Scaling the Level of Difficulty in Single Player Video Games

Maria-Virginia Aponte, Guillaume Levieux, and Stéphane Natkin

Centre d'Etudes et de Recherches en Informatique du CNAM
Conservatoire National des Arts et Métiers, Paris, France
forename.lastname@cnam.fr

Abstract. In this this paper, we discuss the interest and the need to evaluate the difficulty of single player video games. We first show the importance of difficulty, drawing from semiotics to explain the link between tension-resolution cycles, and challenge with the player's enjoyment. Then, we report related work on automatic gameplay analysis. We show through a simple experimentation that automatic video game analysis is both practicable and can lead to interesting results. We argue that automatic analysis tools are limited if they do not consider difficulty from the player point of view. The last section provides a player and Game Design oriented definition of the challenge and difficulty notions in games. As a consequence we derive the property that must fulfill a measurable definition of difficulty.

Keywords : video games, challenge, difficulty, learning, evaluation.

1 Introduction

One of the fundamental issues to tackle in the design of video games is mostly referred as *creating a well-shaped difficulty curve*. This means that one of the core element of a good game design is to make the game just as difficult as it has to be, so that the player feels challenged enough, but not too much. However, game creators cannot rely on strong tools to help them in this task, and there is not even a clear and accepted definition of difficulty as a measurable parameter. For now, game difficulty adjustment is a subjective and iterative process. Level / game designers create a sequence of challenges and set their parameters to match their chosen difficulty curve. Finding the right sequence and tune every challenge relies on playtesting performed by the designers, and, at some important milestones, by focus test groups. Focus tests are costly and it's very hard for a designer to evaluate the difficulty of a challenge he created and played for many hours. Our goal is to provide a clear, general and measurable definition of the difficulty in games. We must rely on accepted definitions of video games and works which relates the difficulty in games to the quality of the games, as perceived by the player. We present related work on automatic gameplay analysis, and then report a first experiment with a basic synthetic player. Finally, we define difficulty, tacking into account the player experience.

S. Natkin and J. Dupire (Eds.): ICEC 5709, pp. 24–35, 2009.
© IFIP International Federation for Information Processing 2009

2 Scaling the Difficulty

Difficulty scaling is a fundamental part of game design [1] [2]. However, this is not an obvious consequence of accepted definitions of video game. Jesper Juul has listed many of them and has proposed a synthesis [3]. We start from Juul's definition to explain why difficulty scaling is so important in game design :

> *'A game is a rule-based formal system with a variable and quantifiable outcome, where different outcomes are assigned different values, the player exerts effort in order to influence the outcome, the player feels attached to the outcome, and the consequences of the activity are optional and negotiable.'*

This definition gives a clear, precise idea of how a game system behaves, and manages to take into account the most interesting parts of the previous definitions. But for our purpose, we must explain more precisely why difficulty is such an important thing. The fact that the player *exerts effort in order to influence the outcome*, and *feels attached to the outcome* is the core point. To point out the important components of a gameplay, and foremost the link between caring about difficulty and making a good game, it is necessary to coin a definition that leaves aside the game's dynamics structure and focuses on video games from the player's point of view.

Robin Hunicke describes a game using a *Mechanics, Dynamics and Aesthetics (MDA)* framework [4]. Mechanics are the tools we use to build a game (e.g. physics engines, pathfinding algorithm...), Dynamics describes the way the Mechanic's components behave in response to the player, and Aesthetics is the desirable emotional responses evoked to the player. Of course, the design goals is the Aesthetics, that is to say the player's emotions. We argue that the difficulty of challenges greatly influences video game's aesthetics and thus play a central role in game design.

Umberto Eco's book *The open work* is a fundamental research about interactive art's aesthetics [5]. Umberto Eco states that when we face a piece of art, we are interpreting it, seeking patterns, looking for information. Depending on our culture and knowledge, we will find something to grab on within the stimulating field of the piece of art. But then we will go further, and find another interpretation and feel lost for short moment, while shaping our new pattern. Moreover, when a piece of art is interactive, the aesthetic value comes both from the tension resolution and from the fact that this resolution is a consequence of our choice. Assuming that a video game is an open work we can propose a similar analysis. Every time the player faces an obstacle, he gets lost for a few seconds. Then he finds and choose a pattern, press the right buttons, and takes pleasure both from resolving a tension and from making a choice. Thus, we can draw from Umberto Eco's work that in video games, challenge is fundamental because it creates tension situations that the player has to solve and the opportunity of meaningful choices.

