UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING
SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Aleksandra Čereković

Realization of multimodal behaviour of embodied conversational agents in real-time

Ostvarivanje višemodalnog ponašanja utjelovljenih razgovornih agenata u stvarnom vremenu

DOCTORAL THESIS
DOKTORSKA DISERTACIJA

Zagreb, 2010.

Doctoral thesis is written at the Department of Telecommunications, Faculty of Electrical Engineering and Computing, University of Zagreb.

Mentor: Associate professor Igor S. Pandžić, Ph.D.

This thesis has 107 pages, 34 figures and 11 tables.

Thesis no.:

The dissertation evaluation committee:

1. Professor Maja Matijašević, Ph.D.,

   Faculty of Electrical Engineering and Computing, University of Zagreb
2. Associate professor Igor S. Pandžić, Ph.D.

   Faculty of Electrical Engineering and Computing, University of Zagreb
3. Professor Toyoaki Nishida, Ph.D.,

   Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University, Japan

The dissertation defense committee:

1. Professor Maja Matijašević, Ph.D.,

   Faculty of Electrical Engineering and Computing, University of Zagreb
2. Associate Professor Igor S. Pandžić, Ph.D.

   Faculty of Electrical Engineering and Computing, University of Zagreb
3. Professor Toyoaki Nishida, Ph.D.,

   Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University, Japan
4. Professor Ignac Lovrek, Ph.D.

   Faculty of Electrical Engineering and Computing, University of Zagreb
5. Assistant professor Mario Kušek, Ph.D.

   Faculty of Electrical Engineering and Computing, University of Zagreb

Date of dissertation defense: 08th July 2010

“Any sufficiently advanced technology is indistinguishable from magic.”
Arthur C. Clarke (1917 - 2008), "Profiles of the Future"

Acknowledgements

I would like to thank my thesis director, Professor Igor Sunday Pandžić, who gave me the opportunity to work as a research assistant at his laboratory and carry out this thesis. I would like to thank him for his support in exploring new research subjects, as well as opportunity to participate in international conferences and workshops, meet new researchers and share ideas with them. I particularly appreciate the opportunity to work on NGECA project with highly capable colleagues from Universities of Kyoto and Tokyo; Huang Hsuan-Hung, with whom I collaborated on building of the universal GECA platform and two experimental ECA systems, and who wrote many of the collaborative papers, Professors Toyoaki Nishida and Yukiko Nakano, who brought constructive and fresh ideas for our joint work and written papers, and students I worked with at eNTERFACE '08 workshop in Paris: Takuya Furukawa and Yuji Yamaoka. I also wish to thank Professor Nishida for the concern, interest and constructive comments provided during the writing of this thesis.

Moreover, I would like to thank all my colleagues for their collaboration, especially Goranka, for her support, and Tomislav, for his help with Visage Technologies software.

Finally, I express my deepest gratitude to my friends for the many fun moments we shared, and my family for all their love and continuous support. I also thank Goran, for his small, but notable role in finishing the thesis.

Zahvala

Zahvaljujem mom mentoru, profesoru Igoru S. Pandžiću, koji mi je dao priliku da radim kao znanstveni novak u njegovom istraživačkom laboratoriju i da ostvarim titulu doktora znanosti. Zahvaljujem mu na pomoći pri istraživanju i mogućnosti da putujem na međunarodne konferencije, upoznam znanstvenike i da s njima dijelim ideje. Naročito cijenim priliku da radim u NG-ECA projektu sa vrlo stručnim suradnicima sa Sveučilišta u Kyotu i Tokyou; Huangom Hsuan-Huanom, sa kojim sam radila na izgradnji univerzalne GECA platforme i dva eksperimentalna ECA sustava, i koji je napisao većinu zajedničkih članaka, profesorima Toyoaki Nishidi i Yukiko Nakano, kojima zahvaljujem na konstruktivnim idejama pri izvedbi zajedničkih projekata i komentarima, i studentima s kojima sam radila na radionici eNTERFACE ’08 u Parizu: Takuyi Furukawi i Yuju Yamaoki. Također, zahvaljujem profesoru Toyoaki Nishidi na brizi, interesu i konstruktivnim komentarima pri pregledavanju ove disertacije.

Zahvaljujem svim mojim kolegama na suradnji, posebno Goranki na podršci, i Tomislavu, za pomoć oko Visage sustava.

Konačno, veliko hvala prijateljima na zabavnim trenucima koje smo proveli skupa i mojoj obitelji na ljubavi i podršci. I Goranu, za malu, ali značajnu ulogu u završavanju ove disertacije.

Foreword: Kate, my housekeeper

It is the Future. I am entering my smart house. At the front entrance I am greeted by Kate, my tall, attractive housekeeper.

"Hello, Sam." - she smiles at me.

"Hello Kate." - I respond to her.

"Would you like to enter the house?"

"Yes."

I hear the sound of unlocking and the door opens. I enter the house, the doors close.

"Kate, it's a bit cold, what’s the temperature?"

"18 degrees, the way you wanted it to be when you left for work"

"Uh, it's pretty cold outside, please, can you heat it up to 22, 23?"

"Of course, Sam."

I leave my car-keys and wallet on the table. I open the fridge.

"Kate, can you order pizza?"

"What kind of pizza do you want me to order?"

"I don't know, the last one was good."

"Alright, I'll order it. It was four seasons from ‘Casa Italiana’."

I sit on the sofa, Kate turns on the TV. She knows what I like. This evening, NBA final games are starting: "Philadelphia Eagles - Los Angeles Stars".

At half-time, the door bell rings. I hear Kate speaking to someone.

"Sam, your pizza is here!"

I have to open the door for the pizza-man. Kate has paid for it via bank-account, but I have to take the pizza myself. After all, she is just an ECA.

Contents

List of Figures

List of Tables

1 Introduction

Computer-generated (CG) virtual humans who have ability to talk to humans in natural way are no longer just a concept. By the end of the 1990s those artificial constructs got their name - Embodied Conversational Agents (ECAs). Nowadays there is considerable ongoing work on improving ECAs in order to establish intuitive human-like interaction with computers. To bring ECAs closer to humanity, researchers are engaged in modelling emotions and personality, creating specifications for interfaces between various modules of ECA systems and studying effects of ECAs on our world.

This thesis is dedicated to the creation of ECAs. The great breadth and complexity of this topic forces us to focus on one part of ECA system, and that is the component which realizes multimodal behaviours for ECAs. In the thesis we devise a multimodal behaviour realization system we name RealActor. Another contribution of the thesis is a tour guide system featuring a tour guide ECA that communicates with two human users. It serves to investigate a multiuser support in ECA systems and it is developed in collaboration with colleagues from Japan.

1.1 Context

The personification of computer interfaces levels human-computer interaction to human-human interaction. While the idea has been around since the first appearance of computers, it is now being realized through Embodied Conversational Agents (ECAs). For years ECAs existed mainly in research laboratories, but recently there has been very encouraging interest beyond the academic world.

What are ECAs? Justine Cassell defines them as anthropomorphic interface agents that are able to engage a user in real-time, multimodal dialogue, using speech, gesture, gaze, posture, intonation and other verbal and nonverbal channels to emulate the experience of human face-to-face interaction [Cassell et al., 2000]. Terms "embodied" and "conversational" can be easily understood, while the term "agent" means that an embodied human-like character acts from its own motivations [Badler et al., 2000]. ECA is an autonomous character who has her/his beliefs and goals. These are expressed in conversation with a user in the form of verbal and nonverbal cues, the same ones that a human expresses while speaking. In addition to that, many of the existing ECAs are designed to show empathy and emotions during interaction and they try to establish a social relationship with the user.

The ideal ECAs would have unrestricted conversational abilities and wondrous knowledge. Present research evidences, however, that there is still a long way to go. The topic of ECAs is multidisciplinary and an ECA system is fairly complex to build. It requires fundamental advances in speech recognition, natural language understanding and generation, dialogue management, cognitive modelling and reasoning, virtual human architectures and computer graphics and animation. Unfortunately, some of these disciplines are not yet advanced enough to effectively achieve what is required.

