Entertainment Computing 2 (2011) 205–213

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: ees.elsevier.com/entcom

Measuring the level of difficulty in single player video games

Maria-Virginia Aponte, Guillaume Levieux *, Stephane Natkin

Centre d'Etudes et de Recherches en Informatique du CNAM, Conservatoire National des Arts et Métiers, Paris, France

ARTICLE INFO

Article history:
Received 22 June 2010
Revised 8 April 2011
Accepted 9 April 2011
Available online 22 April 2011

Keywords:
Video games
Challenge
Difficulty
Learning
Evaluation

ABSTRACT

In 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 two sections provide a player and Game Design oriented definition of the challenge and difficulty notions in games. As a consequence we derive the property that must fulfil a measurable definition of difficulty.

1. Introduction

One of the fundamental issues to tackle in the design of video games is mostly referred to 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 and game designers create a sequence of challenges and set their parameters to match their chosen difficulty curve. Finding the right sequence and tuning every challenge rely mainly on playtesting performed by the designers. Playtesting is a heavy time consuming task, 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 relating the games difficulty to the games quality, as perceived by the player. We present related work on automatic gameplay analysis, and then report a first experiment with a basic synthetic player. We then define difficulty, taking into account the player experience and a function of time. To conclude, we propose a way to explore the link between the player abilities and the probability to lose a challenge, providing an interesting measure for the game designer to explore the difficulty of his game's challenges.

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 a primary component of any gameplay. 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

* Corresponding author.
E-mail addresses: maria-virginia.aponte_garcia@cnam.fr (M.-V. Aponte), guillaume.levieux@cnam.fr (G. Levieux), stephane.natkin@cnam.fr (S. Natkin).

1875-9521/$ - see front matter
doi:10.1016/j.entcom.2011.04.001

the difficulty of challenges greatly influences video game's aesthetics and thus plays 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 and 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 chooses a pattern, presses 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 tension 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 does not 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 multiple 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 the 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. 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 tend to show that this approach is not practicable. The second solution is to experiment the game and measure the parameter. To experiment the game, we may use either a real or a synthetic player. The main advantage of using a real player is that it demonstrates human behaviors. 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 which he thought would help losing players to stay in the game. By running the experiment and analyzing the modified game, he obtained the counter-intuitive result that the rule was not helping the 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 that were not under consideration before, 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. use 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 to be fully formalized, due to the reason 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 [22–24], 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 were built and stored 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.

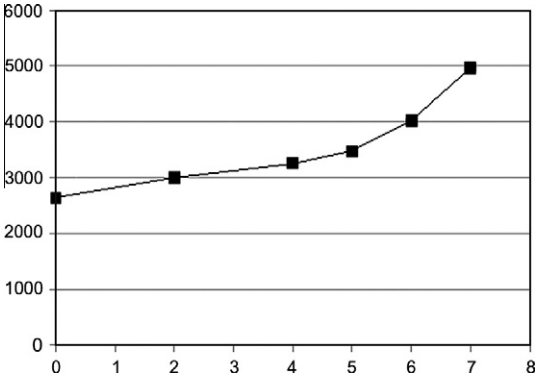

Fig. 1. Pacman score at different speeds – 5000 last games mean value.

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 Fig. 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 does not 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.

5. The meaning of challenge and difficulty

At this point we have used the term of difficulty in games without providing any definition of this word. 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. Instead, we consider the notion of difficulty in the sense used in game design, that is video games are built as combinations or sequences of predefined *challenges*. Thus, the study of the overall difficulty for a given video game involves studying the difficulty players experiment when overcoming individual challenges in that game.

In this section we give a precise definition of challenges in video games and explain how the progression of difficulty while playing a given path of challenges in a video game can be seen as the progression of the difficulties for each challenge in the sequence of challenges the player actually attempt to overcome. We discuss on the aspects a measurable notion of challenge difficulty must involve and we end by defining this notion as a conditional probability.

5.1. Challenges and their difficulty

In game playing, 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 must allow considering either binary (*WIN*, *LOSE*) or discrete (from 0 to *N* points) quantification. Without losing generality, we consider that in all cases the designer can define a binary function that can decide whether the player has won or lost, and thus, we adopt a binary quantification in the rest of this section.

