accommodate the human-agent teaming coordination.

In the hypothesis of a human-agent orchestration or synchronization, we would replace that task with the two new QA factors previously introduced, namely the Entertainment and Environment inspection (see Fig.1). The AI agent software would then be coordinated by the QA team in such a way as to cope with certain tasks, such as wandering about detecting perceptual bugs (Environment Inspection) or measuring the challenge level of a map (Entertainment Inspection). Although we have shown here how such framework can fit into a real case scenario, this human-agent coordination can also be applied to any other production method in which the testing task can always be replaced by human-agent orchestration.

4.2. Two approaches for building a Game Inspection Agent

One of the early stages of any framework development process is the specification of its requirements. In the case of the human-agent coordination it would desirable for the system to be:

- abstract: shall not depend on a particular implementation or architecture;
- configurable: the model shall allow the selection of anomalies to be checked and the planning of test cases;
- agent based: once the test plan has been worked out by the QA team, the software should have autonomous, social and sensing features. Indeed it will need to make autonomous decisions when traversing and inspecting the environment, and to interact with other agents or humans for performing complex tasks or test cases;
- extendible: the model shall be easily integrated with new modules for the detection of new anomalies, and for large scale environments;
- independent from the product to test: this would avoid the design of a new testing system for each title to be released.

If the agent ideally were given all possible information about the internal architecture of the game – such as the scene-graph, the object-space and its mechanics – via some communication language, it would be able to perform a wide range of complex test cases. Indeed if the low level data such as vertices, polygons and textures were translated into high level abstract information such as symbols, terms, assertions and quantifiers, the agent could accomplish high level inference and deal with tasks of abstract meaning such as "*check the accessibility of all secret areas*" or "*kill the guard before he triggers the alarm*". If the language and interfaces used for conveying such information were also game and platform-independent our system would also be product and platform-independent. Unfortunately providing such a "symbolic translator" could be costly in terms of time and resources from the point of view of the game development. After all, the amount of information that the game should disclose to the agent and the communication language that best serves this goal are not known yet.

Nevertheless not all test cases require abstraction and context awareness to be performed. For instance we can look for environment anomalies such as shadow aliasing or image flickering in a flight simulator, an adventure game or in a first person shooter. The peculiar characteristic of the bug – such as its awkward shape or its spatiotemporal pattern – in this case can be detached from the context in which the bug appears. This is where Computer Vision comes in useful, as it provides techniques that aim at making useful decisions about real and physical objects and scenes based on sensed images (Shapiro & Stockman, 2001).

In the light of these considerations and the aforementioned software requirements, a solution to the automation of the game testing problem can be sought by following two different, although not exclusive approaches. Indeed the research can start with the investigation of communication and representation languages that best support the user-agent coordination and the building of a complete ontology of the game in terms of objects, events and relationships that can be found in the virtual world. Once such languages have been identified, techniques similar to the ones presented in Section 2 can be investigated for measuring high level entertainment issues

like challenge, fairness and curiosity of the game. We name this the *Representational approach*.

Likewise, the design can start up considering the low level information generally disclosed by video games – namely the low level output data and the user activity – and combine it with AI and Computer Vision techniques in order to provide the agent with visual bug detection capabilities. We call this approach the *Sub-Representational approach*.

Both Representational and Sub-Representational approaches aim to solve the same problem at different levels, namely at the Entertainment and at the Environment level respectively, hence for achieving a general unique solution both approaches should be further investigated.

4.3. A Sub-Representational approach for building the system

Our preliminary study has been focused on the Sub-Representational approach which aims to give the agent vision-debugger features combining techniques from AI and Computer Vision along with data disclosed by the game. To this end, in order for the agent to be attentive to visual bugs in a human-like manner, it needs to analyze the same images perceived by a human player. Through the GPU pipeline modern computer games not only disclose that information but also the data that the GPU needs for

