Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2012, Article ID 929814, 18 pages
doi:10.1155/2012/929814

Research Article

A Framework for Adaptive Game Presenters with Emotions and Social Comments

Effie Karouzaki[1] and Anthony Savidis[1, 2]

[1] *Human-Computer Interaction Laboratory, Institute of Computer Science, Foundation for Research and Technology Hellas, N. Plastira 100, Vassilika Vouton, 70013 Heraklion, Crete, Greece*
[2] *Department of Computer Science, University of Crete, Knossou Avenue, 71409 Heraklion, Crete, Greece*

Correspondence should be addressed to Effie Karouzaki, karuzaki@ics.forth.gr

Received 22 March 2012; Revised 25 July 2012; Accepted 25 July 2012

Academic Editor: Alexander Pasko

More and more games today try to adjust their gameplay to fit individual players; however, little work has been carried out in the same direction towards game presenter characters. Game commentary should take into account players' personalities along with game progress in order to achieve social player-adapted comment delivery that boosts the overall gameplay, engages the players, and stimulates the audience. In our work, we discuss a framework for implementing artificial game presenter characters that are based on game actions and players' social profiles in order to deliver knowledgeable, socially oriented comments. Moreover, the presented framework supports emotional facial expressions for the presenters, allowing them to convey their emotions and thus be more expressive than the majority of the commentary systems today. We prove our concept by developing a presenter character for multiplayer tabletop board games which we further put under usability evaluation with 9 players. The results showed that game sessions with presenter characters are preferred over the plain version of the game and that the majority of the players enjoy personalized social-oriented comments expressed via multimedia and emotions.

1. Introduction

Our work on game presenter characters has been motivated by the popularity of television game shows and the lack of an analogy in the domain of computer-based entertainment. A significant amount of games played on TV shows are computer-based tabletop ones (Wheel of Fortune, Power of Ten, Who Wants to Be a Millionaire, etc.) with an overall setup emphasizing and amplifying social interaction. Game show presenters are identified as one of the 7 key attributes of appreciation of TV game shows [1]. From a social perspective, they are responsible for keeping the game socially engaging and stimulating. More specifically, presenters provoke social interaction in order to keep the players and the audience constantly motivated and alerted about the game progress. For this purpose, they rely on individual player profiles, current challenge, previous performance, and statistics to provide feedback commonly involving humor, reward, sympathy, surprise, disappointment, enthusiasm, agony, and anticipation.

Nowadays, more and more games incorporate artificial commentator systems (Pro Evolution Soccer Series [2], Buzz! [3], etc.). Such systems are based on game events and use prerecorded voices of actors, game commentators, and TV-show presenters in order to feel more realistic to players. Some of these games ask for player names, in order to use them when addressing to each player. This is a very simple yet extremely clever technique to personalize the presenter's comments and make players get more into their game role, thus achieving a better gaming experience. RoboCup commentary systems [4, 5] receive game events as input and output text and voice that comment on these events. The *Byrne* commentary system [6] also utilizes a visual representation of the presenter's face which incorporates emotion expressions. All such game commentators exhibit the same, predefined behavior for all players, taking into

account only their game progress. They will make the same comments regardless if their players are young or old, businessmen or unemployed, or single or married. In other words, they do not use adaptive social feedback to make game comments. However, a significant amount of work has been done on adaptive social feedback in the field of educational games. As teaching and learning are both social activities, game agents need to be informed about players' social characteristics and use them to control the gameplay and come up with stimulating and engaging comments that will motivate the students to play and learn. The fact that agents with adaptive social feedback can be used to boost the overall gaming experience and improve the learning process is widely accepted, and a lot of scientific work has been done towards this direction [7–11].

Although playing multiplayer games is also considered a highly social activity (such as learning and teaching), little work exists that combines the adaptive social feedback with game presenter characters in this field. We believe that social feedback, used in educational games in order to boost learning, can also be used by game presenter characters in order to better motivate the players and stimulate the audience. Towards this direction, we present a framework for building artificial game presenter characters with emotions, capable of delivering knowledgeable social comments, adapted to individual profiles and game progress for tabletop multiplayer computer games. Our framework utilizes input such as game events and player profiles (containing information about player's name, occupation, age, social status, etc.) and keeps track of incoming information about game progress to *decide and deliver player-adapted and context-sensitive social feedback*.

In particular, comparing to existing methods, we propose a game presenter character with adaptive emotions through multiple emotion states, each state adapted and related to the performance, progress, and profile of each individual player. During game play, and for the current player, the character performs transitions of the corresponding emotion state, while making such emotions visible to the audience through animated facial expressions and eventually deciding player-adapted social comments. This way, the presenter is able to adopt a social behavior that enhances the overall gaming experience. As a case study, we discuss the implementation and evaluation of such a presenter character following our framework for tabletop computer games.

Our framework reveals the first-class components inherent in building presenter characters, with clear separation of concerns across the various components. The latter enables independent development of components and supports better software organization, while it facilitates extensibility and maintainability. Overall, the framework prescribes responsibilities and interoperation among components in a way that can be used to realize different presenter behaviors for various game categories.

In order to prove our concept, we have implemented a game presenter character for *2D board games*, called Amby, based on the proposed framework (a two-minute video of our character in action is available at http://www.ics.forth.gr/hci/files/plang/AmbyVideo.avi). Amby receives raw game events as input and processes them for producing valuable game comments and computing self-emotions. These comments are further adapted to each player separately, based on their social profiles and their in-game progress. Our character also incorporates a range of emotions and communicates them to the audience via animated facial expressions. Amby adjusts his emotional state for each player separately, based on their social profile and their game progress.

Amby's setup was designed to resemble TV-show game setups as illustrated on Figure 1. He is displayed on a separate screen close to the game board and communicates with the game via network messages. He is capable of delivering comments by utilizing a wide range of multimedia, namely, text, images, animations, sound effects, background music, and speech. Amby is also capable of displaying facial expressions to convey emotions.

Our focus was to prove that a game presenter character following the proposed framework is able to make knowledgeable, personalized comments and that his/her presence enhances gaming experience and contributes to player satisfaction. Moreover, the produced system is easily extended and adjusted to fit different games' categories. Our evaluation proves that players favored Amby's presence in the game and found his behavior interesting and engaging. The choices made for each one of the framework components when implementing this case study are detailed within the respective sections. The user evaluation of our work is analyzed in Section 4.

2. Related Work

More and more games today try to incorporate adaptation mechanisms that adjust game parameters to fit individual gameplay. Nowadays, a technique named dynamic game difficulty balancing (DGD) is used, in which the game monitors player performance in order to adjust its difficulty accordingly. For example, Hunicke and Chapman's game [12] controls the game environment settings in order to make challenges easier or harder, while Resident Evil 5 [13] employed a system called the "Difficulty Scale" that grades the players' performance and uses it to adapt the enemy difficulty level. Finally, Mario Kart series [14, 15] features distributing bonus items that help an individual driver get ahead of their opponents based on a driver's position. Other games allow for static personalization through preferences. This method allows players to adjust game attributes themselves before play, in order to fit their preferences. Finally, there is an increasing trend towards using genetic algorithms and machine-learning techniques in order to learn from player's gameplay and adjust the nonplayer characters to become more and more challenging. For example, Demasi and Cruz [16] built intelligent agents employing genetic algorithm techniques to keep alive agents that best fit the user level. All these games monitor player activity and make decisions about adjusting game properties (level difficulty, enemy strategy, etc.). Our work also monitors game activity in order to collect information about each player's progress and the overall game state. However, we focus on game

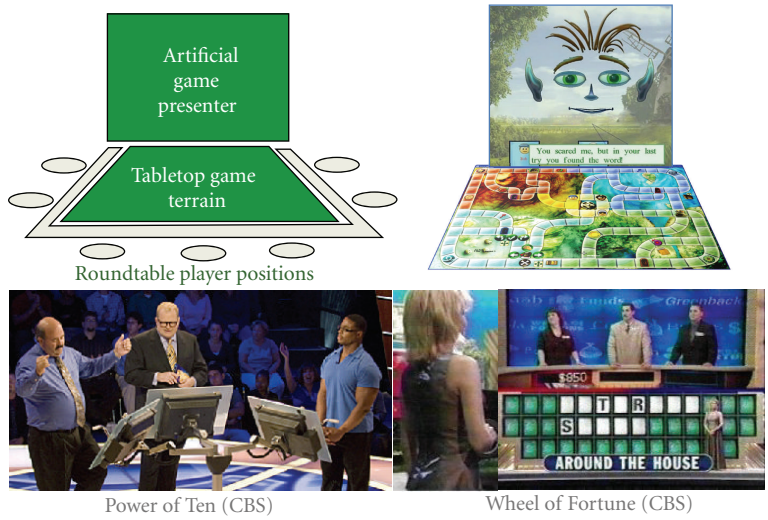


Figure 1: Setup of the Amby presenter character for a tabletop game, resembling TV-show games.

presenter characters; so in our case, the data collected through game monitoring is used for adapting the comments made by the presenter. We also use players' profiles in order to personalize comments and provide adaptive social feedback. The latter is widely used in pedagogical agents with great success [7–11]. Such agents are intended to boost learning experience and are incorporated into educational games. They are aware of the player profile and they adjust their behavior (in the form of feedback) in order to keep them interested and motivated. For example, Conati and Manske [17] discuss how to improve the feedback provided to players, while [18] adds socially oriented conversational abilities to an existing "teachable agent" in order to enhance student's experience within the game. In our work, we use adaptive social feedback for building game presenters for any game, without focusing on learning. Educational game presenters may use individual player information in order to react differently for each one of them and for making knowledgeable comments that fit their profile. Our character differs in the means that we use multiple versions of state parameters per player profile and that we allow adaptation rules to be based on comparisons with other players. For example, comments can be made about which player is doing better in game, which has the highest score, and so forth. Finally, the use of emotions is gaining ground in games. Complicated mechanisms for capturing and interpreting user emotions are incorporated in games that further adapt their gameplay according to these emotions. For example, an abstract cognitive architectural model is proposed in Imbert and de Antonio [19] aiming to support emotions and social behavior, although emotion-driven reasoning is not an explicit component. In affective games, researchers try to find ways of inducing, processing, and utilizing players' emotions in order to take them into account when designing their games. Player's emotions can be utilized to dynamically adapt the gameplay in order to keep players challenged and motivated. Gilleade and Dix used frustration in the design of adaptive videogames [20], while Sykes and Brown try to induce player's emotions through the gamepad [21] in order to control difficulty level in a game. Finally, The Affective Mirror (AM) [22] is an affective multimodal interface that adapts itself to the user's perceived affective state. In our work, we utilize emotion states for the presenter, in order to enhance the social feedback provided. We do not use any mechanism of capturing actual player emotions and we do not make any assumptions about them either. Our target is to build presenter characters that can dynamically build emotions for players based on their profiles and their game progress. The adoption of affective computing features on top of the current implementation may further enrich the personality of our presenter; however, such extra effort was beyond the scope of the discussed work.