Related work on video game player's enjoyment support our analysis and place challenge at the center of video game's aesthetics. In his book *A Theory of Fun*

for Game Design, Ralph Koster states that we have fun playing games when we discover a new pattern, i.e. a strategy that we apply to overcome a challenge [6]. Sweetser and al see challenge as one of the most important part of their *Game Flow* framework [7]. Yannakakis et al measure player enjoyment from challenge, besides behavior and spatial diversity [8].

Mihaly Csikszentmihalyi's Theory of Flow, that researchers have applied to video game as a measure of the player's enjoyment, helps us to make a link between the difficulty of a challenge and the player's enjoyment [9] [10] [7]. A player is in a *Flow* status, and thus enjoying the game, when the task is neither too hard nor too easy. It is thus not enough to create tensions situations and to give the player choices to resolve this tension, a good game design must accurately scale the difficulty of a challenge to have a tension level that leads to the player's enjoyment. Thus, a definition of a game from the Aesthetic point of view and centered on challenges could be:

> *'Regarding challenges, the Aesthetics of a game is created by tension-resolution cycles, where the tension is kept under a certain threshold, and where the resolution of a cycle depends on the player's choices.'*

This definition doesn't take into account every aspect of game aesthetic but is focused on challenge, that most studies consider as a core component of game's aesthetics. Tension situations that the player seeks and try to solve have been created by the game designer and the amount of tension they deliver directly stems from their complexity. As a result, difficulty scaling is a central task of a good game design. Games already propose different difficulty levels [1], and sometimes even Dynamic Difficulty Adjustment [2], manipulating some specific parameters of the gameplay in real time [4], or automatically scaling the game AI capacity [11]. But whichever difficulty scaling method the game designer uses, he must still tune them properly. It is sometimes really hard to guess to which extent a change in a low level parameter will just make the game a bit harder or dramatically change the gameplay [1], and tuning is one of the most time consuming area in game AI development [12]. This is this design process that we want to shorten by providing tools that will help game designers evaluating the impact of any difficulty scaling parameter on the final difficulty curve. To create good gameplay, it's then fundamental to provide game designers with strong tools and a definition of difficulty as a measurable parameter.

3 Related Work: Testing with a Synthetic Player

Our goal is to evaluate a parameter or a set of parameters that can be considered as a measure of a game difficulty. There are two theoretical approaches to evaluate such a parameter. The first way is to find, according to the game structure, a mathematical expression of the parameter and to solve the corresponding equations. The complexity of a game and the notion of difficulty tends to show that this approach is not practicable. A second solution is to experiment the game and measure the parameter. To experiment the game we may either

use a real or a synthetic player. The main advantage of a real player is that he behaves like a real player. In counterpart he plays slowly, becomes tired and his behavior is only known through the game interface. The synthetic player is tireless, plays quickly and his behavior can be fully understood. The design of the synthetic player allows to simulate some foreseen behavior of a real player (risky or careful, for example) and some simple learning techniques.

Gameplay testing has already been the subject of many interesting researches. Alasdair Macleod studied gameplay testing of Perudo, a bidding dice game, simulating plays with a multi-agent system [13]. He wanted to modify Perudo's gameplay to make it more fair, and added a rule he thought it would help loosing players to stay in the game. Running the experiment and analyzing the modified game, he obtained the counter-intuitive result that the rule was not helping loosing players at all. These results shows that self-play testing can help testing gameplay modifications.

Neil Kirby analyzed Minesweeper, replacing the player by a rule based AI [14]. Each rule was related to a different play complexity. He found out that Minesweeper was surprisingly not as hard as he supposed it to be, as the most part of the board was often solved using only the very simple rule. These results point out that automated techniques can provide interesting approaches to study video game difficulty.