For example, let us consider the verbal capabilities of ECAs. ECA engages a user in realistic conversation, in which ECA's responses and claims depend on the user's responses, on what has been previously said and on its own beliefs and desires. This requires an ongoing effective and interactive dialogue in which the user and ECA participate and which is influenced by cognitive and emotional models of ECA. There has been much work on dialogue modelling, natural language understanding and generation for ECAs [Traum, 2008]; however it is still a difficult problem. As a result, existing ECA systems are application-specific, e.g. they have roles of soldiers [Hill et al., 2003], patient assistants [Bickmore et al., 2007], tour guides [Huang et al., 2007] etc. To overcome dialogue drawbacks, settings in these specific applications constrain the relevant interaction and narrow ECA's conversational abilities.

Without any doubts we can conclude that, despite the efforts of the research community, current ECAs do not behave like real humans yet. In the years ahead it is up to researchers to discover the guidelines to bring these artificial constructs closer to human beings and make them an essential part of everyday life.

1.2 Motivation and Specific Objectives

The main objective of the thesis is to achieve a convincing behaviour of ECAs using an animation and behaviour realization system which is driven by descriptive behaviours of verbal and nonverbal cues of ECAs. Motivation for this work originates from an early encounter with these virtual characters.

It the summer of the year 2006 the author of this thesis participated in the one-month workshop on multimodal interfaces "eNTERFACE '06". The aim of this workshop was to gather researchers to develop proposed application projects and then present their research results. Relevant to this thesis, a joint Croatian – Japanese team investigated rapid development of ECAs and explored the issues that may occur in multicultural and multimodal communication with these constructs. As the workshop outcome we devised a tour guide system in which a tour guide ECA for the city of Dubrovnik converses in English, Japanese and Croatian [Huang et al., 2007]. Though we worked with great enthusiasm, the final system was not as good as we expected. Due to restrictions of the speech-recognition engine and simple dialogue model, the system could "understand" only several questions about the city. Moreover, our ECA looked like a wooden doll rather than a human. It was capable of no facial expressions, no head movements and could only perform a couple of hand gestures which were generated on the fly by events coming from the text-to-speech engine. It was obvious that one month was not enough to complete the predefined objectives.

Lack of realism in ECA displays is a well-known issue, evidenced in a state of the art paper which describes existing characters as "wooden" [Vinayagamoorthy et al., 2006]. Therefore, a group of researchers who study the topic of ECAs have proposed a descriptive behaviour language for ECAs, named Behavior Markup Language (BML) [Kopp et al., 2006][Vilhjalmsson et al., 2007]. BML was proposed with the purpose of providing a powerful description method for motion synthesis in the animation system and hence enrich ECAs' communicative utterances and the level of realism in their performance. Since the year 2006 when it was proposed, several BML workshops gathered researchers who worked on improving BML. The necessity of developing several BML-compliant multimodal behaviour realization systems was emphasized at these workshops. Requirements and behaviours of these realization systems were meant to provide guidance in further development of the BML specification. At the moment of the writing this thesis, BML has progressed and it turned out to be more powerful than any existing language with the same purpose. BML community keeps maintaining a website and active discussion lists. The activities in the BML community and the ECA community in general have served as guidelines to define the final objectives in this thesis.

One objective of this thesis was to develop a multimodal behaviour realization system, which we name RealActor, and which uses BML to drive ECAs behaviours. Since it is an insufficiently studied area, another aim was to develop an experimental system in which our ECA interacts with two users. The objective was to learn new lessons regarding development of ECA systems, experiment with various devices for human-machine interaction and analyze impressions of interacting with such a system.

RealActor has been tested and successfully used in two student projects. At the moment it has been released as an open-source system and it remains to be seen see how it will be accepted in the research community.

1.3 Approach

The ideal behaviour realization system should be one that exactly matches the displays on human body with all the variations and expressions resulting from human mood, emotions and reactions to the environment. The animated body should be interruptible since the system should perceive the environment and react to various events which are sometimes unpredictable. Realization and synchronization across modalities should be done in a way that preserves the naturalness of human movements.

In the next chapters we will show that existing realization systems are far from ideal. Current systems suffer from a lack of believability; they are constrained by capabilities of graphics platforms, animation systems and speech synthesis engines.

Our research proposes a behaviour realization system which encompasses real-time behaviour generation, synchronization across modalities including on the fly speech-synthesized events and responses to events from the environment. The system is carefully developed according to the main requirements of ECA systems:

- **Real-time multimodal behaviour execution** - realization of complex communicative utterances, which include verbal and nonverbal behaviour, in real-time.
- **Animation Control** - intuitive high-level mechanisms for character motion control
- **Extensibility** with large number of animations. The feature is especially important for culturally codified ECAs.
- **Applicability** - applicable to a wide range of domains, such as ECAs involved in marketing, virtual tutors, advisors, as well as communicative non-player characters (NPCs) in role-playing games
- **Integrability** - multi-platform, modular and easy to integrate with existing engines and application frameworks
- **Availability** - open-source and publicly available for the benefit of the research community

Within the system we employ a novel solution to synchronize animations with synthesized speech and the dynamic face animation model based upon Facial Action Coding System (FACS) [Ekman and Friesen, 1978]. To build the face model we have chosen Ekman's system because it is widely accepted in psychology and computer science and can be used to synthesize an effectively infinite number of facial expressions.

Regarding multiuser ECA systems, existing papers are mostly centred on the study of behaviours of humans interacting with ECAs - for example, the study on gazing [Rehm et al., 2005][Vertegaal et al., 2001][Rehm and Andre, 2005]. On the other hand, we are more interested in developing support for multiuser communication in order to overcome all the problems presented in the theory of multiparty communication [Traum, 2003].

1.4 Organization of the Thesis

The thesis is composed of several chapters which are divided into sections.

Chapter 2 reviews the state of the art. First, it gives an overview of existing systems with Embodied Conversational Agents and studies on multiuser support in these systems. Second, it presents existing multimodal behaviour realization systems and discusses requirements and issues in the area.

Chapter 3 introduces RealActor, the multimodal behaviour realization system which is the main contribution of the thesis. It presents key challenges of multimodal behaviour realization and the approaches we have used to overcome them. The last section of Chapter 3 gives in-depth overview of the system: it explains the system's design, behaviour realization process and the neural network model which synchronizes synthesized speech with gestures.

Chapter 4 is dedicated to the face animation model employed by the system. It explains Facial Action Coding System (FACS) [Ekman and Friesen, 1978] and the approach we used to model animations based on the FACS. It also explains subjective evaluation conducted with human participants.

Chapter 5 describes the multiuser Tour guide system. It discusses key challenges of integrating multiuser support and initial experiments with the equipment. Finally, it explains the overall architecture and discusses experiments in interaction with human users.

Chapter 6 proposes a conclusion and summary of the contributions.

In the end we give several appendices, notably a list of symbols used in this document and a manual for RealActor with several examples of BML scripts that can be used to synthesize ECA behaviours.

2 Background

In this chapter we present an overview of research related to development of conversational virtual humans - Embodied Conversational Agents (ECAs). Special attention is given to multimodal behaviour realization and related issues.

In interaction with humans ECAs express themselves using verbal and nonverbal cues: gestures, speech, facial expressions and body motions. Behaviour realization system displays these cues and coordinates verbal and physical representations of actions. We are going to show that expressivity of behaviours is affected by models of emotions and personality which are increasingly integrated into ECA systems. Expressivity is displayed through space and dynamics of hand gesturing, intensity of facial displays, head movements... These are just some of the requirements for behaviour realization that will be discussed in this chapter.

The chapter is organized as follows. Section 2.1 begins with an overview of requirements of ECA systems. We further present examples of existing systems and explain how they work. We also discuss state of the art in multiuser support in ECA systems because it is one of the contributions of the thesis. In Section 2.2 we discuss the requirements of multimodal behaviour realization systems. We give an overview of Behavior Markup Language (BML), the multimodal behaviour representation language used in this thesis. Further, we present animation techniques for interactive virtual humans. Finally, we present existing multimodal behaviour realization systems for ECAs and conclude the chapter with a discussion.