Game presenter characters are not new in the bibliography. There are several commentator systems able to present sport or TV-show-like games. These systems try to resemble human behavior. In sport games, these systems work pretty well, as the real TV commentators take into account only game events and are not interested in viewers' personalities or emotions. For example, successful sport commentary systems are incorporated in commercial games like Pro Evolution Soccer [2]. Also, a lot of work has been done for artificial commentators in RoboCup games (see [4, 5]). Closer to our work is the "*Byrne*" system [6], in which a digital character provides automatic soccer commentary and supports facial expressions and verbal narrative *adapted to the current game state by processing annotated game events*. Byrne's automatic narration relies on the notion of observers, being special objects which make abstract conclusions about the game, while they may also post remarks. Some technical issues regarding Byrne are as follows: (i) emotion generation does not consult the previous emotion states (has no emotional memory); (ii) the type of remarks by observers cannot be globally tuned (cannot adapt to a particular audience); (iii) it has no adaptive behavior (cannot comment on the social profile of a particular soccer player). Byrne is targeted for RoboCup commendation so it takes into account

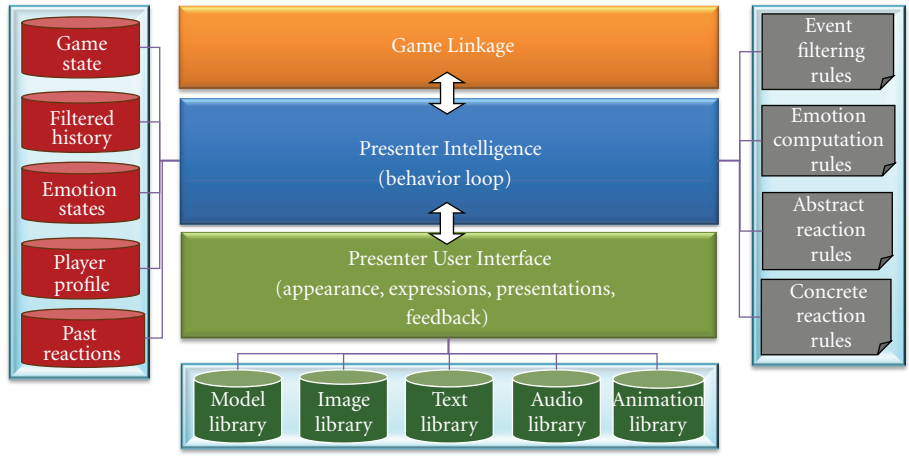


Figure 2: The high-level architecture of the framework.

only player progress. The proposed framework, however, supports the construction of presenter characters that make social comments, in the means that (i) comments that are not game specific and (ii) they can be personalized through player's profiles and action history. For example, comments such as "well done John" and "Suddenly you're doing a lot better that your opponents" can be characterized as social because they are personalized and can be made in any game, in opposed to "great tackling" and "nice uppercut" that can be made only in specific kind of games and do not contain any personalized information. In our framework, the progress of several players can be also compared and used inside the comment lines (e.g., "You have more points than Maria").

Finally, game presenters are also incorporated into TV-inspired video games. These presenters are based only in events received from the respective game and have no adaptation towards the players' social profiles. Quiz-like games such as Buzz! [3] but also video-game versions of many TV shows (Who Wants to Be a Millionaire, Wheel of Fortune, Deal, etc.) incorporate 3D presenters with complicated graphics, expressions, and body postures who, although being able to comment on game progress, are completely ignorant of player's emotions and social profiles. As a result, they make the same comments over and over again, regardless if they address to kids, adults, males, or females. Our work discusses game presenters that are aware of the players, their profiles, and their progress in order to make comments. They also utilize emotion states and facial expressions in order to be more expressive. We present an integrated extensible and adaptable framework for building such characters for any multiplayer game, and we discuss our experience from implementing such a game presenter for multiplayer board games (case study).

3. Framework

This section presents the necessary information that will guide other researchers for enhancing their games through artificial presenter characters that incorporate emotions and are able to make player-adapted, knowledgeable comments. Firstly, we will present the software architecture of the framework (Section 3.1), and then we will detail each of its components in separated sections. In Section 3.2, we will present the component linking the game and the presenter character (Game Linkage), and in Section 3.3, we will analyze the behavioral loop of our character focusing on the enhanced sense-think-act loop and detailing each state separately. Finally, in Section 3.4, we will present the user interface component of the proposed architecture.

3.1. Software Architecture. The system's software architecture, which is visualized under Figure 2, was designed to provide a flexible yet powerful model for building artificial game presenters. Components with different functional roles in the system are clearly separated, providing the flexibility to developers to choose among different implementations for each component. Appropriate communication protocols (APIs) are further defined in order for the different architecture components to cooperate.

The overall architecture is divided into three major components: *Game Linkage*, *Presenter Intelligence*, and *Presenter User Interface*. *Game Linkage* component is the one connecting the game with the presenter character, so it has to provide a communication protocol by which the two components will exchange information. The *Presenter Intelligence* is the framework's larger component. It encapsulates an improvement of the traditional sense-think-act loop for game characters by introducing *reflect* and *adapt* as two extra processing stages to support affective processing and adaptive reactions, respectively. Such extra stages are inherent in the role of our character as a game presenter and are likely unnecessary for typical nonplayer characters of traditional games. The Presenter Intelligence needs to consult the game state, the player's progress, the player's profile, the filtered history, and their own past emotions and reactions in order to make decisions about which game events to comment and the way in which these comments are to be delivered (facial expressions, animations, images, text, speech, audio,

or music). This information is presented on the left part of Figure 2, while the AI rules which describe the decision-making process are displayed on the right part of the same figure. Finally, the *Presenter User Interface* component is displayed at the bottom part of Figure 2. This component practically undertakes the look and feel of the presenter, being also responsible for providing all the functionality necessary for delivering the multimedia-enhanced comments and the character's emotions. A communication protocol is necessary here to describe this functionality.

3.2. Game Linkage. The Game Linkage component provides all the necessary functionality for the game presenter to communicate with the host game. For this purpose, a communication protocol must be agreed between them, describing the specific messages and the content that the two parts will exchange. The majority of these events depend on the game genre; however, they should include every game specific aspect that the presenter would be interested to, such as game state information and specific player actions. Table 1 summarizes the most common events needed for multiplayer tabletop board games. In Section 5, more examples of such communication protocols are given that would suit other game genres as trivia and race games.

Both the presenter and the game side must implement such a communication protocol in order to cooperate. Whenever a player performs an action in the game, or the game state changes for some reason, the presenter must receive the corresponding event and pass it to the Presenter Intelligence component in order to be capable of making valuable comments.

3.3. Presenter Intelligence. In order for game presenters to support social comments, we propose a behavioral model that builds on the traditional sense-think-act loop for game characters. More specifically, we argue for two extra processing states: *reflect*, that computes the presenter's new emotional state, and *adapt,* that takes into account the players' social profile along with their overall game progress in order to decide the exact comment or reaction for them. Overall, the behavioral loop breakdown is as follows.

(a) *Sense*: in this state, the presenter character receives several stimuli from its environment. Input signals can also be enriched with extra semantic content via filtering rules. For example, a dice roll can be characterized as low or high depending on the number and sides of the dices rolled.

(b) *Reflect*: the presenter updates its current emotional state. The presenter "recalls" its emotions for the player in turn and decides a new emotional state depending on the previous one and the received game events.

(c) *Think*: this state utilizes the results from the sense and the *reflect* states in order to decide an abstract reaction suitable for the current game situation. For example, the presenter may decide to comment on a player's repeated failures to open a door in the game. The exact reaction will be decided in the adapt state.

(d) *Adapt*: in this state, the presenter takes into account the player's profile in order to infer the most suitable way to convey the comment decided. For example, repeated failures to open a door may result in a humorous comment and a cartoonish animation if the player is younger than 10 years old or in a criticizing comment if the player is adult. This state also decides the multimedia combinations to use for delivering these comments.

(e) *React*: finally the chosen reactions are to be realized through the character's visualization components. Several multimedia such as images, videos, sound effects, music, animations, text bubbles, speech synthesis, and facial expressions (emotions) must be synchronized in order to convey the decided comments and emotions to the players and audience. Essentially, this state produces the output of the presenter character, which will be given to the *User Interface* component for rendering.

The output of each state is available to all subsequent states. The complete behavioral loop along with a visual explanation of each state is illustrated under Figure 3. All states are detailed in their respective sections further in this paper.