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 subchallenges. 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 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

In video 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 as depicted in Fig. 3 [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 abilities. This correlated progression of skills abilities and difficulty must be kept all along the game.

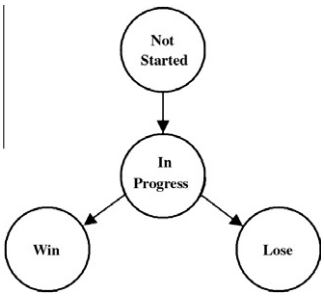

Fig. 2. Automaton of a challenge.

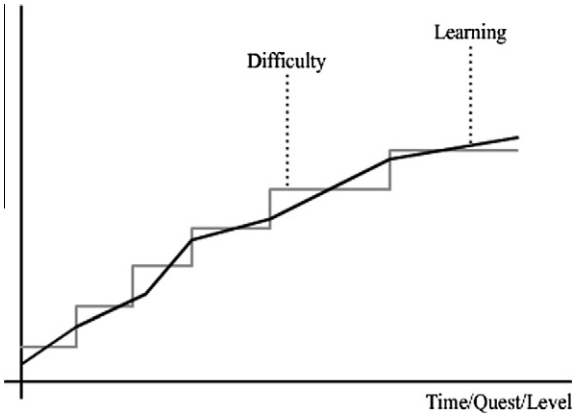

Fig. 3. Difficulty and learning curves.

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'.*

Thus, we propose to study the difficulty of each challenge played in a given sequence as a mean to follow the progression of difficulty for *that particular path in the game*.

We distinguish two ways of controlling the difficulty progression: (a) the progression of skills and (b) the combination of challenges. The progression of skills relates the difficulty of a given challenge according to the mastering of a given set of parameters. Here, difficulty relates 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 of controlling the progression of difficulty we identify is by combining several challenges. The solution of many game challenges rely on mastering a set of basic techniques and then in trying 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.

As an illustration, 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. The point here is that challenge (C) is a combination of challenges (A) and (B), such that the individual mastering of the two latter allow the player to increase his skills and eventually overcome (C).

Based on these assumptions about what we could call, the *composionality of challenges*, we consider that in a game, the progression of difficulty relies on the use of two sets of challenges:

- A set of *basic* or *atomic* challenges, whose complexity can be controlled through a set of parameters.
- A partially ordered[1] set of *composite* challenges, built on top of *atomic* challenges. The solutions of composite challenges can be deduced from those of lower level challenges.

5.3. The difficulty evaluation function for a given challenge

Our aim is to give a measurable notion of the difficulty $D(a)$ for a given challenge a. Let us set up some properties that must fulfil this function:

- 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 challenges 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 challenges already overcome.

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* for a given challenge a and related to tile t as the conditional probability:

$$D(a,t) = Probability\{LOSE(a,t)/A\} \tag{1}$$

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

$$E(a,t) = Probability\{WIN(a,t)/A\} \tag{2}$$

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

$$D^*(a) = \lim_{t\to\infty} D(a,t) \text{ and } E^*(a) = \lim_{t\to\infty} E(a,t) \tag{3}$$

If challenge a must necessarily be finished in the game then we have:

$$D^*(a) + E^*(a) = 1 \tag{4}$$

These functions gives us two kind of information about the challenge difficulty. First, E^* gives us the difficulty of the challenge in terms 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.

The definition of the difficulty of a challenge as a function of time does not mean that the difficulty of a challenge is, in general, simply related to the time spend to overcome the challenge. The difficulty of some challenges is a decreasing function of the time to over come it: disarm a bomb on less than one minute. It can also be an increasing function when, for example, the number of enemies increase with time spend in a room. The time spend to solve an enigma is related to its difficulty. Many other cases can be found in games. In our definition we do not assume the way the time influence the difficulty of a challenge. This relation is needed as we define the Difficulty as a stochastic process. The definition and parameters evaluation of such a process are related to a behavior observed over a given period. Our definition is directly inspired by the formal specification of dependability parameters such the safety or the availability of a system. For example, $S(t)$, the safety of a plane during a fly (probability that a catastrophic event does not occur before the time t) is not, in general, a simple function of the mission time. The asymptotic safety S^*, the probability that the fly end safely, is the equivalent of the asymptotic difficulty D^*.