Both Perudo and Minesweeper are simple games, but automated analysis can also be applied to complex off-the-shelf games. Bullen et al used Unreal Engine (Epic Games) and created a gameplay mutator providing sensors to log useful game events [15]. They tested Unreal Tournament 2004 (Epic Games) using partial and fully automated testing (i.e. both during player vs AI and only AI games). They pointed out that fully automated tests had to be done with a specific AI, because standard AI was not aggressive enough. The fact is that standard Unreal Tournament AI has been created to entertain the player, not to mimic his behavior, and thus is not able to fully explore the gameplay. Recently, Lankveld et al proposed to analyze a game difficulty using incongruity, the distance between the actual dynamics of the game and the mental model the player has built [16]. They plan to infer the complexity of the player's mental model, and thus the difficulty of the game, by monitoring his actions. These works show that, to be useful, a synthetic player must simulate in some way a real player.

Automated game analysis can be done at several levels. Nantes et al distinguish Entertainment Inspection (i.e. gameplay testing), Environment Integrity Inspection (i.e. Sounds, graphics related issues) and Software Inspection [17]. Their system targets Environment Integrity Inspection, using Computer Vision, and especially corner detection to detect aliasing issues in shadows rendering. This is a complementary approach to the one we propose, and Nantes et al acknowledge the need of analysis tools at every inspection level.

As we argued in the previous section, Machine Learning is particularly interesting in automated gameplay testing. If we want the synthetic player to test behaviors we didn't think about before, then it must explore the game state

space by himself. Many researchers explore how machine learning can be helpful to video game development, and especially concerning automated testing. Chan et al used a Genetic Algorithm to create sequences of actions corresponding to unwanted behavior in FIFA-99 (EA Games) [18]. They also consider that the game dynamics is too complex too be fully formalized, because of huge branching factor, indeterminism and the fact that even designers never formally define it. There is thus a need to build an AI driven agent to explore this dynamics, here using evolution techniques. Spronck et al also took the same approach, making neural networks evolve to test a simplified version of the spaceships game PICOVERSE, providing an insightful analysis of its AI driven opponents [19].

Automated learning approaches becomes inadequate when it comes to creating characters with complex behaviors automatically from scratch, as sated John E. Laird [20]. But many researchers use games to evaluate machine learning techniques. The game is often considered as a reference problem to be solved by the AI. Pacman (Namco) [21], for example, has been the subject of many researches, applying various techniques to create synthetic players, from Neural Network Evolution [22] [23] [24], to Reinforcement Learning [25], Genetic Programming [26] and genetic evolution of a rule-based system [27]. Yannakakis et al [8] takes another point of view. They use synthetic characters to maximize player enjoyment, and validate their measure of enjoyment, based on challenge, behavior diversity and spatial diversity . These results show that machine learning techniques can be useful when analyzing a gameplay.

4 Case Study

4.1 The Experiment

These sections present an analysis of a Pacman-like predator-prey computer game. We built a simplified version of Pacman, using only one ghost chasing Pacman. Both Pacman and the ghost use A* pathfinding, the whole graph and shortest path being built at startup and stored to save calculation power. The challenge is to eat a maximal number of pellet without being killed by the ghost. The synthetic player has a partial view of the game. It knows five parameters. The first one had four values, giving the direction of the nearest pellet. The four other dimensions describe the four Pacman directions. For each direction, Pacman knows whether he is facing a wall (0), a ghost one (1) or two (2) step away from him, a free-of-ghosts path less than 18 steps long (3), or free-of-ghosts path longer than 18 steps long (4). We choose this game state abstraction because we consider that the main information that a real player uses is the path to the nearest pellet and the safety of the fourth direction he can take.

The only ghost in the maze, Blinky, has been programmed to chase the player, taking the shortest path to reach him. In Pacman original rules, ghosts periodically enter *scatter mode* and stop chasing the player to go back to their respective board corner. But as a first step, we wanted to maintain the rules at their minimum complexity, so that results could be more easily interpreted. The synthetic player AI was programmed with a Markov Decision Process, using reinforcement

learning with Q-Learning algorithm with eligibility traces ($Q(\lambda)$) [28]. Parameters for $Q(\lambda)$ were $\alpha = 0.1$, $\gamma = 0.95$, $\lambda = 0.90$. We balanced exploration and exploitation using ϵ-greedy action selection algorithm, with $\epsilon = 0.05$.