2.1 Systems with Embodied Conversational Agents

One of the first ECAs brought to life was Gandalf, which appeared in 1996 [Thórisson, 1997]. Gandalf is a 2D cartoonish character who explains to users a wall-sized graphical projection of the solar system using speech and gestures. In the late

1990s several research laboratories explored the issues of building virtual characters, such as the issue of integrating external control into autonomous agent architectures [Blumberg, 1997], achieving animated characters by adding rhythmic and stochastic noise functions to motion synthesis process [Perlin, 1995], creating personality for synthetic agents with prescribed roles [Hayes-Roth et al.,1996]. Today, a large number of researchers are involved in creation of ECAs, and some of them have even devised systems for real-world applications [Bickmore et al., 2007] [Traum et al., 2004] [Kopp et al., 2006]. Also, there has been very encouraging interest for this topic beyond the academic world, for example in the game industry. Involvement of autonomous agents in game scenarios has the potential to ease the work of animators who are developing conversational characters and reduce the process of creating dialogue systems.

In this section we explain the characteristics and requirements of ECA systems. We introduce a pioneer example of Real Estate Agent REA [Cassell et al., 2000] and compare it to examples of present-day systems, commenting on progress in the field. The section will be concluded with a review of multiuser ECA systems and discussion.

2.1.1 Requirements of ECA systems

Existing ECA systems vary in their verbal capabilities, input modalities and ECA roles, but they all engage a human user in natural conversation. During interaction ECAs reason about the current situation and choose appropriate response delivered in verbal output that is synchronized with realistic nonverbal animations. Researchers expect these characters to be [Thibeaux et al., 2008]:

- Responsive – must respond to the human user as well as other unexpected events in the environment.
- Believable – must provide a sufficient illusion of life-like behaviour that the human user will be drawn into the social scenario.

- Interpretable – the user must be able to interpret ECAs' response to situations, including their dynamic cognitive and emotional state, using the same verbal and nonverbal behaviours that people use to understand one another

To compose a non-trivial ECA, many assemblies or components are required. According to their functionalities in the interaction information flow, we group those components into four different categories (Figure 1) [Huang et al., 2008]:

- Assemblies in the input phase
- Assemblies in the deliberate phase
- Assemblies in the output phase
- Platform for Integrating ECA components

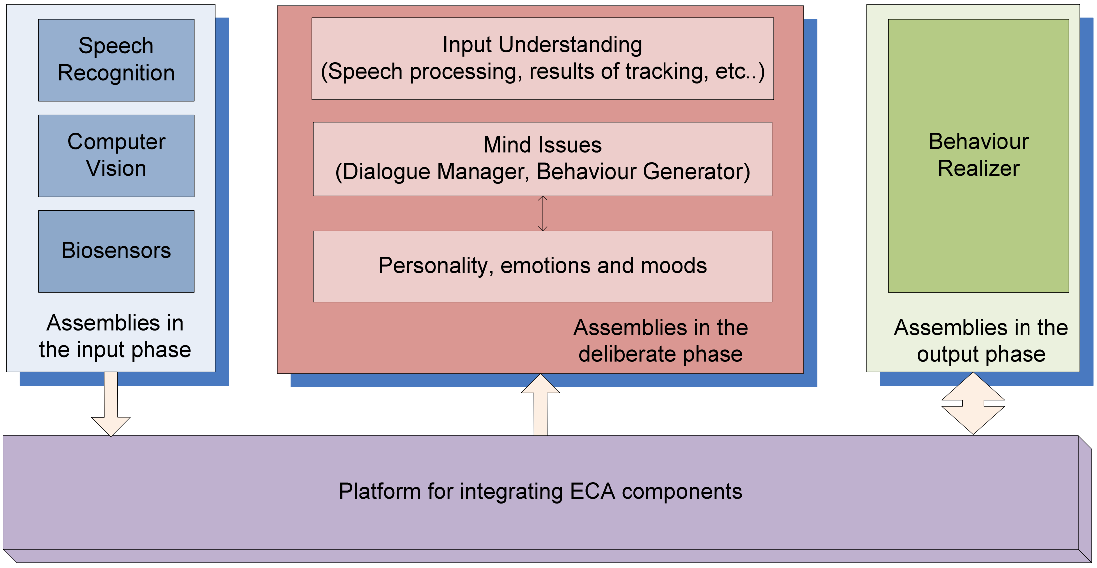

Speech Recognition
Computer Vision
Biosensors
Assemblies in the input phase
Input Understanding (Speech processing, results of tracking, etc..)
Mind Issues (Dialogue Manager, Behaviour Generator)
Personality, emotions and moods
Assemblies in the deliberate phase
Behaviour Realizer
Assemblies in the output phase
Platform for integrating ECA components

Figure 1: A simplified scheme of ECA system

During interaction with an ECA, input modalities collect data from the environment continuously. The inner part of ECA system, which consists of assemblies in the deliberate phase, processes and combines this information to define the user utterance and events from the environment. The selected multimodal behaviours for an ECA are then generated by output assemblies which display ECA's verbal and nonverbal output. Mediation of data between all components is done by the platform or server.

2.1.1.1 Assemblies in the input phase

In natural face-to-face communication nonverbal behaviours are an indispensable counterpart of verbal information. For example, the pronoun “this” without a pointing gesture or matching context becomes ambiguous. The same words spoken in different voice tones accompanied with different facial expressions can be interpreted into totally different meanings. Therefore, embodied agents have to possess capabilities to acquire various types of input from human users. To capture both verbal and nonverbal behaviours of users we propose the following modalities:

- *Verbal cues* are detected with speech recognition algorithms
- *Head movements* such as nods or head shakes can be detected by a head tracker or other sensing devices like an acceleration / motion sensor
- *Gaze direction* can be detected by eye tracking devices
- *Body postures and hand gestures* can be captured by using magnetic and optical motion capture devices or computer vision methods
- *Finger movements* can be acquired by data gloves
- *Emotion recognition* can be done by estimating facial expressions extracted from visual data, by performing acoustic analysis or by employing bio-sensors that measure heart rate, blood pressure, etc.

The most common way in which the existing ECA systems communicate with humans is using only plain text or speech. Though speech-based interfaces are essential for establishing conversation at the natural pace, communications abilities of present ECAs are far from natural. The problem is not merely one of limitations of today's speech recognition algorithms. It also concerns spoken dialogue and natural language understanding which are difficult problems [Traum, 2008]. Therefore, settings in specific applications constrain the relevant interaction and narrow ECA’s conversational abilities. Under these circumstances speech recognition components are adapted to detect keywords defined in dialogue. Another solution to this issue are dialogue trees (e.g. [Ljunglöf, 2009]) that prompt the user to choose among proposed actions using a graphical interface. Trees however fail in the mission to provide natural interaction with humans.

To capture and process user utterances, researchers are also experimenting with inexpensive equipment which provides usable results and is easily portable. Examples are cameras that track the user's gaze, which has a very important role in regulating the conversation flow [Brandherm et al., 2008] [Kipp et al., 2008]. Despite these studies, the usage of nonverbal inputs in ECA systems is merely in an experimental phase.

2.1.1.2 Assemblies in the deliberate phase

The main task of the assemblies in the deliberate phase is to process information that is collected from the environment and decide the behaviour of an agent based on the processing results. This part of the system represents the brain of ECA, which is responsible for reasoning, decision-making, establishing relationships and experiencing emotions in interaction with human users or other ECAs.

The following functional components are at work in the deliberate phase:

- *Input understanding* from verbal and nonverbal modalities. These components are responsible for processing data collected from the environment and for resolving the user's utterance as well as other events. Examples are components which process natural language and those which process body motions, e.g. gestures and gaze direction. Information obtained from such processing is combined into a single semantic representation of the user's utterance.
- *Mind issues*: a non-trivial ECA should have its own conversational goal to achieve. An example is an ECA which sells an airplane ticket or guides a human visitor to the place where he wants to go. Therefore, an inference engine with a background knowledge base and dialogue manager are required for conducting a discourse plan to achieve the agent's conversational goal. The plan should be conducted according to the agent's background knowledge, user input and its own internal mind state.
- *Personalization*: talking to a conversational agent without emotion and facial expression feels uncanny. Personality, emotion, culture, and social role models should be incorporated into an ECA system to improve its believability.