Moreover, difficulty should be defined as a stochastic process, as it is the way it is considered when building a gameplay. Indeed, shaping a player's experience and thus crafting the right difficulty curve implies a through understanding of the time it takes to win a challenge. Thus, $D^*(a)$ is a good, synthetic estimation of the challenge's a difficulty, but $D(a,t)$ gives some precious additional information to the game designer.

In an open game, there is a small chance that two players will reach the same challenge following the same history. Hence A, the set of challenges being solved by the two players, could be different. So, it is necessary to drive the tester with a walk-through. In this case the difficulty D and the easiness E 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 $(aknowinga)$ this second challenge:

$$D^*(b \text{ knowing } a) \leqslant D^*(b)$$

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

$$D^*(b \text{ knowing } a) \leqslant D^*(b)$$

Defining the difficulty of the game as a probability is also very useful as it helps us getting a precise evaluation of the challenge's difficulty, which only depends on a stable, accepted variable of the gameplay: the challenge's binary result. Indeed, only taking into account the binary result would not be appropriate without a statistical approach: the tuning of a gameplay variable can lead to subtle changes in the difficulty, that a binary measure is not able to represent. But from many observations of this binary result, we can get a fairly precise evaluation of the challenge's difficulty.

This section has described a general definition of difficulty for a given challenge in the context of a given path of challenges. It can be measured experimentally, using many players constrained to a specific follow up of challenges by a walk-through. In order to simplify the formulaes of the next section, we assume that any time we write D we implicitly refer to D^*, and that as a result, we can always take again into account the time in any of the next definitions of difficulty. The next section adopts a complementary approach by exploring the relationship between the basic behaviors the player learns and practices when playing a game, and his probability to lose or win a challenge according the level of his mastering of them.

6. Abilities and difficulty

In this section we propose a method to analyse the link between a challenge's difficulty and the abilities the player has to develop in order to win it. We call *abilities* the basic behaviors

[1] In the sense of challenges that can be played following a given scenario.

the player has to learn and practice in order to increase his efficiency at playing the game. In a racing game for example, the player learns to follow the best trajectory for every road bend type, like sharp bends or double bends. In shooting games, the player learns to be more and more accurate with each weapon, and to keep changing his direction, to prevent the other players from aiming at him. If correctly executed, these abilities will help the player to win the challenge. Our objective here is to link the player's *level* for a given ability to the probability to win or lose the challenge.

We define the relation between the level of a given ability and the result of a challenge completion as a conditional probability, that is, the probability to lose a challenge knowing the player's current ability level. What we propose to measure is the player level for one specific ability while he is playing a given challenge. Having done many of these measures[2] for each level of a particular ability, we are able to calculate the probability the player has to win the challenge, knowing his level for that ability. We thus statistically estimate the relation between the ability level of the player and the challenge completion probability. To keep our proposal simple, we consider a finite and small number of levels.

This conditional probability can reveal many aspects of the gameplay. It can be used to understand which ability is the most important to win a challenge. A strong correlation between the ability level and the probability to win the challenge means that this ability is an important aspect of the gameplay for this challenge. It can also be used as a comparison basis between challenges[3]: an important variation of the required ability level between two consecutive challenges denotes an important change in the gameplay between these challenges. It can also be used as a prediction tool: if we know, from statistical measures, the probability one has to win a specific challenge for a given ability level, we can observe a player, calculate his current level for this ability and get as a result his probability to win the challenge. This kind of prediction can be very useful when dynamically adjusting the difficulty of a game, for instance.

The remainder of this section explains how to calculate the probability the player has to lose, knowing his level for one specific ability.

6.1. Assumptions and notations

In this section, we first set up some hypothesis and notations we will use further. We start by defining the player level for an ability.

