IDEAS4Games: Building Expressive Virtual Characters for Computer Games

Patrick Gebhard[1], Marc Schröder[1], Marcela Charfuelan[1], Christoph Endres[1], Michael Kipp[1], Sathish Pammi[1], Martin Rumpler[2], and Oytun Türk[1]

[1] DFKI, Saarbrücken and Berlin, Germany, `firstname.lastname@dfki.de`,
[2] FH Trier, Umwelt-Campus Birkenfeld, Germany, `m.rumpler@umwelt-campus.de`

Abstract. In this paper we present two virtual characters in an interactive poker game using RFID-tagged poker cards for the interaction. To support the game creation process, we have combined models, methods, and technology that are currently investigated in the ECA research field in a unique way. A powerful and easy-to-use multimodal dialog authoring tool is used for the modeling of game content and interaction. The poker characters rely on a sophisticated model of affect and a state-of-the art speech synthesizer. During the game, the characters show a consistent expressive behavior that reflects the individually simulated affect in speech and animations. As a result, users are provided with an engaging interactive poker experience.

1 Motivation

Virtual characters are widely used in a variety of applications, including computer games, where they are notably used for non-player characters, i.e. characters controlled by the computer. In general, virtual characters have the purpose to enrich the game play experience by showing an engaging and consistent interactive behavior. Because this issue influences acceptance in general, computer games, and virtual characters in particular, have to be designed carefully according to Loyall's *suspension of disbelief principle* [1]:

... a character is considered to be believable if it allows the audience to suspend their disbelief ...

The creation of interactive expressive characters with a consistent behavior comes with a whole range of challenges, such as interaction design, emotion modeling, figure animation, and speech synthesis [2]. The interactive drama game Façade [3] or the Mission Rehearsal Exercise Project [4] explicitly address these problems in combination, in an integrated application. However, relevant research is also being carried out in a range of relevant individual disciplines, including believable facial, gesture and body animation of virtual characters [5] [6], modeling of personality and emotion [7] [8], expressive speech synthesis [9] and control mechanisms for story as well as high-level interaction [10].

In the project IDEAS4Games, we investigate how modern ECA technologies can help to improve the process of creating computer games with interactive expressive virtual characters. Based on the experience of a computer game company (RadonLabs, Berlin), we identified four main challenges in creating computer games:

- *Fast creation of game demonstrators*. In order to compete with other game companies the implementation of demonstrators has to be fast and reliable.
- *Localization of game content*. To sell games in other countries content has to be translated into the respective language. The more dialogs a game contains, the higher the costs for the translation.
- *Easy interaction*. The success of a game is tremendously related to an easy interaction concept.
- *Consistent quality*. The quality of audio and visual presentation should be consistent for the whole game. Every exception lowers its acceptance.

Based on our experience in the research area of ECAs, we rely on the following models and technologies for addressing these challenges.

- *Flexible multimodal dialog and story modeling*. Other than the approach of Bosser et al. [11] that is based on an autonomous agent framework for computer games, we rely on the authoring tool SceneMaker [12] for modeling game interaction and dialog content. SceneMaker is able to create a Java executable that flexibly integrates other modules.
- *Sophisticated simulation of affect in real time*. Affect is key to an advanced behavior modeling of virtual characters [21]. Based on a simulation of affect, one can transfer aspects of human affective behavior onto virtual characters to enhance their believability.
- *Expressive speech synthesis*. To enhance the consistent affective behavior of characters, utterances should reflect their affective state. In addition, if it was possible to use synthesized speech rather than recording every utterance needed in the dialog, this would help to control the costs for content localization. It would also enable new game features such as the integration of a player's name in dialogs.
- *3D virtual characters*. Our characters provide lip-sync facial expressions, body expressions such as breathing, and gestural animations.

We focus on the employment of virtual characters in interactive social games, like board or card games. They provide a huge field of application of virtual characters that are in the role of competitors and enrich the overall gaming experience by an engaging performance.

2 Poker Demonstrator

In order to demonstrate that it is possible to meet the requirements stated above, we created a poker computer game and presented it at the CeBIT 2008 fair to a large number of people to collect initial feedback.

By using real poker cards with unique RFID tags, a user can play draw poker [13] against the two 3d virtual characters Sam and Max (see Fig. 1). Sam is a cartoon-like, friendly looking character, whereas Max is a mean, terminator-like robot character. Both are rendered by the open-source 3d visualisation engine Horde3D [14]. The human user acts as the card dealer and also participates as a regular player.

As shown in Fig. 1, we use a poker table which shows three areas for poker cards: one for the user and one for each virtual character. These areas of the table are instrumented with RFID sensor hardware, so that the game logic can detect which card is actually placed at each specific position. A screen at the back of the poker table displays an interface which allows users to select their actions during the game, using a computer mouse. These include general actions, such as playing or quitting a game, and poker game actions: bet a certain amount of money, call, raise, or fold. This screen also shows the content relevant for the poker game: the face of the user's cards, the number of Sam's and Max's cards respectively, all bets, and the actual money pot. The two virtual characters are shown above this interaction screen on a second 42" monitor.