The elicitation of information from a user, goal setting, tracking progress of communication, and tracking ECA's moods and emotions require ongoing interactive dialogue. The dialogue system chooses responses which are appropriate to the current situation and all that has been communicated before. In the past dialogue systems conducted simple pattern-matching techniques (e.g. AIML language [Alice]), but this structure is not sufficient for building agents that are able to show reactive behaviour during communication. For this reason the systems that use finite state machines were created [Churcher et al, 1997]. In a further step, affective computing was involved in the process of decision-making through simulation of effects that personality and emotion have on human beings. Examples of this synergy are dialogue systems which are being integrated with agent-oriented models, such as the Belief-Desire-Intention (BDI) model [Bratman, 1987]. In the BDI model the basic principles of decision making are synthesized by modelling dynamic phenomena which incorporate emotions, personality, culture, and social relationships to reason about problems and select plans from a plan library. Since the BDI approach allows specification of large plans for complex domains, it has been implemented and applied to a number of ECA systems. One example is agent Max, a virtual tour guide agent [Becker et al., 2004] who uses 'Pleasure, Arousal and Dominance' (PAD) mood space [Mehrabian, 1995] and 9 different emotions. In addition to the BDI model, researchers are paying attention to models of emotion activation [Ortony et al., 1998] and connecting them to the reactive component of the dialogue planner. One example is a complex model of emotions based on personality, social context, time, events and reasoning which is integrated inside the Greta agent [Ochs et al., 2005].

For the purpose of reactive decision-making, different models of emotions and personality based on psychological findings have been proposed. In general, in theory of emotions emotional state is viewed as a set of dimensions. In the most popular theory there are six basic emotions: happiness, surprise, sadness, disgust, fear and anger, which are universally recognized across ages, cultures and species [Ekman and Rosenberg, 1997]. Another model does not reduce emotions to a finite set, but attempts to find underlying dimensions into which emotions can be decomposed. An example is the aforementioned 'Pleasure, Arousal and Dominance'

(PAD) mood space which describes emotion as a vector in three-dimensional vector space. Pleasure stands for the degree of pleasantness of the emotional experience, Arousal stands for the level of activation of the emotion, and Dominance describes the level of attention or rejection of the emotion. Third approach of modelling emotions is appraisal approach, which finds that emotions are elicited by events in the environment which are judged as good or bad for us. An example is the OCC-Model [Ortony et al., 1988], in which the subjective interpretation of emotion is evaluated with respect to goals, intentions, norms or standards, taste and attitudes of a character.

The study of personality submits that personality is composed of personality traits which are relatively stable dispositions that give rise to characteristic patterns of behaviour. The most popular model of personality is the psychological OCEAN or Five-Factor-Model [McCrae and John, 1992]. The model organizes personality traits into the five basic dimensions: conscientiousness, extraversion, openness, agreeableness and neuroticism. Dependencies between emotions and the OCEAN-model for personality are often implemented with Bayesian belief networks or fuzzy logic [Arellano et al., 2008].

2.1.1.3 Assemblies in the output phase

The output phase assumes a graphical interface (e.g. monitor, PDA) which displays the ECA's body and speakers which reproduce the ECA's voice. Realization of character behaviours is done by a 3D character animation player - *behaviour realizer* that renders an ECA and its surrounding virtual environment. Realization of multimodal behaviours necessitates coordination of verbal and physical representations of actions. Usually, these representations are described with text or XML-based representation languages, such as Behavior Markup Language (BML) [Kopp et al., 2006]. During realization an ECA uses two channels: verbal and nonverbal channel.

Verbal output for virtual humans is generated in two ways. The first approach is based on text-to-speech (TTS) systems, which convert normal language text into speech. During speech-synthesis process TTS systems output phonetic transcriptions which are then rendered as lip movements of the virtual character. To increase the believability and naturalness of ECA's voice, it is desirable that the TTS parameters, such as prosody properties, speed and volume, can be adjusted to represent specific voice expressions. While in the past TTS systems produced rather synthetic sound, today there are such systems which modulate voice using a set of emotional parameters [Schröder, and Trouvain, 2003]. This approach is often used in existing ECA systems because it allows freedom to change the application scenario and add new behaviours for ECAs without development efforts in the system output.

The second approach to produce speech is based on so-called lip-syncing, which is a technical term for matching lip movements to voice. In lip-sync systems mapping is done by processing an audio signal in order to get the visual parameters which control displacement of the lips. In this area much work has been done (e.g. [Zorić and Pandžić, 2006] [Annosoft]), since applications of such systems range from production of film, video and television programs to computer and video games. Lip-sync is less often used for ECAs due to requirements on their autonomy. ECAs should have great voice capabilities and for complex scenarios it is necessary to record hundreds of audio files. Consequently, in case of change or upgrade of application scenario, it is necessary to re-hire the same actors to capture new files. The lip-sync approach is typically used for languages that do not yet have their own TTS engines. For example we have utilized our lip-sync system [Zorić and Pandžić, 2006] in the Croatian version of our culture-adaptive tour guide agent [Huang et al., 2007].

Researchers stress the importance of generating *nonverbal outputs* (e.g. facial expressions, eye blinks, hand gestures and body vibrations and postures) is because they serve not only for conveying information (redundantly or complementarily with respect to the speech channel), but also for regulating the conversation flow [Bickmore and Cassell, 2001]. Nonverbal displays are also crucial for establishing relationships, since they provide social cues such as attraction. In character animation systems nonverbal outputs are generated using various animation techniques. Timing of these behaviours is defined with aforementioned representation languages. In real-time phase realization systems usually use smart controllers to schedule execution and trigger animations depending on the results of TTS speech synthesis and execution of other animations. Applied animations vary in expressivity of execution and reflect ECA's inner state and experienced emotions.

2.1.1.4 Platform for Integrating ECA Components

A critical part of an ECA system is the platform which seamlessly integrates all the various assemblies described above. This platform relays all sensor data streams, decisions and command messages between the components. There are several requirements for this platform. First, since ECA systems operate in real time, it is essential to establish real-time communication between components. Second, the platform should support transmission of declarative language which describes behaviour of users, the agent and states of the components. Third, the multithreading control and "push/pull" mechanism are a necessary condition for communication between multiple shared objects. This means that within the platform the blackboard, which stores and transmits messages, should be implemented. Fourth, since the ECA components usually take a lot of memory resources, it is desirable that such platform operates in a distributed environment.

Due to lack of some of those features, previous integration solutions were not appropriate for this purpose. Examples are traditional remote method invocation (RMI) APIs, distributed object/service models like CORBA [Object Management Group CORBA] and Web Services [Web Services Activity], which provide only "pull" direction information flows. Another example are solutions that do not support explicit temporal model and real-time applications like KQML [KQML]. This is a reason why the research community MindMakers [MindMakers] proposed OpenAIR specification protocol [OpenAir]. OpenAIR is a specification of an XML-based message format for low-level routing and message passing between stand-alone software modules using a subscribe-publication mechanism. OpenAIR specifies the characteristic features like timestamp, priority fields and a way to transmit binary data streams. It is simple enough to be considered suitable and effective for real-time interactive AI applications. OpenAIR has so far been used as a communications protocol in several ECA and robot systems [Huang et al., 2007] [Metta et al., 2006] or for communication between modules which constitute one component of an ECA system [Jonsdottir et al., 2008].

2.1.2 Situation, Agent, Intention, Belief, Animation Framework

The various assemblies presented in the previous section are generalized in the Situation, Agent, Intention, Behavior and Animation (SAIBA) framework [Kopp et al., 2006]. It is a framework which specifies the way of constructing ECA behaviours from intent planning to behaviour realization (Figure 1); it attempts to specify rather abstract modules which operate in the deliberate and output phase of the ECA system.