The ability level must be calculated for a specific ability, and a specific challenge. Indeed, the player's level for an ability depends on the context and can vary between challenges: for example, the faster the targets will move, the lower the player accuracy will be. To keep the model as simple as possible, the ability level will be a discrete value, for instance[4] one of the following: $\{bad, average, good\}$. We thus assume the following:

- a game is made of challenges, called $c \in C$.
- the player develop abilities, called $a \in A$.
- each ability can be mastered at a certain discrete level $l \in L$. In this article, we consider L as having three levels: $L = \{bad, average, good\}$.
- the function $plevel_c(a) : A \times C \rightarrow L$ is the player level for ability a, when the player is engaged in the challenge c. The method to compute this function is given in the next section.

Given these definitions, we state a link between the challenge c and an level of player mastering for ability a, as the conditional probability $\delta(c, a)$, given by:

$$\delta(c, a) = Probability\{LOSE(c)/plevel(a, c)\} \quad (5)$$

In the next sections, we will statistically evaluate, for each value of $plevel_c(a)$, that is $\{bad, average, good\}$ in this article, the probability the player has to lose the challenge. The first step, explained in the next section, is to calculate the player level for a given ability a and a given challenge c, that we call $plevel_c(a)$.

6.2. Computation of $plevel_c(a)$

We propose to statistically evaluate the player level for an ability from his playing history during a session game. We assume two hypothesis here: the player behavior can be recorded, as a follow up of *game events* that we call a *trace*; the game designer, as an expert, can help us determine the most important events describing the player behavior, in the context of a specific ability we try to measure. We will use this *trace* of game events to estimate the player level for a given ability.

The next following sections describe the traces of events and the calculation of the ability level.

6.2.1. Trace of events

We are measuring a player's ability for some specifics challenge and ability. To do so, we record what happens in the game when the player is trying to solve the challenge, and measure the quality of his behavior along the dimension of our chosen ability. We do not record every change in the game state, but only the changes related to the player ability we want to measure. If we measure accuracy, we only need to record two specific events: when the player shoots and when he hits the target. We will thus record traces of these events, that we define as follow:

- a *game event* $ge_{a,c}$ is an important change in the game state related to the ability a we are measuring during the challenge c.
- a challenge *ends* when it reaches a terminal state, that is when the player has won or lost this challenge (see the automaton in Fig. 2).
- When the player tries to complete a challenge c, he is involved in a *challenge session* $s_{a,c}$, related to an ability a.
- a *challenge session trace* $t_{a,c} \in T_{a,c}$ is the sequence of all game events $ge_{a,c}$ recorded during a challenge session $s_{a,c}$.
- the function $result(t_{a,c}) : T_{a,c} \rightarrow \{win, lose\}$ determines if, at the end of a given challenge session, the player has won or lost the challenge.

6.2.2. Abilities and subgoals

All our measures rely on one central hypothesis: we suppose that the main goal of the player is to win the challenge he is involved in. To overcome it, the player repeatedly tries to reach short term goals, or *sub-goals*, that allow him to move closer from his main goal. When playing a game like God of War (Sony – Santa Monica Studio), the player is slashing group of enemies, one after the other. Each groups can be considered as a challenge. But when playing, his actual goal is to execute a combo at the right moment, to kill as many enemies as possible. He will try to reach this *sub-goal* as often as possible, and eventually will kill all the enemies and progress to the next challenge.

[2] For many players.

[3] We compare them on the basis of the same ability.

[4] The number of levels is not a fixed value, and can be modified to match the needed level of description. But the more precise the model is, the more data it must be fed with to be statistically valid.

We suppose also, that mastering an ability eventually leads the player to reach a certain sub-goal[5] when playing the game. During one challenge session, many games involve the repetition of the basic behaviors we call abilities. Thus, the player will try, again and again, to attain the sub-goal related to the ability we measure, with a certain probability of success. During a challenge session, we will count how many times the player succeed of fail to attain this sub-goal, and thus get a statistical measure of the ability level from his many attempts to reach it.