As we stated before, the presenter needs to process input events and player profile information in order to come up with emotion reactions and player-adapted social comments. In order to make decisions, the character needs some written AI logic, in the form of rules. This is called *Rule System* and comprises the backbone of the character's AI. In order to build a rule system, programmers are free to choose among several programming languages. Typical general-purpose rule-based systems like Prolog and scripting languages such as Python and Lua are among the most popular choices. Other options include standard programming languages like C# or Java, ad-hoc rule languages like XML and customized rule languages such as DMSL (Source code publicly available from: http://www.ics.forth.gr/hci/files/plang/DMSL.ZIP). The latter was introduced by Savidis et al. [23] and was our choice for our case study for several reasons. DMSL is a small-scale easy-to-use domain specific language with flexible syntax. It supports an external profile as a dynamic record, where user profiles can be encompassed and application oriented data can be introduced. Moreover, the specification of independent decision rules can be made by defining uniquely named decision blocks; this makes the language verifiable relying on predicate calculus.

We strongly argue for separating the character's AI rules from the rest of the system. By doing so, one can gain better control over the character's logic, and programmers are able to experiment with the behavior rules without interfering with the rest of the system. The presenter can easily be programmed to adopt different personalities depending on the current set of rules that they follow. For example, a set of rules can correspond to decisions that a kind-hearted

Table 1: The most common messages of the communication protocol (API) for tabletop board games.

Message ID	Message content	Explanation
Start game	Number of players	New game session begins
End game	—	Current game session ends
Set initial score	Player Id, score	Defines the score with which a player starts. It may not be the same for all players since some games may adopt player characters with different abilities and starting values
Set initial lives	Player Id, total lives	Defines which the initial player life is. It may not be the same for all players since some games may adopt player characters with different abilities and starting values
Score change	Player Id, score variation	Triggered when a player's score points changes
Lives change	Player Id, lives variation	Triggered when a player's life changes
Extra turns	Player Id, turns variation	Notifies the presenter whenever a player loses or gains one or more turns
Turn begins	Player Id, turn number	Notifies the presenter about the current player (whose turn begins now)
Turn end	Player Id, turn number	Notifies the presenter that a player have just finished their turn
Possible paths	Player Id, number of possible paths	Triggered when a player needs to make a choice among several possible paths to proceed
Door pass Door fail	Player Id, door Id	Triggered when a player tries to pass through a game door
Cross-point pass	Player Id, cross-point Id	Triggered when player crosses a certain point of the game path (defined by the game developer)
Pause/resume	Is paused	The game is paused/resumed
Play multimedia	Player Id, media type, media URL	The game specifically asks the presenter to reproduce a multimedia file

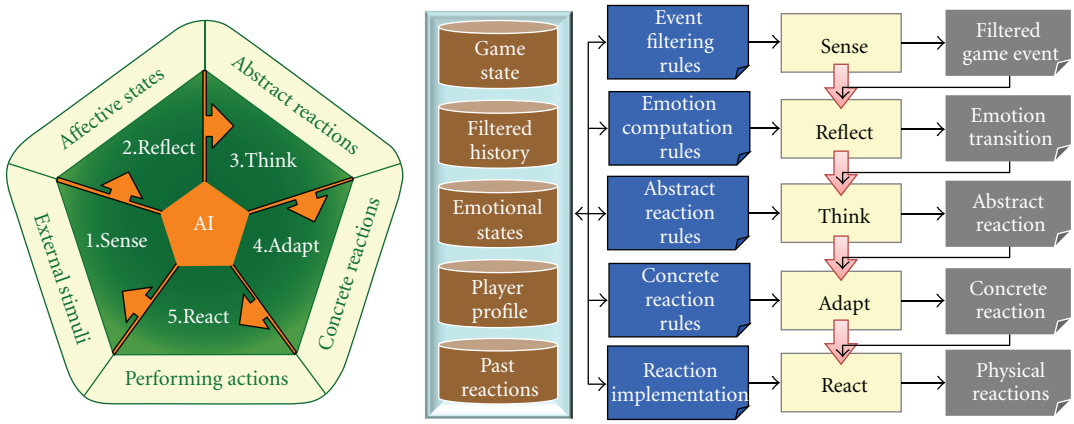


Figure 3: The proposed behavior loop for game presenter characters. Although not shown for clarity reasons, the output of each state is available to subsequent states if needed.

presenter would make, while another set of rules can describe decisions for an evil presenter that enjoys player failures. If the AI is decoupled from the rest of the code, then the programmers are able to switch presenter behaviors just by switching the AI rule set. In our case study, we utilized external files containing DMSL rules in order to define the character's personality. By altering the rule files (written in readable text form), or by switching them with others, we could experiment with our character's behavior.

3.3.1. Sense

(a) Event Interception and Filtering. Monitoring of game and player activities is technically analogous to interaction monitoring, the latter constituting an essential component of adaptive interfaces. It conceptually maps to the *sense* stage of the behavior loop, realizing the basic perception capabilities of the presenter. Practically, by the means *sense,* we describe all kinds of input that a presenter character can get. This input comes from the Game Linkage component. Some of these inputs must be further tagged with extra semantic information in order to be processed correctly. Input events that contain numeric values, for example, should be characterized based on the value's scale. A game presenter cannot say whether 10 score points from a bonus is considered good or poor, because it depends on maximum score points that one can get from bonuses and how often the bonuses contain such score points. Similarly for a racing

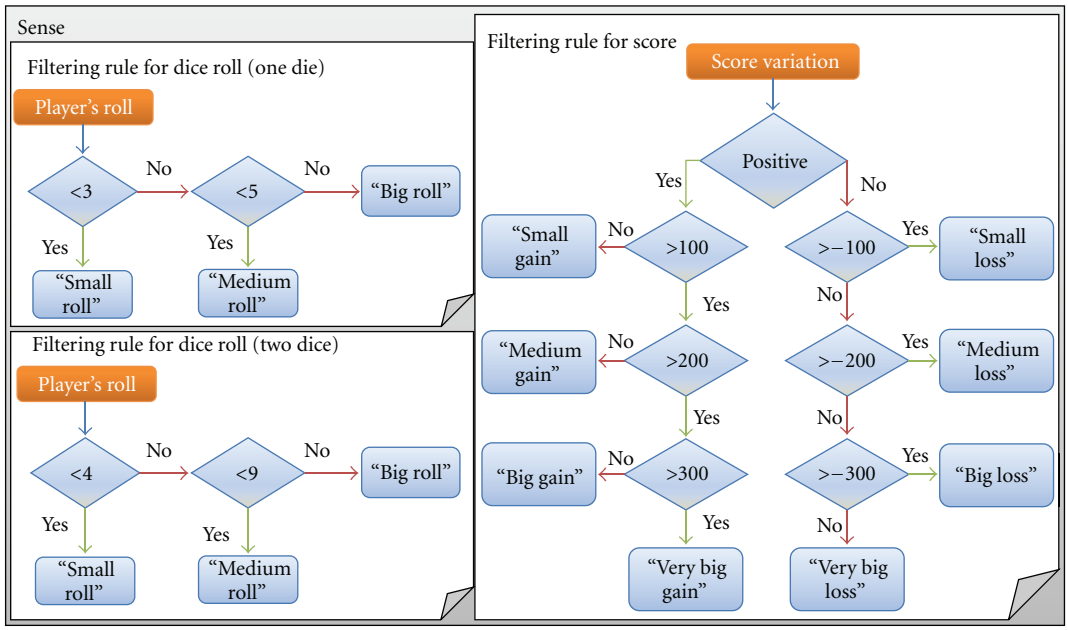


Figure 4: Examples of filtering rules.

game, filtering rules can contain game-specific information such as a car's maximum speed in order for the presenter to determine if the player is going fast enough or not. More examples about the usage of filtering rules in other game concepts are given in Section 5.

In this sense, extra logic is required to "filter" such values and practically map their absolute value to a value range or an abstract value characterization ("big score gain," "medium roll," "few path choices," and so on). This can be achieved by the use of external filtering rules, which provide this extra context to the presenter. In our case study, we received raw events from the game and we used filtering rules to add semantic context to the ones that did not make sense without it. In Figure 4, some examples of filtering rules are depicted.

Filtering rules are also useful when we need to adapt the presenter's behavior to fit another game of the same genre. Instead of altering numeric values across all AI rules, one can just alter the filter rule that controls that value. For example, consider a presenter character for arcade games. Many of them have very similar game mechanics but differ when it comes to score and life points. In that case, the same presenter character would need only to change the filtering rules for score and life points instead of searching through the AI rules for finding hard-coded values and changing the character's logic there.

(b) Game History Bookkeeping. In order to make knowledgeable comments about each player progress in game, the events received in the *sense* state should be kept in memory for further processing and statistic analysis. This procedure helps the character "remember" the game progress for each player separately and be aware of their habits, their choices, and their game history in general. This kind of data is useful for game commentators—actually the event bookkeeping and the assumptions that can be made out of data analysis can make the difference between a mere announcer of the game events and a game commentator. In our case study, previous actions in the game can trigger comments such as "now that you obtained the key, you should be able to unlock the door that you couldn't pass earlier" or "you keep losing score for three rounds in a row." Such comments could help the players through the game and keep them constantly engaged and active. Furthermore, they can stimulate the game audience, as they can hear and see a game presentation close enough to human presentation, that is, containing interesting knowledgeable comments instead of mere event announcements.

3.3.2. Reflect (Emotion State Transitions). The reflect state in the character's behavior cycle is where the presenter's emotions are decided. The presenter keeps a self-emotion state for each player, which is decided based on player actions. For example, the presenter remembers that they were happy about player 1, disappointed for player 2, and so forth. Whenever a player's turn comes, the presenter recalls their emotion state for this player and uses it in order to decide new emotion states. These states' outcome can be further used for comment decision-making in the subsequent states.