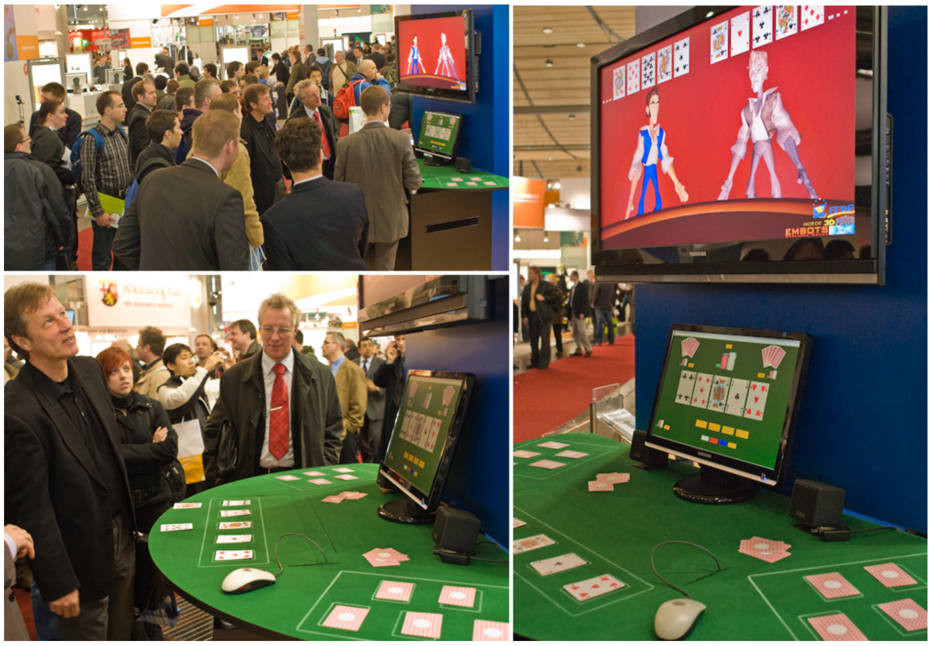


Fig. 1. Poker Demonstrator at the CeBIT 2008 fair

When a user approaches the poker table and initiates a game, Sam and Max explain the game setup and the general rules. In a next step the user has to deal the cards. During the game the virtual characters Sam and Max react to events, notably when the user deals or exchanges their cards. Time is also considered – for example, Sam and Max start complaining if the user deals the cards too slowly, or they express their surprise about erratic bets.

In order to support Sam's and Max's individual character style, different poker algorithms are used. Sam, who represents a human-like poker player, uses a rule based algorithm, whereas Max, the robot poker player, relies on a brute-force algorithm that estimates a value for each of the 2.58 million possible combinations of five poker cards.

Based on game events, the affect of each character is computed in real-time and expressed through the character's speech and body. The richly modeled characters, as well as the easy interaction with the game using real poker cards, have stirred up a lot of interest in the CeBIT 2008 audience (see Fig. 1).

2.1 Poker Algorithms

Brute Force Approach The brute force strategy starts by calculating an evaluation function, which maps a hand (5 cards) to a numeric value. The higher the value, the better are the chances for its owner to score. Our approach is to put all 2,598,960 possible hands in its *natural* order, which is given by the rules of the game and the fact of one hand winning against another in direct comparison. After generating and sorting that list, we assign values to the hands in an inductive manner:

- The lowest hand in the list is assigned a value of zero.
- For each following hand we assign values as follows:
  1. If the comparison of this hand with the previous hand would result in a draw game, the same value is assigned.
  2. If this hand would win against the previous hand, then its value is the one of the previous hand plus the amount of hands that had the same value as the previous hand.

The main purpose of the algorithm is to determine an action to be performed by the player. When playing Draw Poker, the player has a choice at one point to exchange up to three cards. There are 26 options to do so: One option not to exchange cards at all, five to exchange one card, ten to exchange two cards, and ten to exchange three cards. For each option the expected value of the outcome can be calculated. When no card is exchanged, it is simply the evaluation value of the hand. In case of exchanging cards, it is the average value of all possible combinations of the remaining cards with cards from the remaining 47 cards of the deck. In order to determine the best possible action, we have to check $1 + 5 * 47 + 10 * 1,081 + 10 * 16,215 = 173,196$ resulting hands and their value.