SAIBA appeared as a result of trends from the beginning of 2000s. At the time a number of ECA behaviour production processes have been presented in a variety of ECA systems. To avoid replication of work, as well as to allow sharing of modules between various research laboratories, a need to set a common specification of ECA behaviour production emerged. In the year 2005 prominent researchers who work on the subject of ECAs - linguists, psychologists, and researchers in computer graphics and animation – advanced the idea to unify integration and development of multimodal generation skills for virtual humans. This aim resulted in SAIBA framework. In SAIBA ECA behaviour production consists of three stages (Figure 2):

1. planning of communicative intent,
2. planning of multimodal realization of this intent, and
3. realization of the planned behaviours

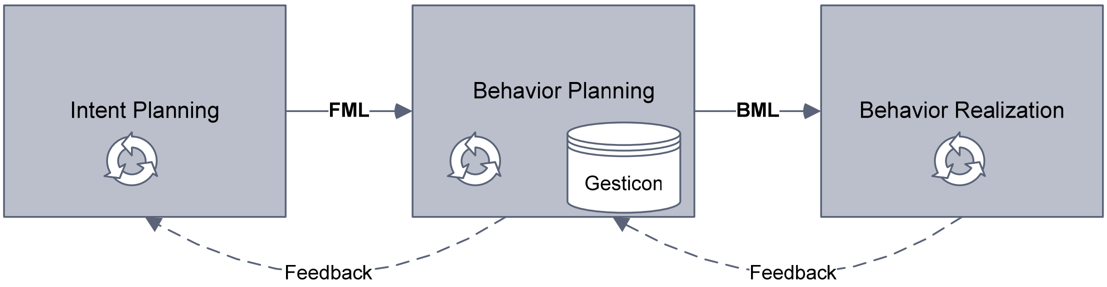


Figure 2: Situation, Agent, Intention, Belief, Animation (SAIBA) Framework

SAIBA developers observe these stages mainly as "black boxes" which are in principle involved in the generation of individual multimodal behaviours. The stages are inter-linked and provide each other with execution feedback and information at separate levels of abstraction. What interests developers is the information passed between the stages, because clear definition of the information allows for a modular architecture and sharing of components without much integration effort. Therefore, they specify two representation languages which serve as clear interfaces between the stages.

The first language, Function Markup Language (FML) [Heylen et al, 2008], is employed for communication between Intent Planning and Behavior Planning stages. This language represents communicative and expressive intent without specifying physical realization of behaviours; it describes agents' goals, plans and beliefs as communicative functions. At the time of writing of this thesis there were several ongoing discussions about Function Markup Language that were insufficient to offer an exact specification.

The second language is Behavior Markup Language (BML) [Kopp et al., 2006] [Vilhjalmsson et al., 2007]. BML provides a general description of behaviours that can be used to control an ECA. The specification provides details in describing behaviour and the relative timing of actions. Since BML serves as input to the multimodal behaviour realization system, it will be explained in the next section in which we introduce the multimodal behaviour realization process, along with technical challenges and existing solutions in that area.

2.1.3 Example Systems

The functionality and examples of ECA system components are explained on examples of the actual systems: a pioneer system, Real Estate Agent, by Justine Casselll [Casselll et al., 1999], and two systems that are still under development - Greta from University of Paris 8 [Greta Documentation] and virtual patients developed by University of Southern California and Institute for Creative Technologies [Kenny et al., 2008] [Parsons et al., 2008]. Greta system is based on SAIBA framework.

2.1.3.1 Real Estate Agent (REA)

One of the prominent examples of early ECAs, Rea, was developed at the end of 1990s at MIT Media Lab. Rea is a real estate salesperson who interacts with human users, shows them around virtual houses and tries to make a sale. The features of human-human conversation being implemented in Rea are both task-oriented and socially-oriented. During interaction the system synthesizes her speech and generates gestures based on a grammar, lexicon of gestures and communicative context.

The Rea system possesses a conversational model which is developed following several conversational rules. The system distinguishes between two types of discourse information which represent contribution to an ongoing conversation: interactional and propositional information. The interactional discourse functions are responsible for creating and maintaining a communication channel, while propositional functions correspond to the content of the conversation and shape the actual content [Casselll et al., 1999]. Both functions may be expressed through verbal and nonverbal channels: a gesture which indicates that the house in question is five minutes on foot from a specific point is interactional, while head nodding which means that Rea understood the user's speech is propositional information. During interaction the system makes a distinction between propositional and interactional discourse contributions. In the discourse model, conversational elements are described as functions rather than behaviours ("turn-taking", "feedback", "initiation and termination"). Based on the functions and contribution to the conversation, the system generates Rea's behaviours which occur in temporal synchrony with one another. Rea's gaze, raised eyebrows and head nods are used to synchronize turn-taking and generate smooth conversation with the user. In the system, multithreading is used to watch for feedback and turn requests.

The constructed system is depicted in Figure 3.

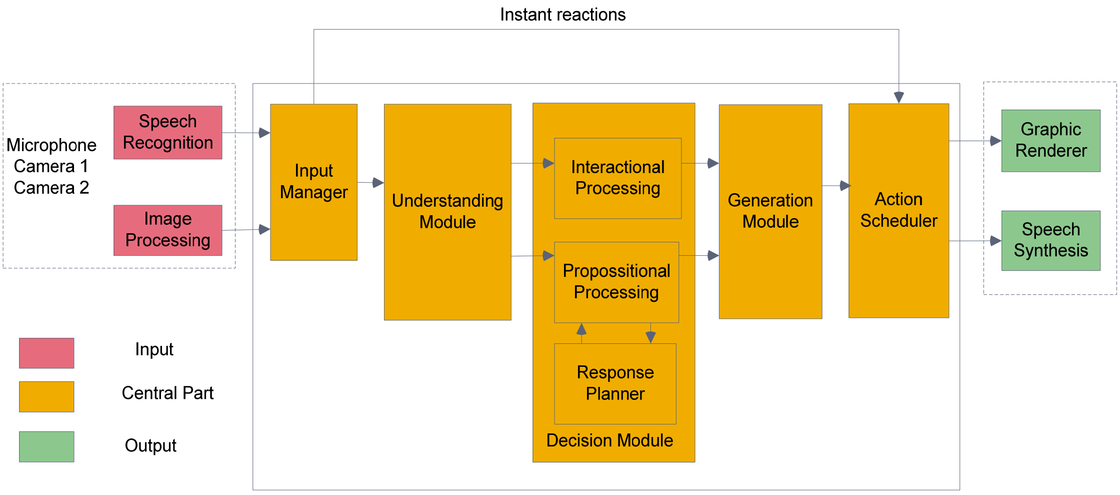


Figure 3: Architecture of the Rea system

Figure 4: A snapshot of Rea’s virtual reality

Input assemblies are two cameras and a microphone which capture the user's nonverbal behaviours and speech, respectively. Two cameras are mounted on top of the ceiling and they track the user’s head and hands. Developers introduce an input manager which collects input from these components and decides whether the system should instantly react or go through the decision process. Instantaneous actions, such as arrival of the user, are handled by a hardwired reaction component, while other actions go thought the deliberate process. The first group of actions are those which directly modify the agent’s behaviour without much delay, e.g. situation in which Rea’s gaze is tracking the movements of the user. In the deliberate phase *Understanding Module* interprets user’s requests as interactional and prepositional behaviours. The deliberate processing *Decision Module* then decides which action to take. To deal with propositional information the system uses a model of the user's needs and knowledge, and to deal with interactional information the system builds a model of the current state of the conversation which stores information such as who is the current speaker, listener, when to give the turn etc. The final system output is handled by *Action Scheduler* which decides which behaviours should be sent to behaviour realization engines. Like user's behaviours, Rea's behaviours are interpreted as propositional and interactional. Propositional behaviours are generated in real time using Sentence Planning natural-language-generation engine, which would later be upgraded to the BEAT system [Casselll et al., 2001]. Interactional behaviours are divided into three categories: acknowledging the user's presence, feedback and turn-taking. Each behaviour has a specific physical description, e.g. in turn-taking, if the user is speaking and Rea wants the turn, she looks at the user.

At the time it was developed, the realization system for Rea was running on two computers: one computer was responsible for displaying graphics and the other to generate speech. Rea communicates with users using voice and her comprehension capabilities are limited to a small set of utterances that have been added to the speech recognition grammar and mapped to spoken dialogue. Rea has a rather cartoon-looking body (Figure 4), her moves look mechanic and she uses synthetic voice. However, in the conducted evaluation the subjects judged her as interesting. She was compared with previous systems which used only verbal voice and results have shown that Rea is "more collaborative and cooperative and better at exhibiting natural language (though both versions had identical natural language abilities)". Users also preferred her to a menu-driven version of computer interfaces. This research result indicated a new era for interface designers. One of the advantages of autonomous agents is that they may allows for new environments beyond the plane of computer desktop, such as virtual museums and places which only exist in fairytales.