The reflection mechanism helps for better gaming immersion and enhanced gaming experience and is very successful in TV-show games where human presenters tend to convey their own emotions to the public. Notice that as players take turns, the presenter may rapidly switch moods as it returns to its previous emotional state with each player. We used this cyclothymic behavior as a mechanism to provoke humor in the game. Our hypothesis was confirmed during the evaluation session, where the majority of the players

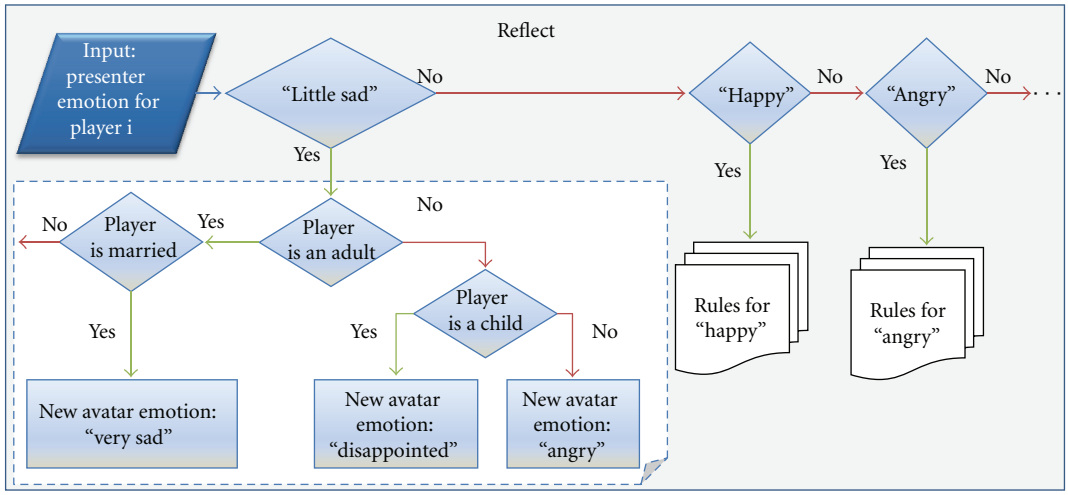


Figure 5: Emotion rules for "Big Score Loss." The presenter was previously in a "Sad" emotional state because the player had lost a lot of scores. As player lost score again in this round, the presenter computes a new emotional state.

Figure 6: The affective state space of our character as an adaptation to the Circumplex model.

actually found this behavior pleasant and funny. However, if such a behavior is not desired, developers can use the presenter's emotion for player i-1 (who have just finished their turn) as an extra input to the reflect state and use decision rules to provoke smoother emotion transitions for their character. We did not use any such mechanism.

In our proposed framework, we emulate the reflection mechanism by inserting the reflect state inside the presenter's behavior loop. As we mentioned above, the presenter keeps an emotion state for each player. Each time an input event comes for a player (via the sense state), the presenter character should compute his/her new emotional state based on his/her previous state and the input event. Such computation is made via reflection rules described by the rule system. An example of *reflect* rules is displayed in Figure 5.

Game designers can base their presenter character's emotion transitions on any cognitive model in order for their character to be believable. Reflection rules can be defined for the presenter emotions based on this model. For each player, the presenter keeps an action history thus being aware of player progress and recent actions. Based on this progress, the presenter can have expectations regarding the player and make appropriate emotion transitions according to the current game event following the cognitive model. It should be recalled that game events are semantically rich, carrying information on everything that happens regarding the game progress, not merely restricted to device input. In our case study, we based Amby's emotional transitions on a simplified version of the Circumplex model [24], illustrated under Figure 6. Comparing to the original Circumplex model, our version does not contain aspects that we consider meaningless in an entertainment context such as (i) *fear* and *disgust* emotions; (ii) distinction among *activation* and *deactivation* since we consider no tangible trophy affecting player lives after the game; (iii) intermediate states for *sadness* like *guilt* or *depression*. Transitions are allowed only between

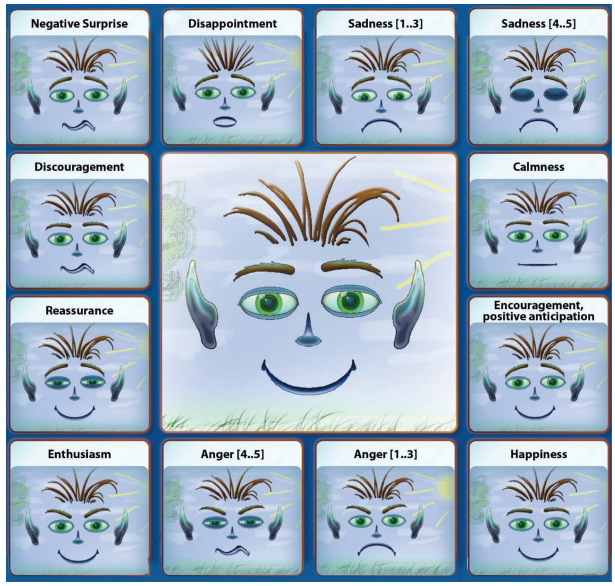


Figure 7: Amby's facial expressions that reflect *his emotions for the players*.

neighbour emotions. Any transition is allowed to and from the "Calm" state.

We argue that the presenter should incorporate a wide range of emotions with intensity values (e.g., one can be a little or very angry). The more emotion states a presenter supports the more realistic would the presenter's behavior be. In our case study, we utilized 7 basic emotions, being "Happiness," "Sadness," "Enthusiasm," "Reassurance," "Anger," "Disappointment," and "Calmness." Amby, however, was capable of making three more expressions, "negative surprise," "Discouragement," and "Encouragement" which we used as transitions between the basic ones. Amby's facial expressions that reflect the basic emotions and the transition expressions are displayed in Figure 7. The majority of our evaluators found Amby to be expressive enough and the range of emotions to be sufficient.

3.3.3. Think (Deciding Abstract Comments). The *think* state follows the *sense* and the *react* state in the behavior loop as displayed in Figure 3. The *sense* state receives external input events from the host game and filters them to induce extra semantic information via filtering rules. Next, the react state computes the new emotional states for the presenter. Taking into account all this information, the *think* state is now in charge of deciding the character's actions. In this state, only abstract reactions are decided (next state, i.e., adapt, will undertake the refinement of those abstract reactions into specific commands). Together, *think* and *adapt* states of the behavior loop form the character's AI. The utilization of the rule system is necessary for both states in order to make decisions.

To give an example of the think process, let us say that we play a game and come to a point where the player had lost a lot of score points in the previous two rounds. At the current round they gain—let us say—a third of their score back. The *think* state must decide if a comment should be made about this. The flowchart diagram of the logic rules that will guide the presenter through this decision is given in Figure 8. The output of that state is the decision to "comment on regaining part of score." We will later see in the adapt state (in Figure 9) how such decisions can result into more personalized and adapted comments for each player.

3.3.4. Adapt (Adapting Social Comments). Imagine that you play a board game and a friend of yours sits next to you and comments on your progress. Your friend is aware of your profile, your social status, your name, your age and they can think of excellent comments that a virtual game presenter would never made. We propose the same idea for all kinds of game presenters. Based on our evaluation study, we argue that being aware of the player profile results into more engaging comments for both payers and audience. A presenter should react differently if they address players with different social profiles. For example, if the presenter decides to show an image in order to encourage the player, a cartoon-like image will be chosen for a child while a more serious one will be chosen for an adult. Additionally, for adults, the game presenter can make use of players' social status, occupation, hobbies, and so forth to comment on their in-game decisions. For example, in a quiz game, the presenter can excuse a player for not knowing the answer because *"it is outside of their job field,"* while if it is a child, the presenter will be able to make a comment about *"not learning stuff like that at school."* In the proposed framework, every player has a profile containing key aspects of their social status along with their name and their age. The profile is loaded at startup, and remains in memory during the game so that the presenter character would be able to consult it at any time.

Taking into account the abstract reaction that was decided in *think* state, the *adapt* state consults the player profile in order to decide more concrete reactions for the presenter. The adaptation process takes place with the use

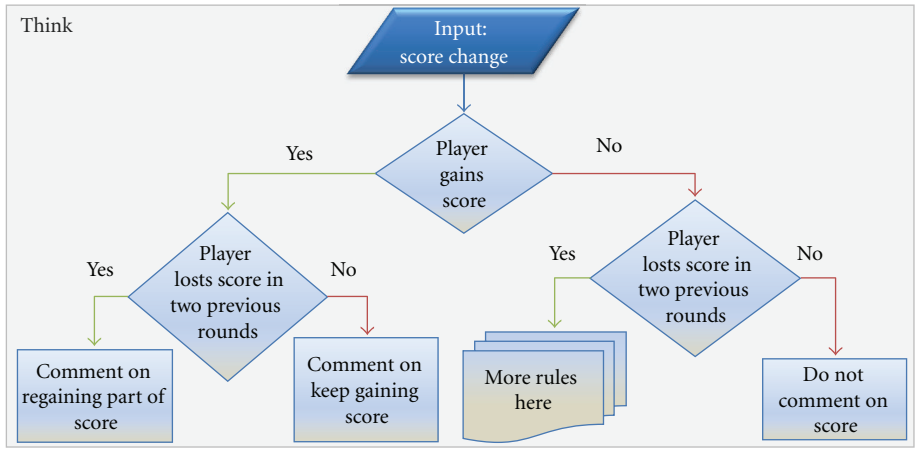


Figure 8: Example of a decision rule diagram used in the *think* state. The output decision suggests that a comment action should be made about "regaining part of score." The latter will be fed to the adaptation rules for further processing.