Rule-based Approach Our rule-based approach was implemented in *JESS* (*Java Expert Systems Shell*) [15], a fast Java-based expert system. The main advantages of a rule based approach are: (a) short development time, (b) readability of code, and (c) ease of implementing variations in terms of "playing style". The latter is of high importance when an AI module is to be used in conjunction with virtual character that should display a personal style. The drawback of rule-based systems are usually (a) performance and (b) maintainability of large code bases. However, for our application performance time of this module was negligible and the code base was quite small. The JESS module is called in the *draw* phase of the game. It handles the recognition of the system's

```
(defrule predict-full-house-1m-a
        "When having two pairs, add prediction full house"
        (not (hand (type full_house)))
        (hand (type two_pair) (indices $?i))
        ⇒ (assert (maybe (type full_house) (missing 1) (have ?i))))
```

Fig. 2. An example of a JESS rule

current hand, prediction of possible hands after the draw and the decision which cards to exchange (see Fig. 2 for a sample rule).

This is done using 36 rules, organized in 4 modules. The system's strategy is to keep the highest current hand and exchange all others (up to 3). To determine which cards to exchange the system consults its computed predictions, usually aiming for the highest potential hand achievable with the fewest new cards. Here, heuristics can be easily changed, e.g. to model risky players, beginners or cautious players, due to the small and readable code base.

3 Modeling of Interaction and Story

The behavior of the two virtual players has been modeled with the authoring tool SceneMaker. Our authoring approach relies on the separation of dialog content and narrative structure, which we have introduced in [12, 16].

3.1 Authoring Dialog Content

Dialog content is organized in *scenes* - pieces of contiguous dialog. Scenes are defined in a multimodal script that specifies the text to be spoken as well as the agents' verbal and nonverbal behavior. The utterances can be annotated with dialog act tags that influence the computation of affect (see Section 4 for details). In addition, system commands (e.g., for changing the camera position) can be specified. Scenes (see Fig. 3 for an example) are created by an author with standard text processing software. The major challenge when using scripted dialog is variation. The characters must not repeat themselves because this would severely impact their believability. For this purpose we use blacklisting: once a scene is played, it is blocked for a certain period of time (e.g., five minutes), and variations of this scene are selected instead. For each scene, several variations can be provided that make up a *scene group*. In our poker scenario there are 335 scenes organized in 73 scene groups. The number of scenes in a scene group varies between two and eight scenes, depending on how much variation is needed.

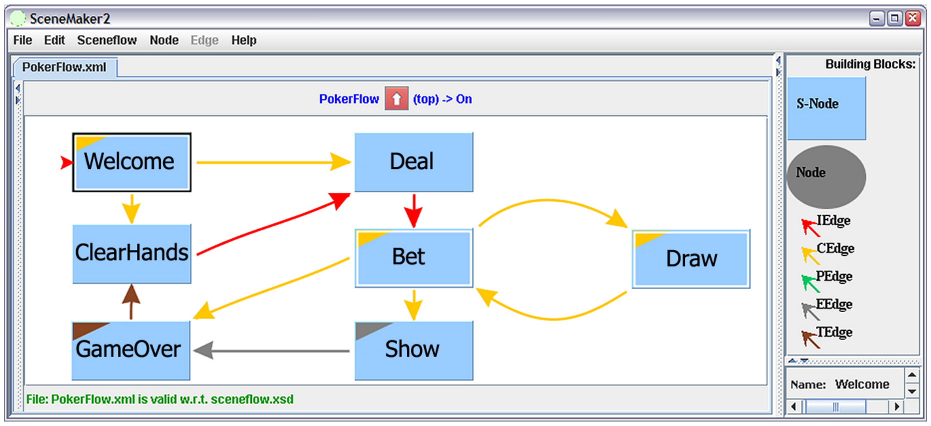
Scene_de: Welcome(3)
Sam: **[camera 1]** Hi. It's great to have [point] you here for a game. **[GoodEvent 0.7]**.
Max: My pal **[S look-to-other]** and I **[nod]** were a bit bored the last seconds.
Sam: **[look-to-user]** Poker isn't a game that can be played by only two ... **[M mod]**

Fig. 3. A Welcome scene from the multimodal script

The name of the scene group is followed by the variation number in parentheses. Each dialog contribution starts with the characters name. At the beginning of this scene the camera is moved to position 1. Sam's first utterance is annotated with the appraisal tag [**GoodEvent**] because the arrival of the user is appraised positively by him. This tag will elicit the emotion *joy* with a certain intensity. The gesture specification in Max's utterance **[S nod]** shows that it is possible to specify gestures of other characters, e.g. for back-channeling behavior. The multimodal dialog script is parsed by the scene compiler which generates a single Java class file for each scene.

3.2 Authoring Narrative Structure

The narrative structure – the order in which the individual scenes are played – is defined by the *sceneflow*. Technically, the sceneflow is modeled as a hypergraph that consists of nodes and edges (transitions). *Supernodes* contain subgraphs. Each node can be linked to one or more scenes and scene groups. Different branching strategies (e.g. logical and temporal conditions as well as randomization) can be used by specifying different edge types [12]. The sceneflow is modeled using the sceneflow editor, our graphical authoring tool that supports authors with drag'n'drop facilities to *draw* the sceneflow by creating nodes and edges (see Fig. 4).