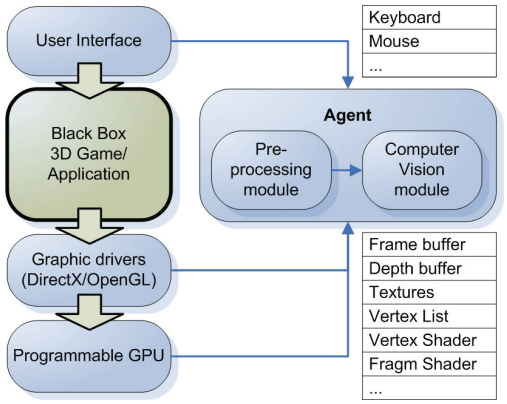


Figure 2: illustration of the Sub-Representational approach and the information available from the user activity and the GPU pipeline.

rendering that scene. This includes object-space geometry, image-space buffers, all textures and all transforms to which each vertex and pixel is subjected during the rendering process as well as source code from any shader for programming vertex and fragment processors. Moreover, through the user interface it is also possible to track the user activity during a game session (see Fig. 2).

In Computer Vision, real images are processed in order to extract some characteristic information to be used for controlling processes, detecting events and modeling the perceived environment. In a very similar way, we aim at evaluating the integrity of a virtual environment by processing the same image perceived by the human player but with the enormous advantage of working with noiseless data as it comes directly from the GPU pipeline and the drivers and not from real sensors.

Access to the graphic pipeline can be achieved by modifying the graphic drivers in such a way as to store the traffic that passes through. In this way the pipeline can be read without either modifying the application output or requiring the programmer to make extensive modifications to the code for debug.

However, to enable the agent to use this extra information effectively each bug needs to be adequately characterized. Such characterization generally consists of building a technique or a set of techniques to use for processing the image so that the bug can be recognized in an unambiguous manner. Sometimes, for bugs particularly complex to characterize it may be useful to pre-process the image in order to highlight or isolate characteristic objects or features in such a way as to facilitate the detection process, hence making it more robust. This will result in a new "characteristic" image different from the one produced by the application that the agent can use for making better inferences. In the next section we will show how to perform both the pre-processing and pattern recognition stages.

5. The Visual Bug Detection process

Following the Sub-Representational approach, our agent is designed as a two-module system: the Pre-Processing module and the Computer Vision module. The former aims at emphasizing the objects of interest in support of the second module which processes the image using the appropriate techniques for detecting the target. Note (Fig. 2) that both modules may use data from the pipeline and from the user interface if needed. In the next sections we apply our theoretical framework to a common visual bug for proving the effectiveness of the cooperation of the two aforementioned modules.

5.1. A case study: Shadow Map Artifacts

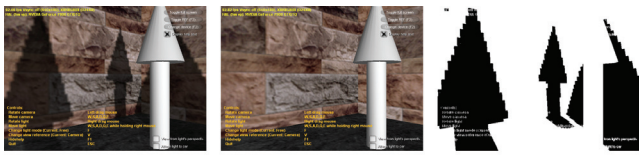
The Shadow Map is a relatively recent technique for casting shadows on arbitrary meshes. Its efficiency and versatility make this algorithm the preferred method for generating shadowing effects both in the film industry and for many computer games.

However, as it is an image-space algorithm, it is prone to various kind of artifacts that usually go under the name of *aliasing* and *acne* effects (Wolfgang, 2006). Our preliminary work focused on the detection of *perspective aliasing* effects peculiar to the standard version of the algorithm. As shown in Fig. 3 the anomaly consists of blocky shadows visible in those regions whose distance from the camera is far less than the one from the light source. The magnitude of the effect depends on the size of the shadow map chosen.

Figure 3: example of a scene affected by perspective aliasing (left) and the related preprocessed image (right) containing only shadows.

5.2. The Pre-processing stage

The obvious point of preprocessing the image for this particular bug is to remove all non-shadow objects from the image to let the CV module only process the objects of interest. This reduces both the possibility of detecting false positives and the computational load. The result of such operation is shown in Fig. 3.



(a) (b) (c)

Figure 4: extraction of shadows from an image. A XNOR operation performed between the original image (a) and a non-shadowed version of it (b) produces an image containing only shadows (c).