The attainment of this sub-goal must only depend on the player level for this ability. For example, in Mario series (Nintendo), we cannot measure the player ability to kill mushrooms from the goal of staying alive: it is a too broad goal, also influenced by the player's capacity to jump from a platform to another one, avoid turtles shells, and many others. But the sub goal of having a dead mushroom is perfect for this ability.

Given a challenge session trace, we assume being able to identify when the player has reached the sub-goal related to the ability, and when he has failed reaching it. If we do so, we can evaluate at which frequency the player has reached the sub-goal, and use this score as the player level for this ability. Having a trace $t(a,c) \in T(a,c)$ of all game events happening during a session for a challenge c and ability a, we need to fulfill the following conditions in order to calculate the player level:

- each ability $a \in A$ that the player can use in challenge c must correspond to a short term goal $g_{a,c}$ of the player.
- there exists a bijective function $subgoal_c(a)$, statically defined with the assistance of the game designer, giving the subgoal $g_{a,c}$ related to an ability a during a challenge c.
- we must be able to count, in challenge session trace $t_{a,c}$, how many times the player has reached $g_{a,c}$, which we define as the function $nsuccess_c(t,a)$
- we must be able to count, in challenge session trace $t_{a,c}$, how many times the player has failed to reach $g_{a,c}$[6], which we define as the function $nfailure_c(t,a)$

If we know how many times the player succeeded or failed to reach the sub-goal related to an ability, we know the probability the player has to reach the sub goal:

$$ProbaSubgoal_c(t,a) = \frac{nsuccess_c(t,a)}{nsuccess_c(t,a) + nfailure_c(t,a)} \quad (6)$$

The result probability is thus a score for the player level on a given ability, during a challenge session trace. We will not keep this value but to lower down the complexity of the player model, map it into the discrete values, of $L = \{bad, average, good\}$. The mapping could, for example, be defined as follow:

$$plevel_c(t,a) = \begin{cases} bad & \text{iff } 0 \leqslant ProbaSubgoal_c(t,a) < 0.3 \\ average & \text{iff } 0.3 \leqslant ProbaSubgoal_c(t,a) < 0.6 \\ good & \text{iff } 0.6 \leqslant ProbaSubgoal_c(t,a) \leqslant 1 \end{cases} \quad (7)$$

The following algorithm shows the calculus of the function $plevel_c(t,a)$. The function do have a challenge as a parameter, as we only give the traces we recorded for a specific challenge and ability.

```
1function plevel(Ability a, Trace t)
2            return Level
3{
4     var g=subgoal(a,c); //Get the subgoal related to a and c
5     var nbsuccess=0; //To count the successes
6     var nbtotal=0; //it Total counter
7
8    foreach subtrace ti of t where players tries
to reach g
9    {
10       if(success(ti, g)) //If goal reached
11          nbsuccess++;
12          nbtotal++;
13  }
14
15  //Get the probability to reach the sub-goal
16  var p=nbsucess / nbtotal;
17
18  //Map it to a discrete value
19  var level="bad";
20  if(p>0.3)
21     level="average";
22  if(p>0.6)
23        level="good";
24
25  return level;
26}
```

Many different means can be used to know if a player tries to reach a subgoal (line 8 of the previous algorithm). We will explore different options, the first one being to manually define a state machine that recognizes when a follow up of events means that the player has succeeded or failed to reach the goal.

Now that we can calculate, for a given ability and a challenge record session, the player level, we are ready to calculate δ, and will do so in the next section. We will record many challenge execution trace for a given challenge, group them according to the player ability level we observe in the trace, and thus calculate the probability the player has to lose in each case of the conditional probability δ, for each player ability level. The next section explains this part of the calculus.

6.3. Calculating δ

We are going to statistically evaluate δ, using the set of traces $T_{a,c}$we have recorded. We partition the set $T(a,c)$ of all the challenge session traces we recorded for challenge c and ability a, into $card(L)$ subsets: $T^{bad}_{a,c}, T^{average}_{a,c}, T^{good}_{a,c}$, using $plevel_c(t,a)$ s:

- $T^{bad}_{a,c}$ will contain all the traces $t_{a,c} \in T_{a,c}$ where $plevel_c(t,a) = bad$
- $T^{average}_{a,c}$ will contain all the traces $t_{a,c} \in T_{a,c}$ where $plevel_c(t,a) = average$
- $T^{good}_{a,c}$ will contain all the traces $t_{a,c} \in T_{a,c}$ where $plevel_c(t,a) = good$.