File: PokerFlow.xml is valid w.r.t. sceneflow.xsd

Fig. 4. Sceneflow editor showing the supernode *On*

Fig. 4 shows the graph structure for the supernode *On* representing the situation that a human user is interacting with the installation. After an initial welcome phase (represented by the supernode *Welcome*), he or she is either prompted to remove the cards from the table (*ClearHands*) or to deal out the cards to start a new game (*Deal*), etc.

At runtime the sceneflow graph is traversed by selecting nodes and edges based on the current game state and the actions of the three players. The scenes that are selected and executed during such a traversal control the multimodal behavior of the virtual agents. Transitions in the sceneflow are triggered either by the players' actions or as a result of context queries. In both cases sceneflow variables used for conditional branching are updated as described in the following section. The sceneflow is stored as an XML document and parsed by the sceneflow compiler which generates Java source code that can be executed by the sceneflow interpreter. After the compilation process both scenes and sceneflow are deployed as a single Java application.

3.3 Game Control

Each time the user places or removes a card, an event is generated and sent to the poker event handler. The same happens when the user chooses an action (bet, call, raise, or fold) by pressing the respective button on the graphical user interface (see Fig. 1).

The poker event handler receives these low-level events and updates the data model that represents the game state as well as the graphical user interface that visualizes it. It also analyses the situation and generates higher level events, e.g., that all cards have been removed from the table or that the user has changed the cards of a player in the drawing phase. At the end of this process, it updates the respective sceneflow variables, which may enable transitions in the sceneflow and trigger the next scenes.

Apart from scenes and scene groups the author can also attach commands to a node. These commands are executed by the sceneflow interpreter each time the node is visited. There are commands that modify the game state (e.g. selecting the next player after a scene has been played in which one of the players announces that he drops out) and commands that access the poker logic to suggest the next action (e.g. deciding which cards should be changed in the drawing phase and which action the virtual players should perform in the betting phase).

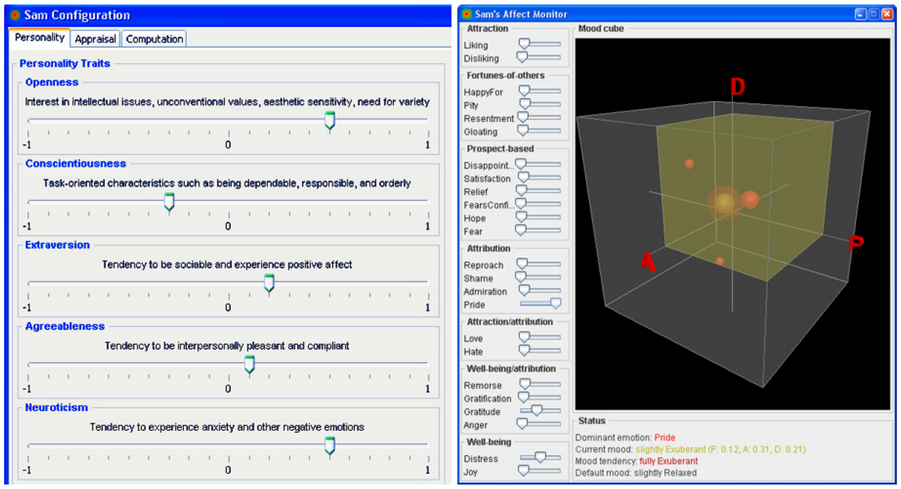
4 Modeling of Affect

4.1 Affect Computation

For the affect computation in real-time, we rely on ALMA, a computational model of affect [17]. It provides three affect types as they occur in human beings: *Emotions* (1) reflect short-term affect that decays after a short period of time. Emotions influence e.g. facial expressions, and conversational gestures. *Moods* (2) reflect medium-term affect, which is generally not related to a concrete event, action or object. Moods are longer lasting affective states, which have a great influence on humans' cognitive functions. *Personality* (3) reflects individual differences in mental characteristics.

ALMA implements the cognitive model of emotions developed by Ortony, Clores, and Collins (OCC model of emotions) [18] combined with the *BigFive* model of personality [19] and a simulation of mood based on the PAD model [20]. The emotion simulation considers the impact of mood and personality on emotion intensities. Personality traits can be configured for each character (see Fig. 5 - left side). Once configured, a *default mood* is computed.