2.1.3.2 Greta

Greta agent (Figure 6) is developed by a research group of Catherine Pelachaud from University of Paris 8 [Bevacqua et al, 2007] [Mancini and Pelachaud, 2007] [Greta Documentation]. Greta is an example of next-generation ECAs – the system is compatible with specifications laid out by the research community, Greta agent can talk and simultaneously express complex emotions, even mixtures of emotions, provide backchannels based on the user's speech processing, and speak several different languages: English, Italian, French, German, Swedish and Polish. At the moment of writing of this thesis, the system computes the agent's intentions only in cases when the agent is in the role of a listener, while the component which calculates speaker's utterances is still under development.

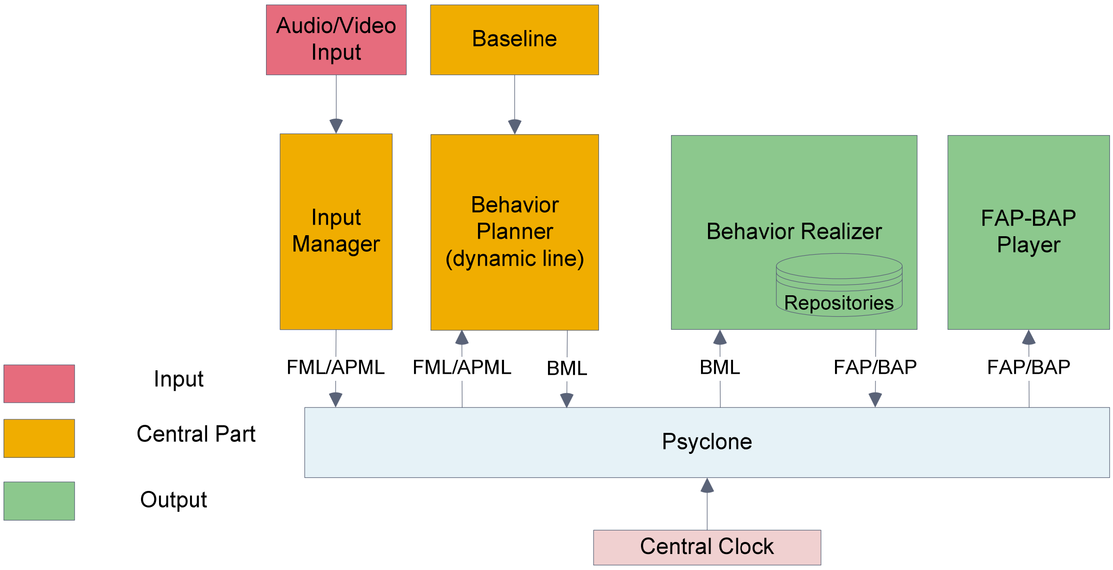


Figure 5: Architecture of the Greta system

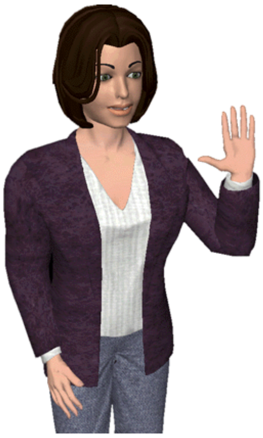


Figure 6: Greta's appearance

A simplified view of the system architecture is depicted in Figure 4. The system is running on *Psyclone server* [List et al., 2005], which is an agent-based platform for mediating messages between components which implements OpenAir specification.

Listener Intention Planner component is responsible for reacting to the user's speech and defining listener behaviours of the agent (backchannels). Theoretical basis has shown that there is a strong relationship between backchannel signals and verbal and nonverbal behaviours of the speaker. Therefore, this component collects information from the speaker and based on the agent's desires calculates responses of the agent's backchannel. The component employs Poggi's taxonomy of communicative functions of backchannels [Poggi, 2007] and the lexicon of backchannels. Listener Intention Planner may be observed as one part of Intention Planner of the aforementioned SAIBA Framework. Another part, Speaker Intent Planner, the module which calculates agent's intentions when he has a speaker role, is still under construction.

Behavior Planner gets the agent's communicative intentions and characteristics and calculates a description of physical behaviours in the BML language. These physical representations consist of a set of face movements, gestures, torso movements and head movements which are characterized by the space and velocity of execution. Expressivity of these behaviours (e.g. slow, fast hand-gestures) depends on the agent's mental characteristics – emotions, moods and personality.

Behavior Realizer and *FAP-BAP Player* are components responsible for realization of behaviours described with BML. They share the main functionality with the system devised in the thesis: input is the BML script and output is a graphical presentation of the agent to which BML behaviours are applied.

Rather than develop the complete ECA system, Greta's developers focus on constructing functional components from the SAIBA framework and proposing symbolic descriptions for expressing various kinds of emotions. In addition to the planner, they construct *Behavior sets* which represent an emotional expression. Example is embarrassment. which may be composed of two head movements (down and left), three gaze directions (down, right, left) , three facial expressions (smile, tense smile and neutral expression), open hand on mouth, and bow. The behaviours are generated depending on the Baseline module in which agent characteristics are stored. To describe the physical characteristics of executing motions they propose six expressivity parameters: amount of activity, amplitude, duration of movements, smoothness and continuity of movement, dynamic properties and repetitiveness [Pelachaud, 2009]. The parameters are applied to the Realization system to modify motion execution. For example, if the agent is low aroused (as indicated in Baseline), when she is happy she will use a light smile, and if she is very expressive she will raise her arms and make a wide smile.

2.1.3.3 Virtual Patients

Virtual patients (VPs) [Parsons et al., 2008] [Kenny et al., 2007] are examples of agents which are being developed in cooperation between University of Southern California (USC) and Institute for Creative Technologies (ICT). VPs have commonly been used to teach bedside competencies of bioethics, basic patient communication, interactive conversations, history taking and clinical decision making [Bickmore et al., 2007]. In terms of system design, these agents are successors to the agents of the Mission Rehearsal Exercise (MRE) project [Traum and Rickel, 2002] which are based around a set of distributed components that comprise the virtual human architecture.

The virtual human architecture is depicted in Figure 7. Major input component are *Speech recognition* which recognizes the user's speech by converting audio into text. It is then processed by *Natural Language Understanding and Dialogue Management* components in the central part of the system. They extract meaning and form an internal semantic representation of the user's utterance. *Intelligent Agent* is a central component responsible for reasoning that embeds emotional and cognitive models which decide about the agent's actions based on the user's utterance. The output of the Agent is a response output string and actions which describe the agent's intentions. These are then processed by the *Nonverbal Behavior Generator* [Lee and Marsella, 2005], a component which plans nonverbal actions – gestures and facial expressions based on lexical analysis and statistical data. The final output is then generated by *SmartBody* [Thiebaux et al., 2008], multimodal behaviour realization system which takes physical descriptions of behaviours as input and produces graphical representation and the agent's speech as output. The virtual human architecture is modular and allows easy integration of new components.