We consider the analysis of Pacman difficulty according to a single gameplay parameter: the player speed. The number of pellets eaten by Pacman is our difficulty evaluation function. We choose this parameter because in Pacman original gameplay, ghosts / Pacman relative speed is already used to scale difficulty [21]. Every 14 frame, Blinky changes its position, moving one step up, down, left or right. Each step is 16 pixel long, the distance between two pellets. Besides going up, down, left or right like the ghost, the player also has the option to do nothing and stay at the same place. We tested the synthetic player's performance for different gameplay configuration.

4.2 Results

We run six experiments with speed varying from 0 (i.e. Pacman and the ghost move every 14 frame) to 7 (i.e. Blinky still moves every 14 frames but Pacman moves every 7 frame). We let the synthetic player develop his policy during 50000 games, Pacman having 3 lives per game.

The figure 1 presents a synthetic view of these results. What we can extrapolate from these is that modifying Pacman's speed, the difficulty tends not to be modified in a linear way. There is much less difference in Pacman score between speed 0 and 5 than between speeds 5 and 7. Such an analysis could be useful for a game designer when tuning the difficulty. He can understand that when Pacman speed gets closer to twice the ghost speed, then the games get really easier. Between 0 and 5, difficulty raises almost linearly.

4.3 Critical Analysis

These results show that it is possible to evaluate a difficulty curve, for a given game with a given challenge whose difficulty can be tuned according to a given parameter. However, this experiment is just an attempt to describe difficulty for a specific game. It doesn't give us a general framework we could apply to any game to measure its difficulty. The next step is thus to find a general and precise definition of the difficulty.

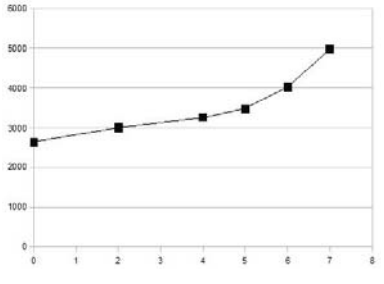

Fig. 1. Pacman score at different speeds - 5000 last games mean value

5 The Meaning of Challenge and Difficulty

To this point we have used the term of difficulty in games without providing any definition of this word. This is a difficult point. First, one cannot talk about difficulty without referring to the ability to perform an action in a given situation. We shall not try to give a general definition of difficulty covering a wide range of psychological aspects from emotional problems to intellectual and physical challenges. We consider the notion of difficulty in the sense used in Game Design.

5.1 Challenges

In all these cases, a level of difficulty is related to the player skills or abilities to overcome a given challenge. We must first define the notion of challenge in games. Starting from Juul's definition, we can consider that a video game challenge is by itself a sub-game: a rule based system with variable and quantified outcomes. According to the quantification, some of the outcomes may be considered either as a success or a failure. A general definition of the difficulty has to take into account that the quantification can be binary (WIN, LOSE) or discrete (from 0 to N points). But the definition must not depend on the units chosen, only on the relation between the outcome value and the feeling of victory or defeat. This relation is an important design choice and is explicitly implemented as a feedback in the game. Thus, we consider that in all cases the designer can define a binary function that can decide whether the player has won or lost.

The notion of difficulty is also related to time. At a given time a player may be still trying to overcome the challenge or has either won or lost. The time to overcome the challenge is also related to its difficulty: one may take hours to solve a difficult chess problem and a few minutes to solve a simple one. But the time to overcome a challenge is also relative to the conditions of the challenge itself. The player has a few seconds to choose an attack in a fighting game, a few minutes to choose a move in a RTS (Real Time Strategy) game and an unbounded delay to take a decision in a turn by turn game. In certain games the time is a criteria of evaluation: you have five minutes to find a bomb before explosion, your rank in a ski slalom game (as in WII FIT) depends on the time taken to finish the slalom. But in all cases, we can assume that the designer knows the minimal time needed to decide whether the challenge is overcome or not.

