From Competitive to Social Two-Player Videogames

Jesús Ibáñez-Martínez
Universitat Pompeu Fabra
Barcelona, Spain
jesus.ibanez@upf.edu

Carlos Delgado-Mata
Universidad Panamericana
Aguascalientes, Mexico
cdelgado@up.edu.mx

ABSTRACT
In this paper we present a strategy to design social videogames (from classic competitive ones) which allow parents to play with their children and have fun in spite of their different levels. A first tennis videogame (based on the classic Pong) which implements this strategy has already been developed. In the paper we first motivate this work and briefly survey related work. Then we describe the general strategy and its application to the case of the tennis videogame.

Categories and Subject Descriptors
I.6.8 [**Simulation and modeling**]: Types of Simulation – *gaming*.
K.8.0 [**Personal computing**]: General – *games*.

General Terms
Algorithms, Design, Human Factors

Keywords
Two-player Videogames, Automatic Adaptation

1. MOTIVATION
When an adult plays with a little child in the physical world, the adult usually adapts his abilities to the particular level of the child, especially when the child is learning to play. For instance, a father can be proficient playing tennis, but when playing with his little son who is starting to play tennis, he does not try to compete with his son. On the contrary, he hits the ball softer than he does when playing with his friends, and he tries to hit the ball so that it gets near his child so it is easier to hit back. In other words, he presents his child with situations which are feasible for him to achieve so that he gets engaged instead of frustrated. He tries to facilitate the learning process by motivating him.

Playing tennis in this way is not competitive for the father. As a tennis game (as a sport), it is not exciting for him. However, it can be emotionally gratifying. The father can enjoy playing and chatting with his son. However, from the adult perspective, this kind of scenario is funny for a short time. If the parent plays tennis for a long time with his child he will get bored.

ICMI-MLMI'09 Workshop on Child, Computer and Interaction
November 5, 2009, Cambridge, MA, USA.

When playing two-player videogames instead of real world games the situation is pretty similar. Two-player videogames are competitive videogames where one player competes against other one. In this kind of games, the parameters that depend on the degree of difficulty of the game (like the speed of the ball in a tennis videogame) are usually the same for both players (even if these parameters can be configured). Thus, the game is as difficult (or as easy) for both players. Moreover, usually this kind of game includes an option for a player to play against the computer (instead of against another human player) and some games allow the player to select the degree of difficulty in this case.

If we translate the father-child situation we mention above to the digital context, we would have a little kid who is learning to play a two-player videogame (more concretely a tennis videogame) and his father who is proficient playing digital tennis wanting to play with his kid. In this case there currently are three possible scenarios. In the first scenario the child plays against the computer in order to learn to play so he can play against his father once he acquires certain abilities. The problem here is that the game lacks social interaction, chat, empathy, etc. In the second scenario the father plays against his child setting a low degree of difficulty so that the child does not get unmotivated nor frustrated. The main problem here is that the father can get bored as the game will be too easy for him. In the third scenario the father plays against his child setting a higher degree of difficulty so that he does not get bored (he thinks). The problem here is that the child will get unmotivated and frustrated as he won't be able to do anything as the game is too difficult for him. Moreover, the father will get bored as he wins without any opposition.

In this paper we propose a novel strategy which provides a fourth possible scenario which fits the abilities of both players better. It is a strategy to design two-player videogames which are able to adapt themselves to the level of the players who are playing at each moment. If a player is playing too badly (the game is too difficult for him), the game gets easier for him so he does not get frustrated. If a player is playing too well (the game is too easy for him), the game gets more difficult for him so he does not get bored. Thus, the players can benefit from social interaction, chat, empathy, etc. while playing, in spite of their levels.

The strategy has already been successfully applied to the design and development of a new version of the classic Pong videogame. Pong was a pretty simple game with simple rules: hit the ball across the playing field and try your best to hit it past your opponents paddle on the other side (see Figure 1).

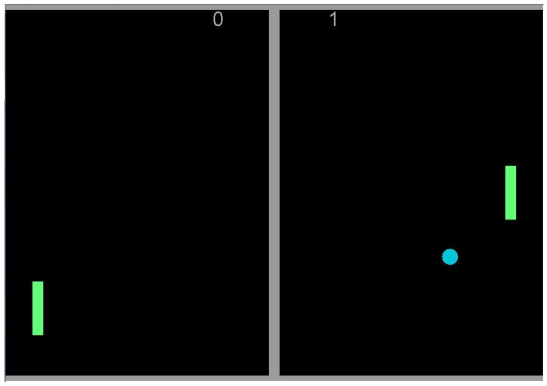

Figure 1. The Pong videogame

The structure of the paper is as follows. First we review related work. Next we describe the general strategy and its application to the case of the tennis videogame. We also show the development of the tennis videogame. Finally we show the conclusions and point out future work.