Our partition groups traces into subsets by ability level. Each of these levels is a different value of the conditional part of the probability δ. The next step is to estimate the probability to lose, for each ability level.

As we said in the previous section, we can also know, for any challenge session trace, if the player has won or lost the challenge, that is, the challenge sessions result $result_c(t,a)$. We can thus

[5] That we assume such that there is a unique sub-goal for each ability, and conversely.

[6] That hypothesis amounts to being able to recognize patterns of events corresponding to the attainment of given sub-goal.

count, in any subset T' of $T_{a,c}$, the number of traces where the player has lost, which we will call $nlost(T')$.

Given the number of lost challenge sessions in each partition subset $nlost(T')$, we can now get the player's probability to lose for each ability level, that is, for each subset of $T_{a,c}$:

$$ProbaLost(T') = \frac{nlost(T')}{card(T')} \quad (8)$$

We can then build the whole conditional probability table for the function δ for an ability a and a challenge c. We calculate from our trace in T^c, the value $ProbaLost(T')$, for every $T', T^{bad}_{a,c}, T^{average}_{a,c}, T^{good}_{a,c}$, These values correspond to the probability to lose for each player level of $plevel_c(t,a)$ and we thus know all the conditional probabilities of the function δ:

$$\delta(c,a) = Probability\{LOSE(c)/plevel_c(a)\} \quad (9)$$

The next algorithm shows the computation of δ using $plevel_c(t,a)$ defined in the previous section. It calculates the function δ for a given player level and an array of challenge session traces, recorded for a specific challenge. Thus, the function need not know for which challenge the calculus is done, as it only gets the session traces of a specific challenge.

```
1 function delta(Ability a, Level l, Trace T[])
2           return float
3 {
4   var nbsuccess=0; //To count the successes
5   var nbtotal=0; //Total counter
6
7   foreach trace t of T
8     {
9       //Take only the traces corresponding to the level we want
10      if(plevel(a,t)==l)
11      {
12        if(result(t)==“win”)
13          nbsuccess++;
14      nbtotal++;
15    }
16  }
17
18  return nbsuccess/nbtotal;
19}
```

6.4. From local to global abilities

In the previous section, we proposed a way to measure the player abilities to win a challenge. These abilities can be called local, because we are only defining and measuring them in the context of a specific challenge. However, we explained in Section 5.2 that what the player learns in a challenge is often useful in the next one, and we defined a relation between atomic and higher level challenges. As a result, many abilities must be shared between challenges: if the player learns to do a specific task to win a challenge, a higher level challenge of the same type has many chances to rely on the same kind of ability, but with different difficulty settings.

Thus, it would be interesting for the game designer to define global abilities, shared between a set of challenges, and thus compare the difficulty of these challenges for a given level of this global ability. If the player must be good at aiming for any challenge of the game, it would be much more interesting to know the probability he has to lose each challenge given his general ability to aim, because we may then compare the challenges between them.

The problem we are facing, when defining global abilities over multiple challenges, is that we will not find the same ability level (Eq. 7) for every challenge. A player may aim poorly and rated as *bad* for the aiming ability on a hard challenge because targets are moving really fast and be rated as *good* on a much easier challenge. The probability that Eq. 7 gives us will not be the same depending on the challenge, because the context of the challenge will influence it.

To transform the local ability given by Eq. 7 into a global one, we propose to observe the local ability ratings the same player gets when playing two different challenges successively. If we make the rough approximation that the player level as not raised a lot between these consecutive challenges and thus consider it constant, the two ratings we get are the measure of the same ability value within two different challenges contexts. Given many examples of consecutive local ability ratings, we may find a function mapping these local abilities to a global one. As a result, we may then define the difficulty of these challenges over the same global ability. These proposition must be further explored through experimentation though.