This leads to the two following definitions: *A video game challenge is a dynamic rule based system with two outcomes WIN or LOSE. At a given time t a challenge can be in one of the four possible states NOT STARTED, IN PROGESS, WIN, LOSE, according to the following automaton (Fig. 2).*

A solution of a challenge is a sequence of player's actions and the corresponding game feedback that leads the automaton from the NOT STARTED state to the WIN state.

In games solving a challenge may imply to solve a set of sub challenges. The structure of the game as a set of quests and levels is a logical, topological and, as a consequence, temporal combination of challenges (see [29] for a formal definition

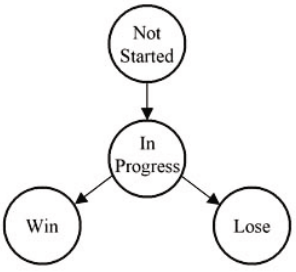

Fig. 2. Automaton of a challenge

of this organization). Consider two challenges a and b and a game where the player must solve a to solve b, a is said to be a *sub-challenge* of b. When the player knows how to solve b, he knows how to solve a. As a consequence any solutions of b includes at least one of the solutions of a.

5.2 Progression of Difficulty

Consider now the type of challenges that are used in games. The choice of challenges and the succession of challenges is related to the flow principle explained in section 2. In many Game Design books, the progression of tension cycles is presented using the Learning/Difficulty curves [30],[31]. At any time of the game, the difficulty of the next challenge must be a little higher than the current level of the player apprenticeship. When he wins the challenge and the tension decreases, the player gets new skills and ability. This correlated progression of skills abilities and difficulty must be kept all along the game.

The same idea is expressed by Jesper Juul using the notion of repertoire of methods [32]

> *'At every instant within a game, a player has created for himself a collection of methods and strategic rules which he has devised and which he applies (the player's repertoire). One strength of a good game is to constantly challenge the player, which in turn leads him to constantly find new strategies, apart of those already in the repertoire'*

There are two ways to control the difficulty: the progression of skills and the mix of challenges. The progression of skills relates the difficulty of a given challenge according to a set of parameters. This notion of difficulty is related to an idea of complexity: what type of problem a human "processor" is able to face taking into account his level of practice. How far can he move buttons, memorize configuration, how precise can be his shot, how long can he stay on one foot? As in any sport or mental exercise, the player's practice enhances his skills, and the same challenge can be solved using parameters chosen to increase the level of difficulty.

The second way to control the difficulty is to mix challenges. The solution of many game challenges relies on mastering of a set of basic techniques and then

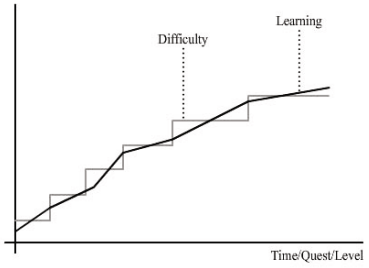

Fig. 3. Difficulty and learning curves

to try to combine them. In strategy games, you master first the strength and movements of units, then the position of production units, then the technological evolution. At each level of a game, the player understands new mechanisms, then slowly combines them. It is the same with basic attacks and combo in fighting games, with group strategy in FPS, and, at last but not least, the increasing complexity of puzzles in adventure games.

Consider the three following challenges: (A) Send a ball in a basket. For a given ball the difficulty of A decrease with the size X of the basket. (B) Press a button when a given event occurs. The difficulty decreases with the accepted error E between the date of the event and the moment when the button is pressed. (C) For given X and E, sending a ball in a basket when a given event occurs is more difficult than A followed by B.

We may state that in a game, the progression of difficulty relies on two sets of challenges:

- A set of basic challenges whose complexity, can be controlled through a set of parameters.
- An ordered set of challenges. The atoms challenges are of the first type. The solutions of higher level challenges can be deduced from those of lower level challenges.

5.3 The Difficulty Evaluation Function

Let us set up some properties that must fulfill the difficulty of a challenge a, $D(a)$:

- D must be measurable using for example a tool able to record the internal states of the game.
- D must allow comparing the difficulty of two challenges, at least of the same "kind" (jumping in platform game, for example)
- D must be relative to the game history, in particular to the progression of the player's skill according to the set of challenge already overcome.
- D must depend on the time used to finish the challenge.