ALMA enables the computation of 24 OCC emotions with *appraisal tags* [21] as input. Elicited emotions influence an individual's mood. The higher the intensity of an emotion's, the higher the particular mood change. A unique feature is that the current mood also influences the intensity of emotions. This simulates, for example, the intensity increase of *joy* and the intensity decrease of *distress*, when a individual is in an *exuberant* mood. The PAD model of mood distinguish eight different mood classes: *exuberant*, *bored*, *dependent*, *disdainful*, *relaxed*, *anxious*, *docile*, *hostile*. Generally, a mood is represented by a triple of the mood traits pleasure (P), arousal (A), and dominance (D). The mood's trait values define the mood class. If, for example, every trait value is positive (+P,+A,+D), the mood is *exuberant* (see Fig. 5 - right side, highlighted area). An AffectMonitor (see Fig. 5 - right side) visualizes in real time emotions and their intensities, as well as the current mood.



Openness
Interest in intellectual issues, unconventional values, aesthetic sensitivity, need for variety

Conscientiousness
Task-oriented characteristics such as being dependable, responsible, and orderly

Extraversion
Tendency to be sociable and experience positive affect

Agreeableness
Tendency to be interpersonally pleasant and compliant

Neuroticism
Tendency to experience anxiety and other negative emotions

Status
Dominant emotion: Pride
Current mood: slightly Exuberant (P: 0.12, A: 0.31, D: 0.21)
Mood tendency: fully Exuberant
Default mood: slightly Relaxed

Fig. 5. ALMA Personality Configuration and Affect Monitor dialogs

Mood and Emotion Control Affective Behavior Aspects According to the current mood and emotions elicited during the game play, the following aspects of the virtual poker characters' affective behavior are changed:

– *Breathing* is related to the mood's arousal and pleasure values. For positive values, a character shows fast distinct breathing. The breathing is slow and faint for negative values.
– *Speech quality* is related to the current mood or an elicited emotion. In *relaxed* mood, a character's speech quality is *neutral*. In *hostile* or *disdainful* mood or during negative emotions, it is *aggressive*, in *exuberant* mood or during positive emotions it is *cheerful*. In any other mood, the speech quality is *depressed*. Section 5 describes how these speech qualities are realised.

Initially, Sam's and Max's behavior is nearly identical, because they have the same default *relaxed* mood that is defined by their individual personality. However, Sam is slightly more *extroverted*, so his mood tends to become more *exuberant*. Max, on the other side, has a tendency to be *hostile*. This is caused by his negative *agreeableness* personality definition. This disposition drives Max towards negative mood (e.g. *hostile* or *disdainful*) and negative emotions faster than Sam, which supports the mean, terminator-like character of Max.

4.2 Authoring Affect

In the poker game, events and actions of the human player strongly influence the selection of scenes and their execution (see Section 3). As a consequence the actions of Sam and Max as well as their appraisal of the situation changes. For example, if Sam loses the game because he has bad cards, a related scene will be selected in which he verbally complains about that. An appraisal tag used in the scene lets Sam appraise the situation as a bad event during the scene execution. The corresponding appraisal tag is [**BadEvent**]. When this is used as input for the affect computation, it will elicit the emotion *Distress*, and if such events occur often, they will lead to a *disdainful* or *hostile* mood. In general, appraisal tags can be seen as a comfortable method for dialog authors to define on an abstract level input for a affect computation module. For doing this, no conceptual knowledge about emotions or mood is needed.

Affect in the Poker Game The poker game scenario covers the simulation of all 24 OCC emotions. The following overview gives a brief explanation how some of them are computed based on appraisal tags. They are grouped according to the OCC classification of emotions.

– *Prospect-based emotions*: *hope*, *satisfaction*, or *disappointment*. During the card change phases, Sam and Max randomly utter their expectation of getting good new cards. Those utterances also contain the appraisal tag [**GoodLikelyFutureEvent**]. As a consequence *hope* is elicited. After all cards have been dealt out, Sam's and Max's poker engine compares the current cards with the previous ones. Depending on the result either the [**EventConfirmed**] or the [**EventDisconfirmed**] appraisal tag is passed to the affect computation. In the first case (cards are better), *satisfaction* is elicited, otherwise *disappointment*.
– *Fortunes-of-Others*: *gloating* or pity. If Sam loses the game, Max (who does not like Sam) always comments the situation with a smirk: “With such cards you could never win!”. The appraisal tag [**BadEventForBadOther**] elicits *gloating* during the utterance. Sam always expresses his sorrow if a user loses a game (Sam likes users). During that, *pity* is elicited by the appraisal tag [**BadEventForGoodOther**].

The influence of emotions and mood on the behavior (speech, breathing) can be observed during the game play and gives some hints on the current situation of Sam and Max.

5 Expressive Synthetic Voices with Reliable Quality

Convincing speech generation is a necessary precondition for an ECA to be believable. That is true especially for a system featuring emotional expressivity. To address this issue, we have investigated the two currently most influential state-of-the-art speech synthesis technologies: unit selection synthesis and statistical-parametric synthesis.

Two criteria are traditionally used for assessing the quality of synthetic speech: intelligibility and naturalness. In the context of this paper, these criteria must be rephrased in line with the aim of supporting the user in *suspending his disbelief*. We address intelligibility in terms of a *reliable quality* of the generated speech output, and naturalness in terms of a *natural expressivity*.