6.5. Understanding the player's abilities that really counts

The automated system we proposed is primarily designed to get a clear, synthetic view of a challenge difficulty, that is, the probability the player has to loose the challenge. But it can also be very useful to understand how this difficulty is created, that is, the core components that make the game hard or easy.

Indeed, the designer has an idea of the abilities the player has to master in order to rule the game. In a shooting game, the player has to move fast, remember the map layout, aim precisely, use certain strategies. But if the designer can, from his through understanding of the gameplay he created, identify the player abilities, he may not exactly know which one of them is the most important, which one of them really makes an actual player win or lose the game.

That why aside from calculating each challenge ability, we designed our software to output a graphical view of the Eq. 5, as well as useful statistical values, such as a linear regression. That way, the designer can visualize the link between any ability he can evaluate as a short term goal and the difficulty of the game, and get a better understanding of the abilities that really count, regarding a specific challenge.

7. Conclusion

There is a lack of a precise definition of difficulty in games, and more important, game designer lack of methodology and tools to measure it. In this paper, we perform a first step to fill this gap. Our approach is twofold: first, we propose a precise and measurable definition of *difficulty for challenges* in a game on behold of the past experience of the player, and second, we provide a method to explore one specific aspect of difficulty in challenges: the relation between what the player learns and the probability he has of overcoming that particular challenge.

In a first experiment using a synthetic player and a basic AI driven player, we are able to extract objective difficulty measures out of a simple game. As a generalization, we propose a measurable definition of difficulty based on the past experience of the player. This notion uses the set of challenge the player has tried to solve, and define the difficulty of a challenge as the probability to lose over time. The obtained function is measurable under two hypothesis: the game design provides a clear specification of challenges; we are able to constrain the players to a specific follow up of challenges while recording the results of their playing ses-

sions. This first measure provides a general vision of the game difficulty.

As a complementary measure, we explore more thoroughly the player learning when playing a game. More precisely, we propose to measure the quality in the execution by the player of some basic behaviours he learns and practices while playing, and which has been previously identified by the game designer as important to overcome a given challenge. We call *abilities* these basic behaviours, that we assume to be measurable during a session game. We propose then to calculate the relation between the level of mastering of a particular ability and the probability the player has to lose a given challenge.

These measures can provide insights to game designer when trying to assess different aspects of a gameplay. They can help them to determine the importance of an ability for a specific challenge, or to compare challenges on the basis of one specific ability. The next step of our research is to implement both measures in different types of games, and to experiment them using real players.

References

[1] D. Boutros, Difficulty is difficult: designing for hard modes in games, Gamasutra, 2008. Available from: <http://www.gamasutra.com/> (last access 01/2009).

[2] E. Adams, The designer's notebook: difficulty modes and dynamic difficulty adjustment, Gamasutra, 2008. Available from: <http://www.gamasutra.com/> (last access 01/2009).

[3] J. Juul, The game, the player, the world: looking for a heart of gameness, in: M. Copier, J. Raessens (Eds.), Level Up: Digital Games Research Conference Proceedings, 2003, pp. 30–45.

[4] R. Hunicke, The case for dynamic difficulty adjustment in games, in: Advances in Computer Entertainment Technology, 2005, pp. 429–433.

[5] U. Eco, L'oeuvre ouverte, Seuil, 1965.

[6] R. Koster, A Theory of Fun for Game Design, Paraglyph Press, Scottsdale, Arizona, 2005.

[7] P. Sweetser, P. Wyeth, Gameflow: a model for evaluating player enjoyment in games, Comput. Entertain. 3 (3) (2005) 3. doi:http://dx.doi.org/10.1145/1077246.1077253. URL http://dx.doi.org/10.1145/1077246.1077253.

[8] G.N. Yannakakis, J. Hallam, Towards optimizing entertainment in computer games, Applied Artificial Intelligence 21 (10) (2007) 933–971.

[9] M. Csikszentmihalyi, Flow: The Psychology of Optimal Experience, Harper Perennial, 1991. Available from: <http://www.amazon.ca/exec/obidos/redirect?tag=citeulike04-20&path=ASIN/0060920432>.