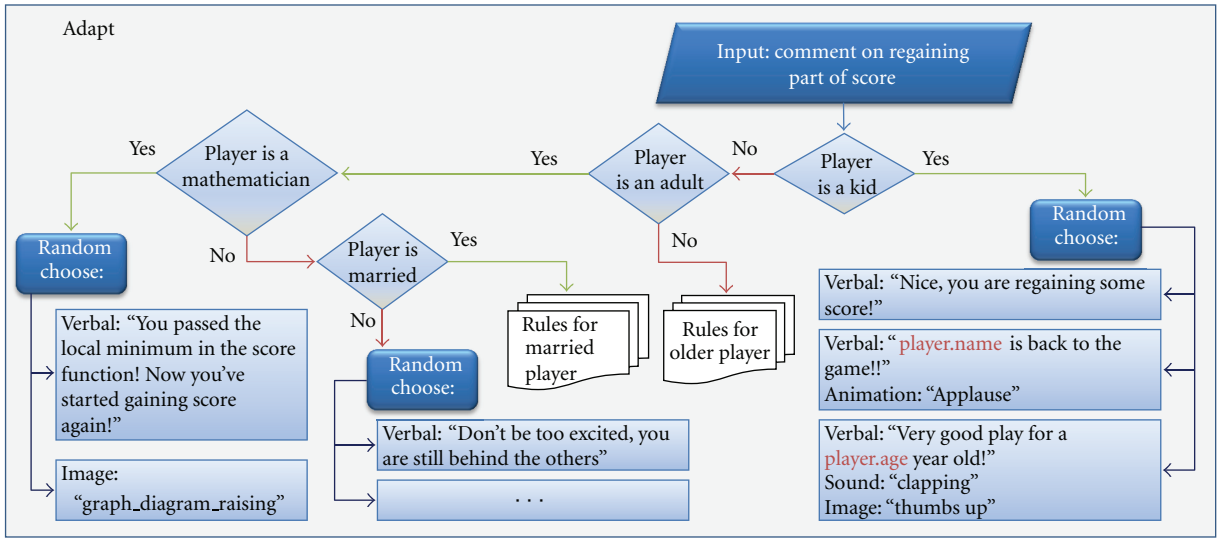


Figure 9: How a presenter character's decision can be further adapted to match a player profile. Notice that the final comment choice is made among reaction combos. In verbal comments, the "player.name" and "player.age" will be replaced with the corresponding attributes from player profile.

of the rule system. In the general case, adaptation rules may radically differ from the AI logic, so they can be described in another language (using another rule system) or provided by a separated component. In every case, the adaptation system should provide rules for adjusting each of the abstract decisions induced from the *think* state in order to be able to produce concrete presenter reactions. Presenter reactions may contain several multimedia, which can be either combined or used separately. The adaptation logic provider can define combinations of multimedia that match together and can be played concurrently to deliver the desired presenter behavior. For example, a closing door animation can be combined with a creaking sound effect, and an encouraging text comment may be accompanied by a cheering sound and an image depicting a gold medal. A multimedia file may be contained in a lot of combos, and combos themselves can contain random choices (any appraisal comment can be combined with any cheering or clapping sound).

By organizing the presenter's reaction in this way, we can ensure that their behavior will be coherent no matter how many multimedia are chosen to be displayed concurrently. Regarding the presenter's behavior, an extra mechanism is needed to ensure that the character will not repeat their comments too often. We can achieve this by remembering the presenter's past reactions and excluding them from the reaction pool. Only when all possible reaction combos are used, the presenter is able to choose one of them again. This way, and provided that a wide set of reaction combos are available, the presenter character will seldom repeat his/her comments. A simple example of the adaptation logic is given in Figure 9. The *think* state has decided to deliver a comment

Table 2: A generic communication protocol between the user interface and the Presenter Intelligence components.

Message ID	Message content	Explanation
Load animations	URL	Load animations from a given path
Load audio	URL	Load audio files (sound effects and music) from a given path
Load images	URL	Load images from a given path
Set emotion expression	Expression Id	Instructs the presenter character to perform the respective facial expression for transitioning to the given emotion
Show comment	Text, position, font, size, color	Instructs the presenter to draw a text comment to screen. Arguments font, font size and color are optional
Show image	URL, position, size	The presenter is ordered to display an image on screen
Play audio	URL, volume, repetitions	The presenter is ordered to play an audio file
Notify audio finished	Audio notification Id	When the audio finishes, the presenter need to notify the intelligence framework through an appropriate notification id
Play video	URL, volume, position, size	The presenter is ordered to play a video on the screen
Notify video finished	Video notification Id	When the video finishes, the presenter needs to notify the intelligence framework via calling this given function
Play animation	Animation Id, animator	User interface framework needs to provide a way of displaying animations
Notify animation finished	Animation notification Id	When the audio stops, the presenter needs to notify the intelligence framework via calling this given function
Say comment	Comment text	Uses speech synthesis
Set background image	URL	Allows setting the background of the presenter character
Set presenter size	Size, position	The presenter character may need to change size in order to emphasize on something or in order to make room for other displaying objects (like images, videos, and animations)
Set presenter position	Position	The presenter may need to move around the screen

about a player regaining some of their score. The player's age is used for picking a set of proper reaction combos. Then a reaction combo is picked randomly from the set. Notice that the reaction sets do not contain past reactions.

In our case study, we incorporated player profiles containing information about their names and their ages. The players' profiles can be modified dynamically during gameplay, incorporating values that hold information about their game progress (for convenience, every player-related attributes that are considered useful to the presenter can be stored directly into the player's profile). The result was two-fold: the players were addressed by name, and our character could adjust his output comments to fit their age range. Evaluation results showed that all players were happy to hear their names from Amby's mouth, and most of them noticed the different treatments among the different ages. Additionally, diversifying the output comments helped the presenter to avoid repeating himself, adding plurality to his expressions, and thus keeping the players and the audience stimulated. The player's profiles helped in the presenter's emotional computation too, since he was programmed to be more soft-hearted towards younger players and more austere to others.

3.3.5. React (Delivering Comments and Expressions). The final state in the behavior loop of our framework is *react.* This state's target is simple: bring the comment decisions made at the previous states to life. In order to do that, the framework needs to utilize the Presenter User Interface component which is capable of displaying the desired multimedia. Additionally, the same component will take over the looks of the presenter character and perform their facial expressions in order to convey their emotions. In the react state, the presenter character receives the decisions taken from the *reflect* and *adapt* states (character's emotions and comments resp.). These decisions must be mapped to specific messages to send to the *Presenter User Interface Component* such as "Show Image," "Play Audio," or "Set Emotion Expression." This functionality is provided by the Presenter User Interface component through a specified API (an example of such an API is displayed in Table 2).

The *react* state has also the role of action coordinator. As each presenter's comment may contain concurrent animations, sounds, images, text, and so forth, these must be well organized and synchronized perfectly in order to provide a smooth output. The presenter's facial expression should be also tuned and timed, and smooth transitions between any two emotions must be programmed. The latter is necessary, because the presenter changes emotions and facial expressions frequently, and more often than not a new emotion is completely different from the previous one. For example, at the beginning of each player's round the presenter recalls the previous emotion for the player, which can contradict the current emotion. In our case study, we used intermediate state as medians whenever such transitions should happen. Timing has the first role when it comes to playing other multimedia too. Some of the issues that need

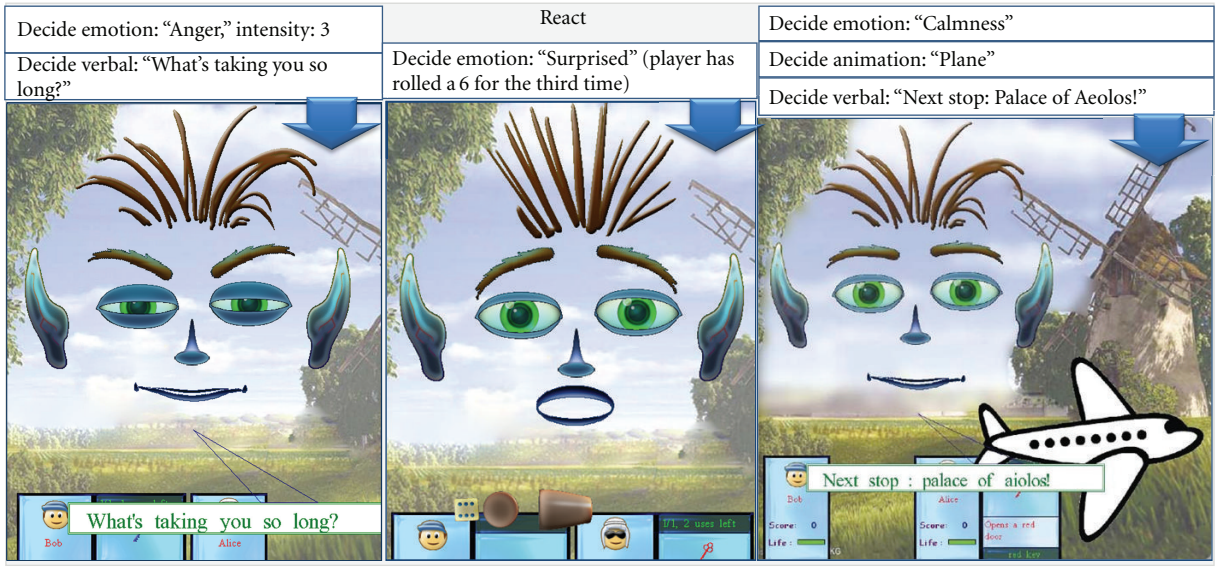


Figure 10: Examples of Amby reactions. At the top part, we display the decisions that correspond to each visual image.

to be addressed in this state are the following.

(i) Sound effects must accompany animations when decided together.

(ii) Images and text should be displayed for a while and then disappear.