Unfortunately, in state-of-the-art speech synthesis technology, the two criteria are difficult to reconcile. The most natural-sounding synthesis technology, unit selection [22], can reach close-to-human naturalness in limited domains, such as speaking clocks or weather forecasts [23], but it suffers from unpredictable quality in unrestricted domains – when suitable units are not available, the quality can drop dramatically. On the other hand, statistical-parametric synthesis, based on Hidden Markov Models (HMMs) [24], has a very stable synthesis quality, but the naturalness is limited because of the excessive smoothing involved in training the statistical models. Insofar, there is currently no technology that simply fulfils both the reliability and the naturalness criterion.

In IDEAS4Games, we investigated two methods for approximating the criteria formulated above. For our humanoid character, Sam, we created a custom unit selection voice with a "cool" speaking style, featuring high quality in the poker domain and some emotional expressivity; for our robot-like character, Max, we used an HMM-based voice, and applied audio effects to modify the sound to some extent in order to express emotions.

5.1 Expressive domain-oriented unit selection

Sam's voice was carefully designed as a domain-oriented unit selection voice [25]. Domain-oriented voices sound highly natural within a given domain; they can also speak arbitrary text, but the quality outside the domain will be seriously reduced. We designed a recording script for the synthesis voice, consisting of a generic and a domain-specific part. We used a small set of 400 sentences selected from the German Wikipedia to cover the most important diphones in German [26]. This is a small number – usually, between 1,500 and 10,000 sentences are recorded for a synthesis voice of standard quality. Consequently, it is clear that the general-domain performance of the voice must be expected to be quite restrained. In addition, about 200 sentences from the poker domain were recorded, i.e. sentences related to poker cards, dealing, betting, etc. – sentences of the kind that are needed for the game scenario. The 600 sentences of the recording script were produced by a professional actor in a recording studio. The speaker was instructed to utter both kinds of sentences in the same "cool" tone of voice.

In a similar way, the same speaker produced domain-oriented voice databases consisting of approximately 600 sentences each for a *cheerful*, an *aggressive* and a *depressed* voice. We built a separate unit selection synthesis voice for each of the four databases, using the open-source voice import toolkit of the MARY text-to-speech synthesis platform [27].

To generate speech from text, the MARY TTS system converts any given text into a linguistic *target*, an abstract description of the ideal units needed to generate the speech. The unit selection component identifies an optimal sequence of *units*, i.e. short audio-snippets from a given voice database, that best fit to the given target as well as to their respective neighboring units, and concatenates the selected units like pearls on a string, with only minimal signal processing involved. If – and only if – suitable candidates are found, the resulting speech will be of high quality. Naturally, the speech output will have the same speaking style as the original recordings. Changing expressive style, in this method, can only be done by switching from one voice database to another.

In our application, most of Sam's utterances are spoken with the neutral poker voice. As the voice database contains many suitable units from the domain-oriented part of the recordings, the poker sentences generally sound highly natural. Their expressivity corresponds to the "cool" speaking style realised by our actor. Selected utterances are realised using the *cheerful*, *aggressive*, and *depressed* voices. For example, when Sam loses a round of poker, he may utter a frustrated remark in the *depressed* voice; if the affect model predicts a positive mood, he may greet a new user in a *cheerful* voice, etc. Note that, as for the neutral poker voice, these emotional voices can potentially speak any text; however, only within a relatively small domain of poker-related emotional utterances, the quality will be optimal.

5.2 Robust HMM-based synthesis

The voice of the robotic character, Max, uses statistical-parametric synthesis based on Hidden Markov Models. When the voice is built, the statistical models are trained on a speech database. After training, the original data is not needed anymore; at runtime, speech is generated from the statistical models by means of a vocoder. As a result of the training process, the speech generated with an HMM-based voice has a certain degree of resemblance to the speaker and speaking style of the database used for training; however, the speech sounds muffled due to the averaging in the statistical models as well as the vocoding. While their speech is less natural, HMM-based synthesis voices have one great advantage, however: the quality is largely stable, independently of the text spoken.

We started from the open-source system HTS released by the Nagoya Institute of Technology [28], ported the code to Java, and incorporated it into our MARY TTS platform [29]. The HMM-based voice for our robotic character Max was created from recordings of a male speaker in the BITS speech synthesis corpus [30], a phonetically balanced German speech corpus containing about 1700 sentences per speaker. The resulting voice sounds rather monotonous, but intelligible and of stable quality throughout all utterances.