[10] W. Jsselsteijn, Y. De Kort, K. Poels, A. Jurgelionis, F. Belotti, Characterising and measuring user experiences in digital games, in: International Conference on Advances in Computer Entertainment Technology, 2007.

[11] G. Andrade, G. Ramalho, H. Santana, V. Corruble, Extending reinforcement learning to provide dynamic game balancing, in: IJCAI 2005 Workshop on Reasoning, Representation, and Learning in Computer Games, 2005, pp. 7–12.

[12] B. Scott, Architecting a game AI, in: AI Game Programming Wisdom 1, Charles River Media, Inc., 2002.

[13] A. Macleod, Game design through self-play experiments, in: ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, ACM, New York, NY, USA, 2005, pp. 421–428. doi:http://doi.acm.org/10.1145/1178477.1178572.

[14] N. Kirby, AI as gameplay analysis tool, in: Game Programming Wisdom 4, Course Technology, Cengage Learning, 2008, pp. 39–49 (Chapter 1).

[15] T. Bullen, M.J. Katchabaw, N. Dyer-Witheford, Automating content analysis of video games, in: Canadian Game Studies Association (CGSA) Symposium, 2006.

[16] G. van Lankveld, P. Spronck, M. Rauterberg, Difficulty scaling through incongruity, in: Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, Stanford, California, USA, 2008.

[17] A. Nantes, R. Brown, F. Maire, A framework for the semi-automatic testing of video games, in: Artificial Intelligence and Interactive Digital Entertainment Conference, AAAI, 2008.

[18] B. Chan, J. Denzinger, D. Gates, K. Loose, J. Buchanan, Evolutionary behavior testing of commercial computer games, Congress on Evolutionary Computation, CEC2004 1 (2004) 125–132, doi:10.1109/CEC.2004.1330847.

[19] P. Spronck, Evolving improved opponent intelligence, in: GAME-ON Third International Conference on Intelligent Games and Simulation, 2002, pp. 94–98.

[20] J.E. Laird, Game developers magazine.

[21] J. Pittman, The pac-man dossier, Gamasutra, 2009. Available from: <http://www.gamasutra.com/> (last access 01/2009).

[22] G.N. Yannakakis, J. Hallam, Evolving opponents for interesting interactive computer games, in: Animals to Animats 8: Proceedings of the Eighth International Conference on Simulation of Adaptive Behavior (SAB-04), The MIT Press, Santa Monica, CA, USA, 2004, pp. 499–508.

[23] S.M. Lucas, Evolving a neural network location evaluator to play ms. pac-man, in: Proceedings of the 2005 IEEE Symposium on Computational Intelligence and Games (CIG05), 2005.

[24] M. Gallagher, M. Ledwich, Evolving pac-man players: can we learn from raw input? in: Computational Intelligence and Games (CIG), 2007.

[25] J. Bonet, C. Stauffer, Learning to play pac-man using incremental reinforcement learning, 2001. Available from: <http://www.ai.mit.edu/people/stauffer/Projects/PacMan> (accessed 5.12.2008).

[26] J.P. Rosca, Generality versus size in genetic programming, in: Genetic Programming 1996: Proceedings of the First Annual Conference, MIT Press, 1996, pp. 381–387.

[27] M. Gallagher, A. Ryan, Learning to play pac-man: an evolutionary, rule-based approach, in: Evolutionary Computation (CEC '03), vol. 4, 2003, pp. 2462–2469.

[28] Watkins, C.J. Learning from delayed rewards, Ph.D. Thesis, Cambridge, 1989.

[29] S. Natkin, L. Vega, A petri net model for computer games analysis, International Journal of Intelligent Games & Simulation 3 (1) (2004) 37–44.

[30] S. Natkin, A.-M. Delocque-Fourcaud, E. Novak, Video Games and Interactive Media: A Glimpse at New Digital Entertainment, AK Peters Ltd., 2006.

[31] E. Byrne, Game Level Design (Game Development Series), Charles River Media, 2004. Available from: <http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/1584503696>.

[32] J. Juul, Half-Real: Video Games between Real Rules and Fictional Worlds, The MIT Press, 2005. Available from: <http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0262101106>.