(iii) Animation films and images should not hide the verbal comments.

(iv) If the character utilizes a speech synthesis tool.

  (a) One verbal comment must wait for the previous one to stop first (otherwise multiple voice comments will sound concurrently).

  (b) Lip synchronization must be scheduled carefully in order to feel natural.

(v) The presenter's face should never freeze. Emotions must come and go, and blinking must be scheduled frequently.

(vi) Facial expressions must change smoothly from one to another.

In Figure 10, we display three screenshots of our presenter character (Amby) to show how facial expressions, comments, and images can be combined towards a smooth result. A video of Amby is also available online (a two-minute video of our character in action is available at http://www.ics.forth.gr/hci/files/plang/AmbyVideo.avi).

3.4. Presenter User Interface. In order to visualize the presenter character, a user interface component is required. Such a component must provide all the necessary functionality for handling the multimedia libraries and displaying the presenter character along with the multimedia-enhanced comments on a screen. Programmers are able to choose among several options depending on each presenter's needs (GUI libraries, game engines, standard programming languages such as Java, C#, and so on). The user interface component undertakes the management of the media libraries, that is, the 2D or 3D models, still images, animations, audio files, and videos. In order for the user interface and the Presenter Intelligence components to communicate, it is necessary to establish a protocol offering the required functionality. In Table 2, an example of such a protocol is presented.

In our case study, we adopted an in-house game engine for 2D games in order to build our character's user interface. The adoption of a cartoon-like appearance for our character was mainly due to practical reasons, since our primary objective has been to develop a game presenter with emotions as an adaptive system, rather than to experiment with animated facial expressions. In this context, we designed an anthropomorphic character, with emphasis on outlines of facial characteristics rather than on visual details. This way, we had the flexibility of easily deriving expressions by transforming only key features such as the eyes, eyebrows, and mouth. Finally, all character activities, including facial expressions, social feedback, and the maintenance of game statistics, are handled through generic components manipulating external media files such as images, audio, or animation definitions. This way, not only distinct actions were handled by merely mapping to corresponding media files but also the presenter's reactions became easily extensible.

4. Evaluation

4.1. Process. The evaluation process has been coordinated by usability experts and involved 9 users aged between 8 and 35. The users have been invited to evaluate the game presenter character of a tabletop game, with particular emphasis put in collecting feedback on whether the overall game experience

is improved compared to the game alone. The evaluation of the game per se was not part of the process since we needed to focus only on the evaluation of the game presenter. The procedure is described below.

All users were put together in one group, and the evaluators explained the game mechanics to them. Participants were then free to play the game for 15 minutes without Amby's presence and then play another 15 minutes with Amby (no player or player activities have been simulated). In both sessions, the participants were asked to think aloud in order for the coordinators to take notes. After the gaming sessions, they were asked to complete a questionary about their experience. The questions used the *Likert scaling technique* [25] to measure user's satisfaction about Amby. Some of the investigated topics were which version of gameplay do players preferred, whether Amby made the game more engaging for the players and the audience, whether he was expressive enough (sufficient number of emotions), and whether he was annoying or not. A couple of questions were used to evaluate Amby's adapted comments per se; however, it was made clear to the participants that comment lines were written by game authors based on their personal gaming experience and they were in no way representative for all players (e.g., game authors think that a comment such as "come on, it's not the end of the world" would be encouraging for a child and can be chosen by a presenter being in a encouraging state when addressing to a child). For commercial games, however, a presenter's comment lines should be written by psychologists and game experts in order to represent his/her emotions and be adapted to players' profiles. Nevertheless, they turned out to be fairly good. Finally, the participants were asked to enumerate a few things they liked and/or disliked about the game presenter character. Their comments along with their answers to the questioner were collected and analyzed. The most important findings are listed below.

4.2. Results. As we stated before, all the answers given from the participants were collected and analyzed by the usability experts. In Figure 11, some of the most important questions are presented, along with the participants answers to them. As the questions used the Likert scale, each participant ranked a given statement from 1 to 5, 1 representing the "total disagree" and 5 representing the "total agree" to it. For each question, the weighted average value of all answers is presented (blue bars).

Additionally, user comments about what they liked and/or disliked about our game presenter character were summarized. We present the evaluation results below.

Things that players liked about Amby were as follows:

(i) cartoonish appearance (younger players mostly),

(ii) expressive animations,

(iii) use of several multimedia for commenting the game,

(iv) most of the comments were unique,

(v) content-aware comments,

(vi) personalized comments.

Things that players did not like about Amby were as follows:

(i) cartoonish appearance (adult players mostly),

(ii) cannot adjust the volume level,

(iii) Amby's complains about not playing quick enough (part of Amby's logic decided to make a comment like "*What's taking you so long?*" each time a player was idle for a long time),

(iv) Computer-generated voice. Proposed prerecorded audio files,

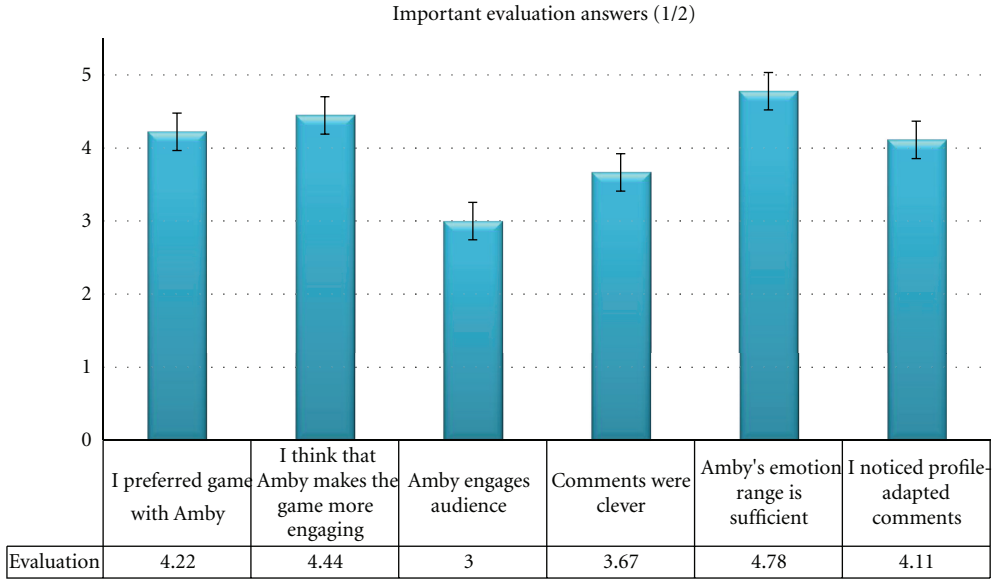
(v) some comments were not as good as hoped.

The evaluation showed that although Amby was a case study implementing only part of the proposed framework, his presence clearly enhanced the gaming experience. Amby's appearance issues can be resolved by better graphics and/or by utilizing another UI component, volume level is just a matter of configuration, and time to wait until making a comment for idle players as well as the comments made are changed easily via altering the AI logic, which is decoupled from the rest of Amby's structure. Finally, the synthetic voice always lacks proper fluctuation, tone, and emotion in comparison to human voice. However, recorded voices can be incorporated very easily as well. Each recorded phrase would be an audio file with a unique id, and the rules inside the *adapt* state would make sure that verbal comments (as texts) and corresponding audio files are decided together (as combos).

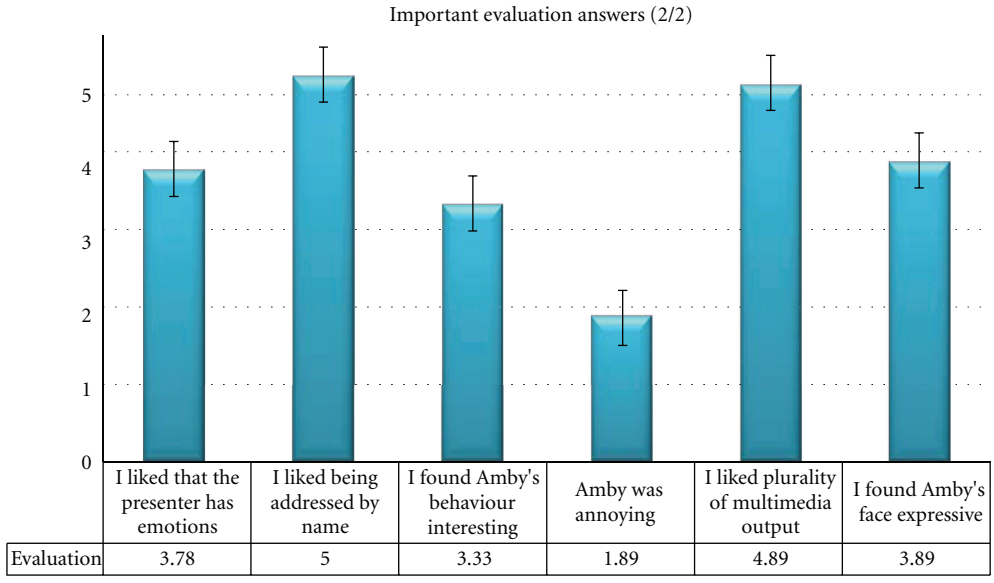
5. Discussion

Although most of our examples refer to tabletop board games, the framework can be utilized for building presenters for almost any multiplayer game. The key lies in the fact that the Presenter Intelligence is based merely on game events and player profiles and is decoupled from the rest of the game code. The game designers should supply the presenter with game events (Game Linkage) and write logic rules based on these events. For example, if we needed to build a presenter for a quiz game, the communication protocol between the game and the presenter would include events regarding the questions, the help available, timing issues, player progress, and so forth. Table 3 summarizes an example of the most common events for this kind of games. The Presenter Intelligence component should utilize logic rules that rely on these game events along with player profile to make knowledgeable comments, such as "For a lawyer, you answer physics questions pretty fast," "your kid would know how to answer that," and "although you are the youngest player, you've reached the bonus first!" and so on.