Expressivity for Max' voice is performed differently from Sam's voice. Here, we use audio effects to modify the generated speech. At the level of the parametric input to the vocoder, we can modify the pitch level, pitch range and speaking rate. In addition, we apply audio signal processing algorithms that modify the generated speech signal, using linear predictive coding (LPC) techniques. In this framework, our code provides a vocal tract scaler, which can simulate a longer or shorter vocal tract; a whisper component, adding whisper to the voice; a robot effect, etc. We can control the degree to which these effects are applied. Through a trial-and-error procedure, we determined coarse settings for expressive speech: *aggressive* speaking style is simulated by a lengthened vocal tract, lowered pitch, increased pitch range, faster speech rate, and slightly whisperised speech; *cheerful* speech is simulated by a shortened vocal tract, higher pitch level and range, and faster speech rate; and *depressed* speech is simulated by a slightly lengthened vocal tract, lower pitch level and much narrower pitch range, and a slower speech rate. A lot could be optimised about the application of these effects, e.g. the extent to which they are applied could be linked to the emotional intensity in a gradual way. Here, we merely intended to illustrate that audio effects can be applied to shape an HMM-based voice towards a certain expression.

6 Discussion and Conclusion

We have presented an ECA-based computer game employing RFID-tagged poker cards as a novel interaction paradigm. It is built around the integration of three components: a powerful and easy-to-use multimodal dialog authoring tool, a sophisticated model of affect, and a state-of-the-art speech synthesizer.

In our authoring approach, the separation of dialog content and narrative structure makes it possible to modify these two aspects independently. The script can be edited using standard text processing software – no programming skills are required. When the structure and the script are compiled into Java code, they are checked for integrity. This approach makes it extremely easy to rapidly develop and fine-tune an interactive application.

The flexible dialog script is most effective when it is supported by an equally flexible high-quality speech synthesizer. We have shown that the state of the art is still limited in that respect. On the one hand, the HMM-based voice used for our character Max is appropriately flexible, so that any changes in the dialog script can easily be rendered with his voice, in a limited but reliable quality. On the other hand, we have created a very high quality unit selection voice for our character Sam, but the quality comes at the price of high effort and limited flexibility. Extending the domain, or adding other expressions, would require additional recordings. Also, with unit selection technology it is not yet possible to express degrees of a certain expression: Sam's voice is either cheerful or not, but it is not possible to make him sound "slightly cheerful", or to make him sound "increasingly aggressive" as the game unfolds. We are working on voice interpolation methods that should make this possible in the future [31, 32].

Our computational model of affect plays a major role in modeling the expressive behavior of the poker characters Sam and Max. This is supported by two aspects: 1) the use of appraisal tags, which simplifies the generation of affect so that authoring affect does not require any deep knowledge about a model of emotion or mood; and 2) the real-time computation of different affect types as they occur in humans. Especially the continuous computation of short-term emotions and medium-term moods allow for a smooth blending of different aspects of affective behavior that help to increase the believability and the expressiveness of virtual characters.

Overall, we have shown how state-of-the-art research approaches can be combined in an interactive poker game to show new possibilities for the modeling of expressive virtual characters and for future computer game development.

Acknowledgments. The work reported here was supported by the EU ProFIT project IDEAS4Games (EFRE program). Part of this research has been carried out within the framework of the Excellence Cluster Multimodal Computing and Interaction (MMCI), sponsored by the German Research Foundation (DFG). The authors would like to thank Michael Schneider for supporting us with his RFID-technology expertise. Also, we would like to thank Nicolas Schulz for supporting us with the Horde 3D visualisation engine and extensive support.

References

1. Loyall, A.B.: Believable Agents: Building Interactive Personalities. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. (1997)
2. Gratch, J., Rickel, J., André, E., Cassell, J., Petajan, E., Badler, N.I.: Creating interactive virtual humans: Some assembly required. IEEE Intelligent Systems **17** (2002) 54–63
3. Mateas, M., Stern, A.: Façade: An experiment in building a fully-realized interactive drama. In: Game Developers Conference, Game Design Track. (2003)
4. Swartout, W., Gratch, J., Hill, R., Hovy, E., Marsella, S., Rickel, J., Traum, D.: Toward virtual humans. AI Magazine **27** (2006) 96–108
5. Martin, J.C., Niewiadomski, R., Devillers, L., Buisine, S., Pelachaud, C.: Multimodal complex emotions: Gesture expressivity and blended facial expressions. International Journal of Humanoid Robotics, Special Edition "Achieving Human-Like Qualities in Interactive Virtual and Physical Humanoids" (2006)
6. Kipp, M., Neff, M., Kipp, K.H., Albrecht, I.: Toward natural gesture synthesis: Evaluating gesture units in a data-driven approach. In: Proceedings of the 7th International Conference on Intelligent Virtual Agents. LNAI 4722 (2007) 15–28
7. de Rosis, F., Pelachaud, C., Poggi, I., Carofiglio, V., de Carolis, B.: From greta's mind to her face: Modelling the dynamics of affective states in a conversational embodied agent. Int. Journal of Human Computer Studies **59** (2003) 81–118
8. Marsella, S., Gratch, J.: Ema: A computational model of appraisal dynamics. In: Agent Construction and Emotions. (2006)
9. Schröder, M.: Emotional speech synthesis: A review. In: Proceedings of Eurospeech 2001. Volume 1., Aalborg, Denmark (2001) 561–564
10. Prendinger, H., Saeyor, S., Ishizuk, M.: Mpml and scream: Scripting the bodies and minds of life-like characters. In: Life-like Characters – Tools, Affective Functions, and Applications. Springer (2004) 213–242
11. Bosser, A.G., Levieux, G., Sehaba, K., Bundia, A., Corruble, V., de Fondaumière, G., Gal, V., Natkin, S., Sabouret, N.: Dialogs taking into account experience, emotions and personality. In: 6th International Conference on Entertainment Computing. Volume 4740 of LNCS., Berlin Heidelberg, Springer (2007) 356–362
12. Gebhard, P., Kipp, M., Klesen, M., Rist, T.: Authoring scenes for adaptive, interactive performances. In: Proc. of the 2nd Int. Joint Conference on Autonomous Agents and Multi-Agent Systems, ACM (2003) 725–732
13. Wikipedia: Draw Poker, http://en.wikipedia.org/wiki/Draw_poker. (2008)