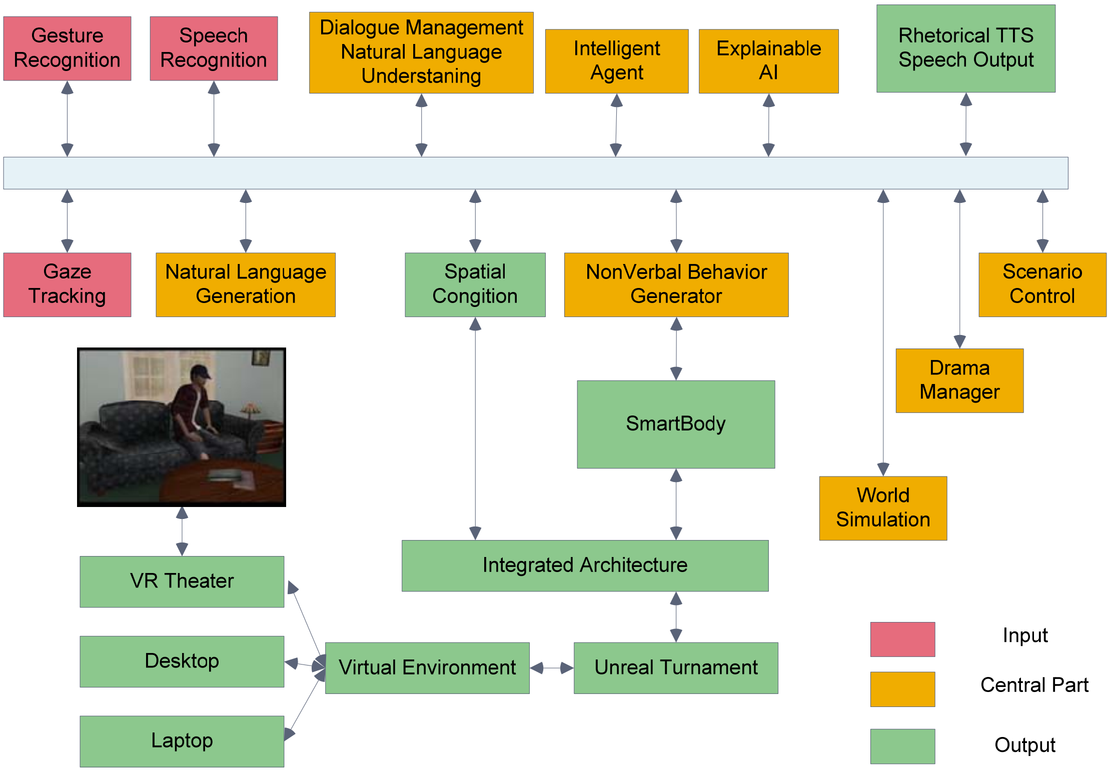


Figure 7: Components of the Virtual Human architecture

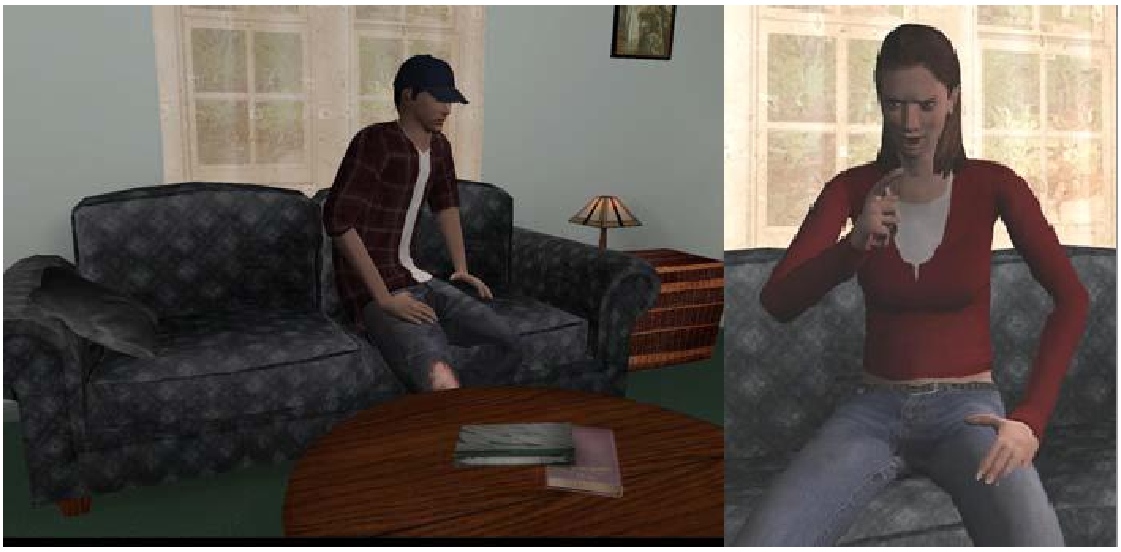


Figure 8: Virtual Patients: Justin and Justina

At the moment the developers have built two VPs: Justin, an adolescent boy with conduct disorder [Kenny, 2007], and Justina, a female assault victim character that has PTSD [Parsons, 2008]. The main challenge in building these characters was to build an interactive dialogue system which expresses interactions characteristic of the specific mental issues these patients have. Therefore, the developers conducted a study on behaviours of patients with PTSD and conduct disorder. For example, diagnostic criteria for PTSD include a history of exposure to a traumatic event, and duration of the disturbance is usually greater than one month.

VPs have been evaluated in a series of interviews with real human subjects. The biggest drawback was a failure of the speech recognition engine that did not contain all of the words or phrases asked by the subjects. In this case the patients responded with ‘I don’t get what you mean’ and the subjects got frustrated. Nevertheless, most subjects were able to continue with questioning and make a diagnosis, which is a good result for future studies. At the moment of writing, virtual patient systems have been upgraded with the sensors and input modalities which increase their capabilities.

2.1.4 Multiuser ECA systems

Most of the existing ECA systems constrain themselves to maintain conversations with a single user or conversation between several ECAs and a single user (e.g. [Traum, Marsella et al., 2008]). The reasons are the difficulties which multiuser interactions bring into dyadic ECA systems. Multiuser interactions are less predictable than dyadic interactions and more complex to model. Moreover, it is hard to collect and process data coming from several modalities and track the progress of communication.

For this reason, there are few works dealing with ECAs and multiple users. Most of them deliver theoretical bases for devising dialogue models or specify ECA behaviours in multiparty interaction, such as gazing and posture shifts. Traum [Traum and Rickel, 2002] [Traum, 2003] investigates issues which arise in multiparty dialogues, such as recognizing and giving the turn, handling participants’ channels and backchannels. However, he does not give any practical solutions to these issues. Another group of researchers [Vertegaal et al., 2001] analyses gaze patterns in a conversation between three human participants. They find how gaze behaviours regulate the flow of conversation depending on participants’ roles. They discover that the participants look about seven times more at the individual they listen to than at others, and about three times more at the individual they speak with than at others. The conclusion is that gaze is an excellent predictor of conversational attention in multiparty conversations. These rules are then applied to the computational model of gazing. Rehm et al. [Rehm et al., 2005] build the Gamble system in which an ECA plays a dice game with two human participants. The application constrains the relevant interaction: turn-taking is round-based because it follows the game scenario and social interaction is focused on game-flow rather than establishing relationships between participants. The system uses a relatively simple dialogue model and can understand only a limited number of words, which are variations of the word "yes/no", "I believe you" and "never". It works only in a predefined environment setting in which human participants sit at the opposite side of the table. The system serves more as a test-bed for investigating social behaviour of an agent towards users than for practical applications. In further study on the system the same authors observe human gazing patterns in interaction with the agent [Rehm and Andre, 2005]. They note that people spend more time looking at the agent that is addressing them than at a human speaker. This phenomenon can be explained by the fact that prolonged eye contact in a social interaction can be considered impolite and rude; hence, the agent in this game may have been regarded as an artefact rather than a human being.

2.1.5 Discussion

While pioneer ECAs have been created by individual labs, in past ten years a lot of attention has been given to building and improving individual modules which make up an ECA system. Even a single component is created by plenty of people or several research labs (e.g. [Huang et al., 2007] [Traum et al., 2008]). In parallel with these studies, researchers aim to clarify interfaces between the functional modules of ECA systems in order to allow sharing of components which are created in various labs. Example of functional architecture is the aforementioned SAIBA framework. The trends appeared because of the complexity of ECA systems which demand knowledge in multiple disciplines from their developers. For this reason, labs are cooperating in order to create effective and believable ECAs. The existing ECA architectures evolve according the proposed specifications. Their modularity allows effortless replacement or integration of the various components, so it becomes possible to create individual ECAs which are applied to various domains using a single architecture.

In current studies, most of all, researchers are studying the effects of ECAs on our evolving world. Cassell has investigated the role that the ECA can play in children's lives using Story Listening System (SLS) which can provide support for learning language and literacy skills for children [Cassell, 2004]. Virtual patients have been studied to conclude whether they provide reliable and applicable representations of live patients which can augment simulated patient programs. Considerable work is also being done on observing the behaviours that humans perform while conversing with other humans or agents, including conversation, verbal and nonverbal gestures (e.g. [Kenny et al., 2007], [Rehm and Andre, 2005]).

Nevertheless, natural interaction with ECAs has not yet been achieved. Speech technology, which allows for natural and unconstrained interaction with ECAs, is not 100% accurate. Proper speech processing requires understanding the voice regardless of the speaker’s gender, age and culture. Other input sensors, which collect the user’s nonverbal behaviours measure attention and affects, are being studied experimentally. Dialogue systems are restricted by application domains, as well as natural language understanding and generation components. Despite the emotion and personality models, virtual characters are not completely believable. This is partly a consequence of the issue of synthesizing realistic behaviours for conversational characters.