Let A be the set of all challenges that have been solved before time 0. We define $LOSE(a,t)$ and $WIN(a,t)$ as the following events:

- $LOSE(a,t)$ = the automaton of a reaches the state LOSE before time t, starting at time 0.
- $WIN(a,t)$= the automaton of a reaches the state WIN before time t, starting at time 0.

We propose a definition of the difficulty D as a conditional probability:

$$D(a,t) = Probability\{LOSE(a,t)/A\}$$

The Easiness E of a can be also defined in the same way:

$$E(a,t) = Probability\{WIN(a,t)/A\}$$

At all time $D(a,t) + E(a,t) \leq 1$. We can also consider the steady state difficulty and easiness:

$$D^*(a) = \lim_{t\to\infty} D(a,t) \text{ and } E^*(a) = \lim_{t\to\infty} E(a,t)$$

If challenge a must necessarily be finished in the game $D^*(a) + E^*(a) = 1$. These functions gives us two kind of information about the challenge difficulty. First, E^* gives us the difficulty of the challenge in term of the probability that a player overcomes it. But we also can be more precise and with $E(a,t)$, get the probability that a player has to overcome the challenge before time t. We assume that designers are able to implement in the game code some triggers associated to the transitions in each challenge automaton during a test performed by players. The time needed to perform a challenge and the fact that a challenge has been successful or not can be recorded.

But in an open game, there is a small chance that two players will reach the same challenge following the same history. Hence the A of the two players will be different. So, it is necessary to drive the tester with a walk-through. In this case the Difficulty and the Easiness can be statistically estimated, and, under some ergodic assumptions $D^*(a)$ and $E^*(a)$ also.

This can lead to validate experimentally numerous assumptions about the learning and the difficulty curves. For example, if a is a sub-challenge of b then $D^*(a) < D^*(b)$. In the same case, if the player plays twice a, even if he loses the first execution, the second one should be less difficult. Lets denote (a *knowing* a) this second challenge:

$$D^*(a \textit{ knowing } a) \leq D^*(a)$$

If a is a sub-challenge of b and if the player has already played a before b then

$$D^*(b \textit{ knowing } a) \leq D^*(b)$$

More generally, this can lead to validate experimentally the theory of learning and difficulty curves.

6 Conclusion

One of the core component of a good gameplay is the good tuning of the challenge difficulty. In this paper, we have presented the link between challenges

and player's enjoyment, in term of tension-resolution cycles. However, there is a lack of a general definition of the difficulty in games, methodology and tools to measure it. We have reported a first experiment using a synthetic player. This experiment shows that with a basic AI driven player, we can extract objective difficulty measures out of a simple game. But as we stated before, the player ability to overcome a challenge depends on his experience of the game. Thus, we propose a definition of difficulty taking into account the past experience of the player. This definition relies on the main properties presented in this paper. This function is measurable as long as the game design provides a clear specification of challenges. The next step of our research is to implement, in different types of games, and to experiment the evaluation of the function using real players. This will lead to an experimental validation of the apprenticeship and learning curves.