For a nonturn-taking game, whenever the presenter receives an *important* game event for a player, the respective player profile is directly supplied to the rule system. Unimportant events are either filtered by higher-level rules or are ignored. Such a game example might be a racing game, where all players play concurrently, that is, there is clearly no turn-taking discipline. In this case, the game would



(a)



(b)

Figure 11: Evaluation diagrams for the most important questions about Amby. For the evaluation we used the Likert scale. The diagrams illustrate the weighted average values (their numeric form can be seen below each question) given by 9 evaluators. The black lines represent the standard error.

post events to the presenter in real time, reflecting actual player actions or related progress. An indicative list of such events is shown in Table 4. Then, an appropriate set of rules should be encompassed in the intelligence component of the framework to decide comments with a social style suited to this particular genre.

Game presenters should be directly aware of the game they present, so, inevitably many of the events in the Game Linkage are specific to a particular game. But also, the majority of such events can be reused across games of similar genre or common game mechanics. Most of the reflection and intelligence rules would also apply to a wide range of similar games or games with similar mechanics. For example, rules that refer to dice should be common to any game that adopts the dice mechanism. Filtering rules (detailed in Section 3.3.1(a)) can be used to fine-tune details such as the number of dice used in each game. The user interface component doesn't need to change.

In order to introduce alternative presenter characters for the same game, one should have two or more different rule sets describing the presenter actions in each case. For example, if a game has a good presenter and game authors wish to experiment with an evil one, they would have to make a rule set that describes this behavior and

Table 3: The most common messages of the communication protocol (API) for Trivia games.

Message ID	Message content	Explanation
Start game	Number of players	New game session begins
End game	—	Current game session ends
Turn begins	Player Id, turn number	Notifies the presenter about the current player (whose turn begins now)
Turn ends	Player Id, turn number, answered correctly, time up	Notifies the presenter that a player have just finished their turn. Also informs the presenter why the turn ended (player gave an answer or time was up)
Question	For player Id, category, type, Is checkpoint, time to answer	Notifies the presenter that a question for player X has been asked. The Category of the question (e.g., history) and the question type (e.g., multiple choices) must be known, along with how much time has the player to answer
Player pressed buzzer	Player Id, time	Notifies the presenter whenever a player presses their buzzer. The presenter can compare the time from such events and comment on player reflexes
Player asks help	Player Id, help type, helps left	Notifies the presenter that a player has asked for help
Player reached checkpoint	Player Id, check point Id	Notifies the presenter that a player has reached a checkpoint in the game
Score (Money) changed	Player Id, amount	Notifies the presenter whenever a player gains or losses score (or money)
Pause/Resume	Is paused	The game is paused/resumed

Table 4: A sample of common messages of the communication protocol (API) for race games.

Message ID	Message content	Explanation
Start race	Number of players	New game session begins
End race	List of players and positions	Current game session ends. The list of players is used to denote their order (who came first, whether a player was close enough to finish, whether they are left behind)
Route selected	Route, miles, difficulty, day/night, forward/backward, location	Notifies the presenter about the selected route.
Overtake	Player 1, player 2, time	Notifies the presenter whether a player is now ahead of another.
Gained bonus	Player, bonus	Notifies the presenter each time a player gets a bonus
Turbo boost	Player, amount	Notifies the presenter if a player uses turbo
Pit stop	Player, elapsed time	Notifies for a player pit stop and the elapsed time
Checkpoint	Player, checkpoint, time	Notifies whenever a player reaches a checkpoint
Pause/Resume	Is paused	The game is paused/resumed

chooses the appropriate comments. The latter may be a time-consuming process, since the respective rule set may be quite big and complicated in order to take into account as many aspects of player profiles and game progress as possible. However, once the rules are there, switching behaviors at startup is only a matter of loading the respective rule set. The proposed framework does not support switching behaviors during gameplay. Images, animations, and sound effects can also be changed by selecting the respective resource paths without affecting other implementation aspects of the presenter. In this case, the Game Linkage and the user interface components do not require modifications since all changes are local to the Presenter Intelligence component, while the communication semantics are not affected.

Finally, for adopting an alternative appearance for the presenter, one needs to replace only the user interface component. Both Game Linkage and the Presenter Intelligence component are not affected. For example, game developers wish to upgrade from 2D to 3D, they need to implement the visualization and animations using an appropriate 3D graphics library.

The proposed architectural framework is modular and flexible to be applied for building game presenters across various game categories. Apparently, game presenters need to convey game-related comments. Following our framework, the adjustments required to support games of similar genres are practically of a small scale.

6. Additional Features

In this section, we will discuss our experience about the presenter character we implemented, along with some extra features we added. As mentioned before, our game presenter character, named Amby, is designed to comment on multiplayer turn-taking tabletop board games. We used a board game named "The four elements" developed by an in-house game engine. The game features multiple players, each of them having score points, a life-bar (0–100), and

Figure 12: An example of player's state and inventory as shown on Amby's screen. Both players have their inventories open. The blue shadow behind Bob's head indicates that it is his turn to play.

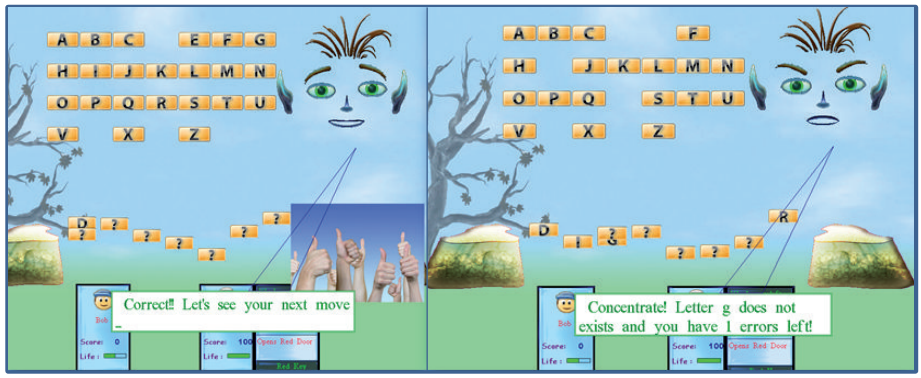


Figure 13: Screenshots from the Hangman minigame.

an inventory able to hold items. Throughout the gameplay, the player can pick up, drop, use, and lose inventory items. Some of these items are used to unlock doors inside the game; others can be used to transfer to specific locations, others to give life, and so forth. Each inventory item has a name, a description, an icon, and usage times (e.g., a life potion may have up to two usages). Without Amby's presence, the players with their lives and inventory are all displayed on the game terrain via soft dialogues [26]. As our game presenter character was designed to function on a separate screen from the game, we decided that he should undertake the display of the player icons, their score, their life, and their inventory in order to free some space on the board-game screen. Players should be able to interact with their inventory with game controls, and Amby should display the inventory for them. Amby was also responsible of displaying the dice (again without Amby it would be displayed on the game terrain as well). In Figure 12, an example of player state and inventory as shown on Amby's screen is displayed. Specified containers display the player's avatar (or photograph or icon), their name, their score points, and a life-bar showing their life points left.

The blue shadow behind the player icon indicates whose turn it is (in Figure 12 its Bob's turn). We also display player inventories next to player containers. Due to space limits, Amby displays only one item at a time. From top to bottom, we display item's placed within the inventory list (1/1 means this is the first object in the list containing 1 item), how many usages are left for this object, it's icon, a two-row text description, and its name. All inventory-control events are received from the game via the "*Game Linkage*" component. The same applies for inventory commands. When a player needs to see their inventory, he or she uses the dialogs provided by the game in order to control what they see on Amby's screen.

Another additional feature we implemented for Amby is a set of minigames offered by Amby to the host game. The latter is unaware of their implementation details and their gameplay but is capable of requesting them and utilizing their outcome when players finish their play. More specifically, we have implemented two minigames, Hangman and a card-draw. The game may have in-game obstacles that introduce challenges for players in order to proceed. Such a challenge may require a player to succeed in a mini game. In this case, the game will ask Amby to start a minigame session providing some extra information such as the difficulty level. Amby consults the player profile (through think/adapt processes) and decides the specific game parameters for the mini game. To give an example, let us say that the game requests Amby to start a *difficult Hangman* minigame for a player. Amby consults the player profile to decide the word, as a difficult word for a child may be easy for an adult. Then the game session starts. As all input events must come from the game (so that the player doesn't change controls), Amby requests letters from the game. As the player provides these letters, the game sends them to Amby, and he decides if the letter belongs to the word or not and proceeds accordingly. Of course, Amby remains active and keeps commenting on player' progress within the mini game. When the minigame finishes, Amby informs the game about the outcome (win or lose, time elapsed, incorrect letters given, etc.) and continues commenting on the main game. Figure 13 contains two

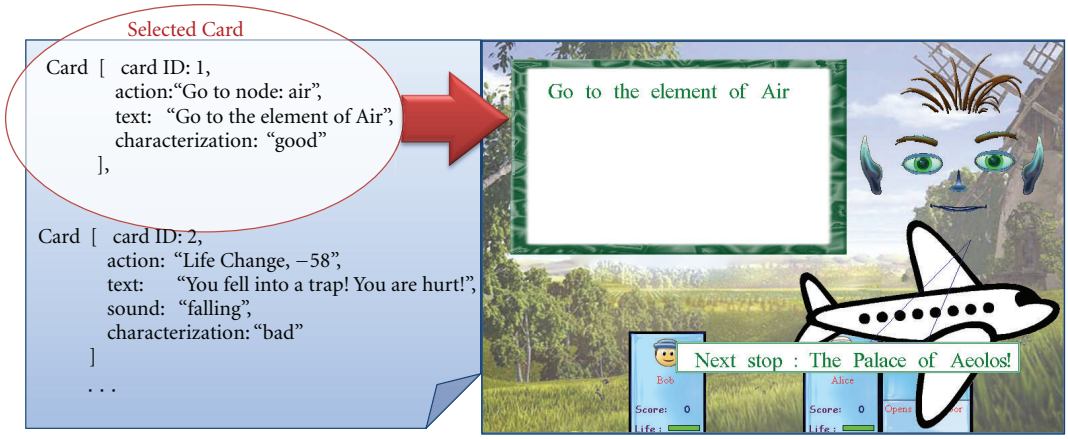


Figure 14: An example of the card-draw game. On the left, we display part of the configuration file that contains all cards. In this example, the random choice yields the first card.

screenshots of our implementation of the Hangman mini game.