2. RELATED WORK

The concept of flow [3] described by Mihaly Csikszentmihalyi is very relevant to our objective. Flow is the mental state of operation in which the person is fully immersed in what he or she is doing by a feeling of energized focus, full involvement, and success in the process of the activity. Flow is a mental state of enjoyment shared by people in a variety of situations, such as rock-climbing, chess-playing, composing music, and playing videogames. Colloquial terms for this or similar mental states include: to be on the ball, in the zone or in the groove. The concept of flow, in [4], refers to an individual's optimal experience. In a state of flow, the individual experiences intrinsic enjoyment from undertaking a task that feels almost effortless and natural, while also causing the individual to feel focused and challenged. One of Csikszentmihalyi's inspiring achievements is the definition of the Flow Channel (see figure 2).

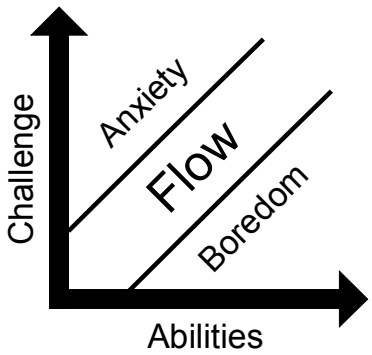

Figure 2. The Flow Channel

In order to maintain a person's Flow experience, the activity needs to reach a balance between the challenges of the activity and the abilities of the participant. If the challenge is higher than the ability, the activity becomes overwhelming and generates anxiety or frustration. If the challenge is lower than the ability, it provokes boredom. Thus, the relation between game difficulty and player ability is crucial for enjoying the game.

Dynamic difficulty adjustment (DDA), also known as Dynamic game difficulty balancing or Dynamic game balancing is the process of automatically changing parameters, scenarios and behaviours in a videogame in real-time, based on the player's ability.

There are few works dealing with DDA. Literature includes works which explore the use of concrete techniques (inventory theory [10][11], dynamic scripting [13], genetic algorithms [8], reinforcement learning [1], etc.) to control concrete elements (inventory [10][11], agents behaviours [1][8][13], etc.) of complex videogames. Unfortunately, implementing such systems in complex videogames poses many challenges to game developers. As a result, DDA techniques are not widespread.

Thus, while most works deal with concrete mechanisms for concrete elements of complex videogames, our work is focused on a general strategy for adding DDA to a simple kind of game where two players play against each other (a sort of game not studied in literature). We define a general strategy for this kind of games and we apply the strategy to a concrete videogame.

Finally, to put this work in perspective, it should be noted that the videogame we have designed and developed incorporating our approach is based on the classic Pong. Pong, while not the first videogame, was the first coin-op arcade game and the first mainstream videogame that was available to almost everyone. Pong was invented by Ralph H. Baer in late 1960s [2] and it was later licensed to Magnavox, which successfully marketed it. An arcade version of the game was developed by Atari, the company founded by Nolan Bushnel, in the 1970's. Pong was a pretty simple game with simple rules: hit the ball across the playing field and try your best to hit it past your opponents paddle on the other side. The origins of Pong lie with an abstract tennis game created with an old oscilloscope and some vacuum tubes by Willy Higinbotham, way back in 1958 [7].

3. STRATEGY

The ultimate objective of the strategy is that both players enjoy the game, independently of their particular level. The game should be neither very easy nor very difficult for any player, in order to avoid that they get bored or frustrated. This general objective should be more concretely defined according to some criteria which depend on the concrete game. In the particular case of the tennis videogame, we defined the following concrete objectives:

- the number of points per player should be balanced. That is, the performance of both players should be as similar as possible so they keep engaged.
- the number of hits per point should not be very small nor very great. If there were very few hits per point the game would be frustrating. If there were too many hits per point the game would be boring.

In order to achieve these objectives, a set of parameters and an algorithm (which varies these parameters to balance the level of difficulty for each player) are defined.

3.1 Parameters

A set of parameters should be defined in order to control the difficulty of the game for each player. These parameters must allow increasing or decreasing the difficulty of the game for each particular player without affecting the performance of the other player.