Although we preprocessed the image by manually modifying the pixel shader there is at least one way of making this approach automatic. A picture containing only shadows can be obtained from an exclusive-nor (XNOR) between the original shadowed image and a non-shadowed version of it. A trivial way of generating the non-shadowed image, while still using the same code for rendering the scene, is to modify the shadow map in such a way as to set all values to the global maximum of the current map plus a certain bias. In this way there will be no pixels considered in shadow as the distance of each pixel seen by the camera from the light spot will be surely smaller than the related distance written in the shadow map. This process is depicted in Fig. 4.

5.3. The Pattern Recognition stage

Perspective aliasing effects can be regarded as series of corners grouped as to form jagged edges of different length, shape and magnitude. In Computer Vision fairly robust techniques for detecting corners of different size and shape base their principle on a multi-scale descriptor of the image structure known as second moment matrix (Harris & Stephens, 1988).

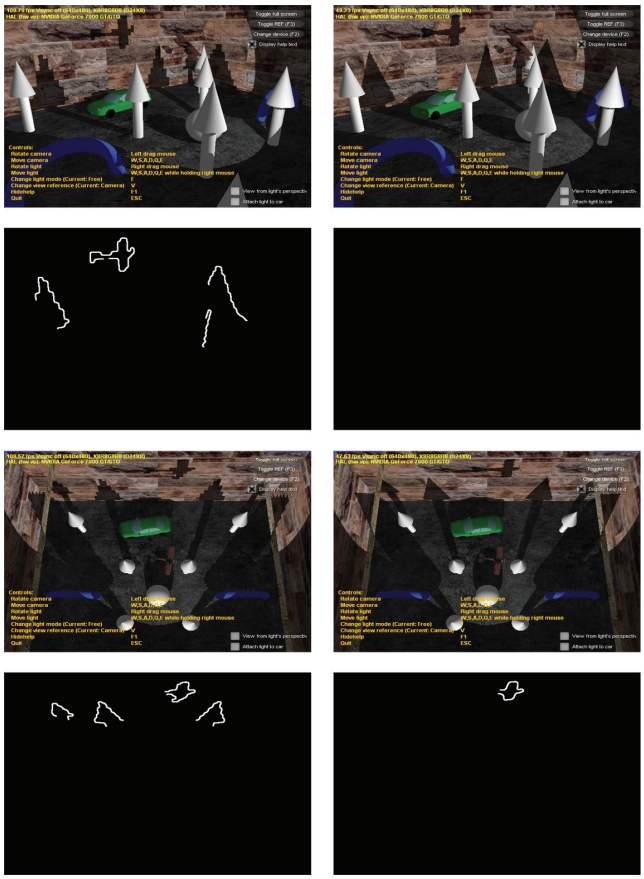
Our solution combines the Harris corner measure based on such descriptor and the Canny edge detector algorithm (Canny, 1986) for indentifying jagged edges within an image. A jagged edge can be thought of as a line bending through a set of corner points displaced within a certain distance from each other. To detect such lines it is useful to consider the Harris response R, which combines the trace $\mathrm{tr}(M)$ and the determinant $\det(M)$ of the second moment matrix:

$$R = \det(M) - k\,\mathrm{tr}^2(M)$$

where $k > 0$ is a parameter.

It can be noted that the thresholding of R produces a blob-like structure where the size of the blobs depends mainly on the size of the Gaussian filter used for convolving the matrix M. Likewise, it is easy to see that each blob will contain at least a Harris corner point.

It turns out that by choosing the proper size of the filter, it is possible to group all corners points of a jagged line within the same blob. Such a line is the Canny edge passing through those corner points. The minimum number of jags in an edge can be controlled by counting the corners nearby the detected Canny segment, whereas the minimum jag size can be parameterized by the size of the filter used for computing R. The bigger the filter, the bigger will be the smallest aliasing effect that can be detected by the algorithm. In fact, in order to detect artifacts of different magnitudes and to make the algorithm robust to small variations in sizes, we compute the second moment matrix at different scales.



(a) (b)

Figure 5: detection of the shadow map aliasing. Columns (a) and (b) show a buggy and a bug free version of the same game respectively. Rows two and four show how blocky shadows are detected (white lines) by the algorithm.