The second mini game that we supported was a card-draw. Essentially the game asks Amby to draw a card, display it to the player, and send the card action to the game. The cards are described into configuration files, and Amby makes a random choice among them. Each card describes the action that will be sent back to the game, a text containing the legend of the card and a characterization. The last field is necessary because Amby doesn't have AI rules for deciding specific reactions for each card. Instead, generic rules are provided for commenting on card categories—for example, "if the card's action is a "*Go to*" then decide to show an airplane or a boat animation" and "if the card's characterization is good, then set Amby emotion to Happy."

An example of a configuration file for the cards along with Amby's reaction when drawing it is displayed in Figure 14. When the game is over, that is, when the card is drawn and displayed, Amby informs the game about the result, that is, the card's action. The game then executes this action and the gameplay continues. In our example, Amby sends to the game the action "Go to node: air."

7. Summary and Conclusions

In this paper, we have presented a framework for building artificial game presenter characters with emotions, capable of delivering knowledgeable social comments, adapted to individual profiles and game progress. The framework has three major components being Game Linkage, Presenter Intelligence, and Presenter User Interface, each of them having a distinct role. Game Linkage provides the presenter with events containing player actions and game state, being practically everything that a game presenter might be interested in. The Presenter Intelligence computes the character's actions and emotions following an enhanced version of the classical sense-think-act loop, being sense, reflect, think, adapt, and react. States reflect, think, adapt, and react utilize the corresponding sets of logic rules, describing the character's actions and emotion transitions based on player's profiles, their in-game action history, and the overall game progress. Finally, the Presenter User Interface component utilizes some graphic library to synchronize and deliver the decided actions. The latter includes facial animations for delivering presenter's emotions and a wide range of multimedia for commenting purposes, being sound effects, music, images, videos, animations, text, and speech synthesis. Following our framework, we built a presenter character for a multiplayer tabletop board game and put it under evaluation with 9 users.

In this sense, our work demonstrates the feasibility of a generic framework for game presenters with adaptive social comments. The emphasis on the architectural split between the game input linkage, core character intelligence, and the user interface allows for the framework applicability to different kinds of games. The presented framework also enables researchers to experiment with alternative presenter behaviors, and adopting more advanced character visualization since modifications in the Presenter Intelligence component does not affect Game Linkage and user interface. Additionally, the evaluation process has concluded that the game presenter improves the overall game experience and amplifies the social interaction within multiplayer game sessions. The combination of humor together with social comments relying on personalization through player profiles and gaming history played a key role in this context. Another important element improving game experience is how comments are delivered, involving several multimedia resources such as text, images, video, animations, sound effects, music, and facial expressions. The latter is used to convey presenter emotions to players and improve the overall social interaction. In this work, we did not focus on computing the actual emotions of players, something strongly related to the affective computing field and usually deploying computer vision. We followed an alternative path of making an artificial presenter dynamically building emotions for players based on their profiles and the way they act and perform in the game. The adoption of affective

computing features on top of the current implementation may further enrich the personality of our presenter; however, such extra effort was beyond the scope of the discussed work.

We believe that our framework can be a guide for game developers towards creating presenter characters for their games, encompassing emotions and adaptive social feedback. Moreover, we hope to inspire researchers to experiment with alternative presenter behaviors and appearances in several kinds of games.

Disclosure

This paper has not been published elsewhere (except in the form of an abstract or as part of a published lecture, review, or thesis) and is not currently under consideration by another journal published by Hindawi or any other publisher.

References

[1] B. Gunter, "Understanding the appeal of TV game shows," *Zeitschrift Für Medienpsychologie*, vol. 7, no. 2, pp. 87–106, 1995, PhycINFO.

[2] Pro Evolution Soccer Series, http://en.wikipedia.org/wiki/Pro_Evolution_Soccer_(series).

[3] Buzz! Game series, http://www.buzzthegame.com/en-gb/.

[4] D. Voelz, E. Andrè, G. Herzog, and T. Rist, "Rocco: a RoboCup soccer commentator system," in *Proceedings of the ROBOCUP-98: ROBOT Soccer World Cup II*, vol. 1604/1999 of *Lecture Notes in Computer Science*, no. 50-60, 1999.

[5] E. Andrè, K. Binsted, K. Tanaka-Ishii, S. Luke, G. Herzog, and T. Rist, "Three RoboCup simulation league commentator systems," *AI Magazine*, vol. 21, no. 1, pp. 57–65, 2000.

[6] K. Binsted and S. Luke, "Character design for soccer commentary," in *RoboCup-98: Robot Soccer World Cup II*, M. Asada and H. Kitano, Eds., vol. 1604 of *Springer Lecture Notes in Computer Science*, pp. 22–33, 2008.

[7] C. Conati and M. Manske, "Evaluating adaptive feedback in an educational computer game," in *Intelligent Virtual Agents*, vol. 5773/2009 of *Lecture Notes in Computer Science*, pp. 146–158, 2009.

[8] C. Conati and M. Klawe, "Socially intelligent agents in educational games," in *Socially Intelligent Agents—Creating Relationships with Computers and Robots*, K. Dautenhahn et al., Ed., Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[9] C. Conati and X. Zhao, "Building and evaluating an intelligent pedagogical agent to improve the effectiveness of an educational game," in *Proceedings of the International Conference on Intelligent User Interfaces (IUI '04)*, pp. 6–13, Island of Madeira, Portugal, January 2004.

[10] P. H. Tan, S. W. Ling, and X. Y. Ting, "Adaptive Digital Game-Based Learning Framework," in *Proceesings of 2nd International Conference on Digital Interactive Media and Entertainment and Arts (DIMEA '07)*, pp. 142–146, Perth, Western Australia, 2007.

[11] S. M. Fisch, "Making educational computer games 'educational'," in *Proceedings of the Interaction Design and Children (IDC '05)*, pp. 56–61, June 2005.

[12] R. Hunicke and V. Chapman, "AI for dynamic difficulty adjustment in games," in *Proceedings of the 19th National Conference on Artificial Intelligence*, pp. 91–96, usa, July 2004.

[13] Resident Evil 5 Official Strategy Guide, "Prima Publishing," 2009.

[14] Mario Kart series, http://en.wikipedia.org/wiki/Mario_Kart.

[15] Dynamic Game Difficulty Managing, http://en.wikipedia.org/wiki/Dynamic_game_difficulty_balancing#cite_ref-0.

[16] P. Demasi and A. Cruz, "Online coevolution for action games," in *Proceedings of the 3rd International Conference on Intelligent Games and Simulation*, pp. 113–120, London, UK, 2002.

[17] C. Conati and M. Manske, "Adaptive feedback in an educational game for number factorization," *Frontiers in Artificial Intelligence and Applications*, vol. 200, no. 1, pp. 581–583, 2009.

[18] A. Gulz, M. Haake, and A. Silvervarg, "Extending a teachable agent with a social conversation module effects on student experiences and learning," in *Proceedings of The 15th International Conference on Artificial Intelligence in Education*, vol. 6738/2011 of *Lecture Notes in Computer Science*, pp. 106–114, Auckland, New Zealand, 2011.

[19] R. Imbert and A. de Antonio, "An emotional architecture for virtual characters," in *Proceedings of the International Conference on Virtual Storytelling (ICVS '05)*, Springer Lecture Notes in Computer Science, no. 3805, pp. 63–72, 2005.

[20] K. M. Gilleade and A. Dix, "Using frustration in the design of adaptive videogames," in *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE '04)*, pp. 228–232, Singapore, 2004.

[21] J. Sykes S, "Affective gaming: measuring emotion through the gamepad," in *Proceedings of the Extended Abstracts on Human Factors in Computing Systems (CHI EA '03 CHI '03)*, 2003.

[22] S. Shahid, E. Krahmer, M. Swerts, W. A. Melder, and M. A. Neerincx, "You make me happy: using an adaptive affective interface to investigate the effect of social presence on positive emotion induction," in *Proceedings of the 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops (ACII '09)*, pp. 1–6, Amsterdam, The Netherlands, September 2009.

[23] A. Savidis, M. Antona, and C. Stephanidis, "A decision-making specification language for verifiable user-interface adaptation logic," *International Journal of Software Engineering and Knowledge Engineering*, vol. 15, no. 6, pp. 1063–1094, 2005.

[24] J. Russell and G. Lemay, "Emotion concepts," in *Handbook of Emotion*, M. Lewis and M. Haviland-Jones, Eds., Guilford Press, New York, NY, USA, 2000.

[25] R. Likert, "A technique for the measurement of attitudes," *Archives of Psychology*, vol. 22, no. 140, pp. 1–55, 1932.

[26] A. Savidis and Y. Lilis, "Player-defined configurable soft dialogues: an extensible input system for tabletop games," in *Proceedings of the 5th ACM International Conference on Interactive Tabletops and Surfaces (ITS '10)*, pp. 287–288, November 2010.