In the case of the tennis videogame, each player level, L, is modelled as a real value in (-1, 1). The initial value of each player is 0. Positive values of L indicate that the player is playing well and negative values of L indicate that the player is playing badly. The level of the player determines the value of three parameters (which are modelled as real numbers):

- RL, the player's racquet length, MinRL ≤ RL ≤ MaxRL
- RS, the player's racquet speed, MinRS ≤ RS ≤ MaxRS
- BS, the ball speed when the player's opponent hits the ball, MinBS ≤ BS ≤ MaxBS

Every period of time, the level of each player (L) is recalculated by the algorithm described in the next section. The value of L determines then the value of the three parameters by following simple proportional rules as shown in Table 1. MinRL, MinRS and MinBS are the minimum values of RL, RS and BS respectively. These are also the initial values for theses parameters. MaxRL, MaxRS and MaxBS are the maximum values of RL, RS and BS respectively. Note that L and BS are directly proportional while both L and RL and L and RS are inversely proportional.

Table 1. Parameters

L	RL	RS	BS
1			MaxBS
0	MinRL	MinRS	MinBS
-1	MaxRL	MaxRS	

These parameters allow increasing or decreasing the difficulty of the game for one player. On the one hand, the first two parameters (RL and RS) are used in order to make the game easier for a player who is playing badly. The smaller the player level, the greater both the racquet length and the racquet speed are. On the other hand, the last parameter (BS) is employed in order to make the game more difficult for a player who is playing very well. The greater the player level, the greater the ball speed when the opponent hits the ball.

3.2 Algorithm

Every period of time the performance of both players is evaluated and compared. If there is a great difference of level between both players (one of them is playing much better than the other), the algorithm balances the difficulty degree of the players by varying the parameter values. Moreover, even if there is no great difference of level between both players, the algorithm still tries to evolve the game in order to improve the engagement of the players.

In the concrete case of the tennis videogame, we consider that a player is playing much better than the other one in a given time period if the first player has scored many more points than the second one during that period.

There are two possible reasons for a great difference between players. On the one hand, maybe one of the players is playing very badly. If this is the case, the game should be simplified for this player by setting appropriate parameter values. On the other hand, maybe one of the players is playing very well. In that case, the game should be made more difficult for this player.

In the case of the tennis videogame, we distinguish between these two cases by taking into account the mean number of hits per point in the time period. If there are very few hits per point we consider this a symptom that at least one of the players is playing badly. As we have already detected that there is a great difference between both players, then we can assume that the player with the worst performance is playing badly. On the other hand, if there are enough hits per point we consider this a symptom showing that both players are playing at least relatively well. As we have already detected that there is a great difference between both players, then we can assume that the player with the best performance is playing very well.

As we said above, even if there is no great difference of level between both players, the algorithm still tries to evolve in order to improve the engagement of the players. If both players are playing very well (the game is too easy for them), the game should be made more difficult for both players so they do not get bored. If both players are playing badly (the game is too difficult for them), the game should be simplified for both of them. Again, in the case of the tennis videogame we detect that the players are playing badly if there are too few hits per point. They are playing very well if there are too many hits per point.

Thus, the general algorithm for the case of the tennis videogame is as follows:

```
00 every period of time:
01       if muchBetter(player1, player2)
02                if fewHitsPerPoint()
03                         decreaseLevel(player2)
04                else if manyHitsPerPoint()
05                         increaseLevel(player1)
06       else if muchBetter(player2, player1)
07                if fewHitsPerPoint()
08                         decreaseLevel(player1)
09                else if manyHitsPerPoint()
10                         increaseLevel(player1)
11       else if fewHitsPerPoint()
12                decreaseLevel(player1)
13                decreaseLevel(player2)
14       else if manyHitsPerPoint()
15                increaseLevel(player1)
16                increaseLevel(player2)
17       updateParameters()
```

Where **muchBetter(player1, player2)** is a Boolean function which returns true if player1 has scored many more points than player2 during the last period of time. It returns false in any other case. **fewHitsPerPoint()** is a Boolean function which returns true if the mean number of hits per point in the period is smaller or equal than a given threshold value. It returns false in any other case. **manyHitsPerPoint()** is a Boolean function which returns true if the mean number of hits per point in the period is greater than a given threshold value. It returns false in any other case. **decreaseLevel (P)** is a function which decreases the value of the variable L (level) of the player P according to a particular function, if it is possible (if the value of L is not the minimum value). **increaseLevel(P)** is a function which increases the value of the variable L of the player P according to a particular function, if it is possible (if the value of L is not the maximum value). **updateParameters()** is a function which updates the parameters RL, RS, and BS of both players accordingly to current players' level (L) as shown above in section "Parameters".

As a consequence of this updating of parameters, if a player has been playing too badly during the last period (the game is very difficult for him), his racquet will grow and it will also be quicker so that it is easier for him to reach the ball. If a player has been playing too well (the game is very easy for him), the ball will move quicker when hit by his opponent so that the game is more challenging and funny for him.