14. Schulz, M.: Horde3D – Next-Generation Graphics Engine. Horde 3D Team, http://www.nextgen-engine.net/home.html. (2006–2008)
15. Hill, E.: Jess in Action: Java Rule-Based Systems. Manning, Greenwich, CT, USA (2003)
16. Klesen, M., Kipp, M., Gebhard, P., Rist, T.: Staging exhibitions: methods and tools for modelling narrative structure to produce interactive performances with virtual actors. Virtual Reality **7** (2003) 17–29 ISSN:1359-4338.
17. Gebhard, P.: Alma - a layered model of affect. In: Proc. of the 4th Int. Joint Conference on Autonomous Agents and Multiagent Systems, ACM (2005) 29–36
18. Ortony, A., Clore, G.L., Collins, A.: The Cognitive Structure of Emotions. Cambridge University Press, Cambridge, MA (1988)
19. McCrae, R., John, O.: An introduction to the five-factor model and its applications. Journal of Personality **60** (1992) 175–215
20. Mehrabian, A.: Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. Current Psychology: Developmental, Learning, Personality, Social **14** (1996) 261–292
21. Gebhard, P., Kipp, K.H.: Are computer-generated emotions and moods plausible to humans? In: Proc. of the 6th International Conference on Intelligent Virtual Agents (IVA 2006), Marina Del Rey, California, Springer (2006) 343–356
22. Hunt, A., Black, A.W.: Unit selection in a concatenative speech synthesis system using a large speech database. In: Proceedings of ICASSP 96. Volume 1., Atlanta, Georgia (1996) 373–376
23. Black, A.W., Lenzo, K.A.: Limited domain synthesis. In: Proceedings of the 6th International Conference on Spoken Language Processing, Beijing, China (2000)
24. Yoshimura, T., Tokuda, K., Masuko, T., Kobayashi, T., Kitamura, T.: Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis. In: Proceedings of Eurospeech 1999, Budapest, Hungary (1999)
25. Schweitzer, A., Braunschweiler, N., Klankert, T., Möbius, B., Säuberlich, B.: Restricted unlimited domain synthesis. In: Proc. Eurospeech 2003, Geneva, Switzerland (2003)
26. Hunecke, A.: Optimal design of a speech database for unit selection synthesis. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany (2007)
27. Schröder, M., Hunecke, A.: Creating German unit selection voices for the MARY TTS platform from the BITS corpora. In: Proc. SSW6, Bonn, Germany (2007)
28. Zen, H., Nose, T., Yamagishi, J., Sako, S., Masuko, T., Black, A., Tokuda, K.: The HMM-based speech synthesis system version 2.0. In: Proc. of ISCA SSW6, Bonn, Germany (2007)
29. Charfuelan, M., Schröder, M., Türk, O., Pammi, S.C.: Open source HMM-based synthesisser for the MARY TTS platform. In: Proceedings of the 16th European Signal Processing Conference (EUSIPCO 2008). (2008) submitted.
30. Ellbogen, T., Schiel, F., Steffen, A.: The BITS speech synthesis corpus for German. In: Proc. 4th Conference on Language Resources and Evaluation (LREC), Lisbon, Portugal (2004) 2091–2094
31. Turk, O., Schröder, M., Bozkurt, B., Arslan, L.: Voice quality interpolation for emotional text-to-speech synthesis. In: Proc. Interspeech 2005, Lisbon, Portugal (2005) 797–800
32. Schröder, M.: Interpolating expressions in unit selection. In: Proc. 2nd International Conference on Affective Computing and Intelligent Interaction (ACII'2007), Lisbon, Portugal (2007)