5.4. Experimental Results

Figure 5 shows some preliminary results of the two-stage approach described before. The game used as test-bed is a shadow map sample taken from the Microsoft DirectX® 10 SDK. For the extraction of the frame-buffer form the GPU pipeline we used the tool Microsoft PIX. In this paper we only show the input images from the game and the final outcome of the algorithm leaving out the intermediate pre-processed images filtered out from non-shaded textures. In column (a) the shadows are affected by aliasing due to the too low resolution of the shadow map (112x112). Column (b) shows the same scene rendered with a higher shadow map resolution (512x512) in order to reduce the artifact. The behavior of the algorithm is shown in rows two and four; the white lines indicate those regions recognized as anomalies. It is interesting to note that since the algorithm processes only the final frame-buffer, its efficiency and detection rate do not depend on the number of light spots, nor on the complexity of the environment.

6. Conclusion and Future Work

We have introduced a framework for making video game testing a semi-automatic process by combining Artificial Intelligence and Computer Vision technology. Such a Framework needs to be built on the basis of two different, non exclusive approaches that bring a unique, robust and complete solution. The results we have achieved following the Sub-Representational approach provide evidence that an agent featuring Computer Vision capabilities can effectively deal with the detection of some environment anomalies. Moreover agents provided with GPU-Vision Debugger features, combined with the monitoring of the user activity can result in useful automatic regression test tools. For instance, the agent can replicate the user actions, previously tracked in an older version of the game, to check for visual anomalies in a never build of the same game. This is a way to make the regression test process for environments completely automatic with no need for any other information about the internal architecture of the game. However, the pattern detection capabilities of the agent may not suffice for test cases belonging to what we have called Entertainment Inspection. To include these testing activities, the Representational Approach needs to be investigated to enable the agent to make more abstract inferences and therefore deal with tasks that require planning of activities, knowledge of the environment and awareness of the game typology under test.

7. References

Canny, J. (1986). A computational approach to edge detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Vol. 8, pp. 679-698): IEEE Computer Society.

Denzinger, J., Chan, B., Gates, D., Loose, K., & Buchanan, J. (2004). *Evolutionary behavior testing of commercial computer games*. Paper presented at the 2004 IEEE Congress on Evolutionary Computation.

Desurvire, H., Caplan, M., & Toth, J., A. (2004). *Using heuristics to evaluate the playability of games*. Paper presented at the CHI '04 extended abstracts on Human factors in computing systems.

Harris, C., & Stephens, M. (1988). *A combined corner and edge detector.* Paper presented at the ALVEY Vision Conference, Cambridge, UK.

Irish, D. (2005). *The Game Producer's Handbook*. Boston, MA: Course Technology Press.

Laird, J. E. (2001). Using a Computer Game to Develop Advanced AI. *Computer, 34*(7), 70-75.

Macleod, A. (2005). *Game design through self-play experiments*. Paper presented at the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology.

Merrick, K., & Maher, M. L. (2006). *Motivated reinforcement learning for non-player characters in persistent computer game worlds*. Paper presented at the ACM SIGCHI international conference on Advances in computer entertainment technology.

Riley, D. M. (2007). 2006 U.S. Video Game and PC Game retail sales reach $13.5 billion exceeding previous record set in 2002 by over $1.7 billion. *The NPD Group Press Release* Retrieved August 15th 2007, from http://www.theesa.com/

Shapiro, L. G., & Stockman, G. C. (2001). *Computer Vision*. Upper Saddle River, New Jersey: Prentice Hall.

Urlings, P., Sioutis, C., Tweedale, J., Ichalkaranje, N., & Jain, L. (2006). A future framework for interfacing BDI agents in a real-time teaming environment. *J. Netw. Comput. Appl., 29*(2), 105-123.

Wolfgang, E. (2006). *Shader X4: Advanced Rendering Techniques*. Hingham, MA: Charles River Media, Inc.

Wooldridge, M. J., & Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review, 10*(2), 115-152.

Yang, K., Marsh, T., & Shahabi, C. (2005). *Continuous archival and analysis of user data in virtual and immersive game environments*. Paper presented at the ACM workshop on Continuous archival and retrieval of personal experiences.

Yannakakis, G. N., & Hallam, J. (2006). Towards capturing and enhancing entertainment in computer games. In *Advances in Artificial Intelligence, Proceedings* (pp. 432-442). Berlin: Springer-Verlag.