4. DEVELOPMENT

We designed and developed the tennis videogame implementing the described strategy (see figure 3). We employed design patterns for the software design, as we wanted the videogame to be easily modifiable and extended. In particular, we used the *strategy* design pattern where possible so that programmers can easily supply variants of the different algorithms. This allows for exploring new mechanisms to change the level of the game. Furthermore, the current design of the videogame facilitates the creation of new videogames implementing the general strategy. The programmer still has to program the concrete algorithms for making the new game easier or more difficult, but he can reuse the general structure.

Figure 3. Two players enjoying the developed videogame

The game was developed in Java [9] (Java has recently been claimed as a good language for videogame development [5]), employing Java2D [12] for graphics. Thus, the game is portable. It can be played on any PC. The game is also highly configurable. All the game parameters, including the parameters we have mentioned above (MinRL, MaxRL, etc.), can be configured through a configuration file.

The current version of the videogame runs as a Java application and it is visualized on a window. We are creating two new versions of the game at the moment. The first one runs on a full-screen display, while the second one runs on an applet so the users can play the game on a web browser.

The game supports two different kinds of input device. On the one hand, the players can use the PC keyboard. On the other hand, they can utilize a gamepad (in particular we use the Logitech Dual Action gamepad). We have employed the JInput library [6] for programming the gamepad support.

5. CONCLUSIONS AND FUTURE WORK

We have introduced a novel strategy to design videogames which allow parents to play with their children and have fun in spite of their different levels. A first tennis videogame which implements this strategy has already been developed and it has been described in the paper. Even though we have not yet carried out formal user studies, the players who have played the game so far have found it funny.

As future work, we are preparing a study with users in order to evaluate the player experience when playing the designed adaptive tennis videogame in comparison to when playing the game without the adaptation mechanisms. In addition, we will explore the application of the defined strategy to design other adaptive videogames.

6. ACKNOWLEDGMENTS

We would frankly like to thank Leticia Lipp for generously proofreading. This work has been partially funded by the Spanish Ministry of Science and Innovation in the Learn 3 project (TIN2008-05163/TSI).

7. REFERENCES

[1] Andrade, G., Ramalho, G., Santana, H., and Corruble, V. 2005. Challenge-Sensitive Action Selection: an Application to Game Balancing. In Proceedings of the IEEE/WIC/ACM international Conference on intelligent Agent Technology (September 19 - 22, 2005). IAT. IEEE Computer Society, Washington, DC, 194-200. DOI= http://dx.doi.org/10.1109/IAT.2005.52

[2] Baer, R. H. 1972. Television Gaming Apparatus and Method. U.S. Pattent 3,659,285. Filed August 21, 1969.

[3] Csikszentmhalyi, M.1990. Flow: The Psychology of Optimal Experience. HarperCollins Publishers.

[4] Csikszentmihalyi, M. 1996. Creativity: Flow and the Psychology of Discovery and Invention. HarperCollins Publishers.

[5] Davison, A. 2005. Killer Game Programming in Java. O'Reilly Media, Inc.

[6] Davison, A. 2007. Pro Java 6 3D Game Development: Java 3D, JOGL, Jinput, & JOAL APIs: Java 3D, JOGL, Jinput, and JOAL APIs. Apress Academic.

[7] DeMaria, R. and Wilson, J. L. 2002. High Score! The Illustrated History of Electronic Games. Osborne/McGraw-Hill.

[8] Demasi, P., and Cruz, A. 2002. Online Coevolution for Action Games. In Proceedings of The 3rd International Conference on Intelligent Games And Simulation (London, 2002), 113-120.

[9] Flanagan, D. 2005. Java In A Nutshell. O'Reilly Media, Inc.

[10] Hunicke, R. 2005. The case for dynamic difficulty adjustment in games. In Proceedings of the 2005 ACM SIGCHI international Conference on Advances in Computer Entertainment Technology (Valencia, Spain, June 15 - 17, 2005). ACE '05, vol. 265. ACM, New York, NY, 429-433. DOI= http://doi.acm.org/10.1145/1178477.1178573

[11] Hunicke, R., Chapman, V. 2004. AI for Dynamic Difficulty Adjustment in Games. In Proceedings of the Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence (AAAI '04) (San Jose, California) AAAI Press.

[12] Knudsen, J. 1999. Java 2D Graphics. O'Reilly Media, Inc.

[13] Spronck, P., Sprinkhuizen-Kuyper, I., and Postma, E. 2004. Difficulty Scaling of Game AI. In Proceedings of GAME-ON 2004: 5th International Conference on Intelligent Games and Simulation (eds. El Rhalibi, A., and D. Van Welden) (Belgium, 2004), 33-37.