References

1. Boutros, D.: Difficulty is difficult: Designing for hard modes in games. Gamasutra (2008), `http://www.gamasutra.com/` (last access 01/2009)
2. Adams, E.: The designer's notebook: Difficulty modes and dynamic difficulty adjustment. Gamasutra (2008), `http://www.gamasutra.com/` (last access 01/2009)
3. Juul, J.: The game, the player, the world: Looking for a heart of gameness. In: Copier, M., Raessens, J. (eds.) Level Up: Digital Games Research Conference Proceedings, pp. 30–45 (2003)
4. Hunicke, R.: The case for dynamic difficulty adjustment in games. In: Advances in Computer Entertainment Technology, pp. 429–433 (2005)
5. Eco, U.: L'oeuvre ouverte. Seuil (1965)
6. Koster, R.: A Theory of Fun for Game Design. Paraglyph Press, Scottsdale (2005)
7. Sweetser, P., Wyeth, P.: Gameflow: a model for evaluating player enjoyment in games. Comput. Entertain. 3(3), 3 (2005)
8. Yannakakis, G.N., Hallam, J.: Towards optimizing entertainment in computer games. Applied Artificial Intelligence 21(10), 933–971 (2007)
9. Csikszentmihalyi, M.: Flow: The Psychology of Optimal Experience. Harper Perennial (March 1991)
10. Jsselsteijn, W., de Kort, Y., Poels, K., Jurgelionis, A., Belotti, F.: Characterising and measuring user experiences in digital games. In: International Conference on Advances in Computer Entertainment Technology (2007)
11. Andrade, G., Ramalho, G., Santana, H., Corruble, V.: Extending reinforcement learning to provide dynamic game balancing. In: IJCAI 2005 Workshop on Reasoning, Representation, and Learning in Computer Games, pp. 7–12 (2005)
12. Scott, B.: Architecting a game ai. In: AI Game Programming Wisdom 1, Charles River Media, Inc. (2002)
13. Macleod, A.: Game design through self-play experiments. In: ACE 2005: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology, pp. 421–428. ACM, New York (2005)
14. Kirby, N.: Ai as gameplay analysis tool. In: Game Programming Wisdom 4. Course Technology, Cengage Learning, pp. 39–49 (2008)
15. Bullen, T., Katchabaw, M.J., Dyer-Witheford, N.: Automating content analysis of video games. In: Canadian Game Studies Association (CGSA) Symposium (2006)

16. van Lankveld, G., Spronck, P., Rauterberg, M.: Difficulty scaling through incongruity. In: Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, Stanford, California, USA, October 22-24 (2008)
17. Nantes, A., Brown, R., Maire, F.: A framework for the semi-automatic testing of video games. In: Artificial Intelligence and Interactive Digital Entertainment Conference. AAAI, Menlo Park (2008)
18. Chan, B., Denzinger, J., Gates, D., Loose, K., Buchanan, J.: Evolutionary behavior testing of commercial computer games. In: Congress on Evolutionary Computation, CEC 2004, June 2004, vol. 1, pp. 125–132 (2004)
19. Spronck, P.: Evolving improved opponent intelligence. In: GAME-ON 3rd International Conference on Intelligent Games and Simulation, pp. 94–98 (2002)
20. Laird, J.E.: Game developers magazine (2000)
21. Pittman, J.: The pac-man dossier. Gamasutra (2009), `http://www.gamasutra.com/` (last access 01/2009)
22. Yannakakis, G.N., Hallam, J.: Evolving opponents for interesting interactive computer games. In: Animals to Animats 8: Proceedings of the 8th International Conference on Simulation of Adaptive Behavior (SAB 2004), Santa Monica, CA, USA, pp. 499–508. MIT Press, Cambridge (2004)
23. Lucas, S.M.: Evolving a neural network location evaluator to play ms. pac-man. In: Proceedings of the 2005 IEEE Symposium on Computational Intelligence and Games, CIG 2005 (2005)
24. Gallagher, M., Ledwich, M.: Evolving pac-man players: Can we learn from raw input? In: Computational Intelligence and Games, CIG (2007)
25. Bonet, J., Stauffer, C.: Learning to play pac-man using incremental reinforcement learning (2001), `http://www.ai.mit.edu/-people/-stauffer/-Projects/-PacMan` (accessed December 5, 2008)
26. Rosca, J.P.: Generality versus size in genetic programming. In: Genetic Programming 1996: Proceedings of the First Annual Conference, pp. 381–387. MIT Press, Cambridge (1996)
27. Gallagher, M., Ryan, A.: Learning to play pac-man: an evolutionary, rule-based approach. In: Evolutionary Computation (CEC 2003), vol. 4, pp. 2462–2469 (2003)
28. Watkins, C.J.: Learning from Delayed Rewards. PhD thesis, Cambridge (1989)
29. Natkin, S., Vega, L.: A petri net model for computer games analysis. Int. J. Intell. Games & Simulation 3(1), 37–44 (2004)
30. Natkin, S., Delocque-Fourcaud, A.M., Novak, E.: Video Games and Interactive Media: A Glimpse at New Digital Entertainment. AK Peters Ltd. (2006)
31. Byrne, E.: Game Level Design. Game Development Series. Charles River Media (December 2004)
32. Juul, J.: Half-Real: Video Games between Real Rules and Fictional Worlds. MIT Press, Cambridge (2005)