2.2 Realization of Multimodal Behaviours of ECAs

The behaviour realization process, which is briefly introduced in the previous section, is thoroughly explained in further reading. Moreover, we describe requirements for reactive and realistic behaviour realization and state of the art in the area of animating expressive virtual characters.

2.2.1 Multimodal Behaviour Realization Process

Multimodal behaviour realization is done through real-time control of animation algorithms which are applied to the content of behaviour representation scripts. To describe this process in detail we refer to Figure 8, which is a schematic view made on the basis of functionality and requirements of multimodal realization systems.

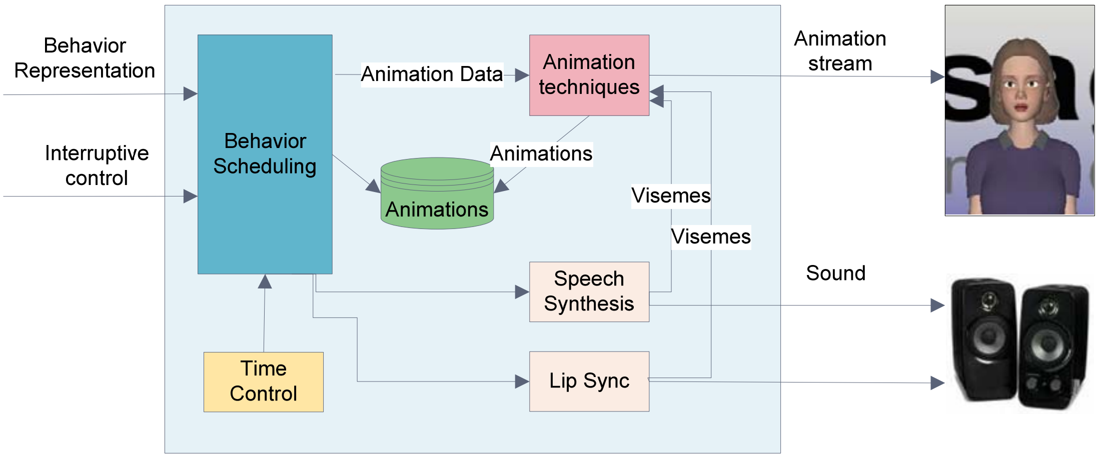


Figure 9: Multimodal behaviour realization process

One of the inputs into the realization system is the physical description of the ECA's behaviour, which is defined through a *behaviour representation* language. Physical description tells the system what nonverbal behaviours will be applied to the ECA's body and what the ECA will say. So far, several representation languages which vary in level of abstraction have been proposed. Most recent is the Behavior Markup Language (BML), which is part of the SAIBA framework introduced in Section 2.1.2. BML has been established for the purpose of standardizing realization process and thus has been utilized in several production tools and components [Vilhjalmsson et al., 2007]. Structure of the representation language has an important role in the realization process since its addresses a variety of behaviours that should be efficiently controlled by the realization system.

Another input of the realization system we define as *interruptive control*. These functions affect the scheduling mechanisms which coordinate animation algorithms applied to the ECA's body (*Behaviour Scheduling*). During realization the functions act as interruptions which happen due to various events in the environment. Example is a situation in which an ECA should respond to a person who enters the room while still speaking. In that case, all planned behaviours are re-scheduled – ECA stops speaking and greets the newcomer.

The behaviour realization process is done by applying behaviours to the ECA's body. Since most representation languages, including BML, define multimodal behaviour as a combination of primitives (e.g. emotion expression, speech and hand gesture), realization process relies on coordination of executed animations across body parts. This is also an approach which has been proposed in neuroscience research dealing with biological motion organization [Thoroughman and Shadmehr, 2000]. Non-trivial nonverbal behaviours are composed into motion controllers which are then controlled hierarchically and combined using motion blending techniques. The control of behaviours is provided with a central unit we name *Behaviour Scheduler*. Scheduler receives time information from *time control* and decides the moment of applying behaviours to an ECA. Realistic and responsive behaviour may be achieved using a combination of *animation techniques*. Selected animations, which are pre-recorded or procedural animations, may be combined using motion blending, time warping or other animation techniques. Animation stream and sound are produced as output of the realization process and it may be generated either using *text-to-speech synthesis* (TTS) or *lip-sync* from audio files.

In sections that follow we present background on behaviour representation (Behavior Markup Language) and animation techniques for interactive virtual humans. Finally, we explain existing animation and multimodal behaviour realization systems.

2.2.2 Behaviour Representation: Behavior Markup Language

In parallel with the development of ECAs and their modules, several representation languages have been developed. Most of them define relations between speech and nonverbal behaviours. Example is Greta system and APML language [Hartmann et al., 2002], which uses speech as reference for generating and realizing gestures and other nonverbal behaviours. Using speech as guidance for gesture timing preserves conversational rules, including structure of communicative utterances and speech prosody, but it also limits the possibility of describing human behaviours since some gestures may occur in absence of speech.

BML specification is set up to overcome the issues of existing representation languages and become a technical standard. BML is based on XML and provides descriptive power for behaviour execution and feedback mechanisms to control execution progress. Its method of specifying behaviours grants a high level of control over animation execution. In BML multimodal behaviours are described as a set of primitive behaviours - nonverbal behaviours and speech. Primitive behaviours are called BML elements. They are further annotated with attributes, which vary among different behaviours (speech, locomotion, gesture…) and specify additional information related to physical execution of the behaviour. For example, element "head" specifies type of head movement ("nodding" or "shaking"), number of repeats and amount.

In BML a multimodal behaviour (a user's utterance) is defined by a single <bml/> block which contains primitive actions specified by a number of BML elements. Actions are synchronized via behaviour synchronization points delimiting execution phases of each BML element (Figure 10)

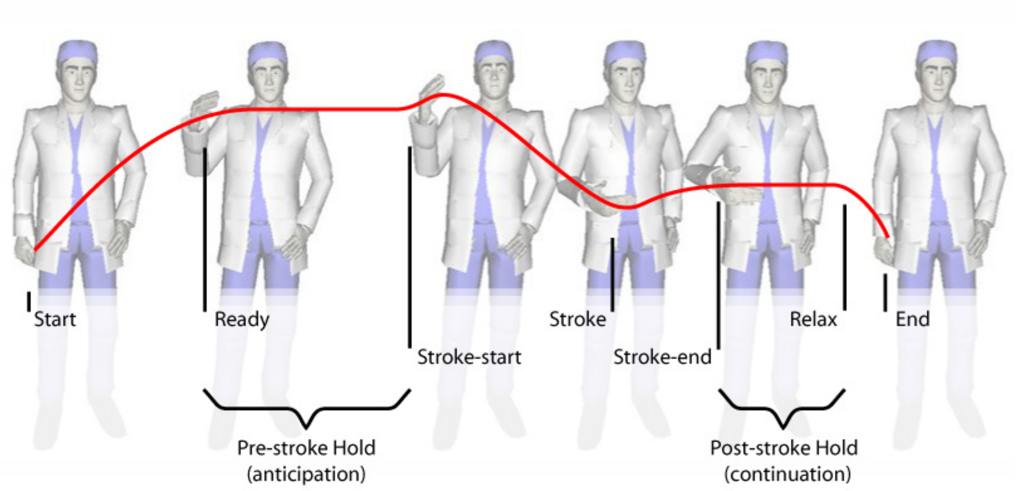


Figure 10: Phases and standard synchronization points of BML elements

The points are the same for all BML elements and are used for making cross-references in a multimodal behaviour. Behaviour realization system's scheduler uses these references to control the execution process. BML time control also includes pauses and time delays which postpone behaviour execution. This is a core property of the BML language which each BML-compliant behaviour realizer must support. However, to provide support for animations which cannot be defined with core BML, two more optional levels are defined:

- Levels of description. Additional levels of description are embedded within BML as external support for existing representation languages or new languages with advanced realization capabilities. An example is Speech Synthesis Markup Language (SSML, [SSML]), which is designed for assisting the generation of synthetic speech in Web and other applications.
- New behaviours. Developers of BML realizers can explore new behaviour elements and add them to behaviour execution. Those experimental components are identified as non-standard BML by utilizing XML namespaces to prefix elements and attributes.

Concatenation of behaviour elements and alignment of sync points provide a rich representation of multimodal behaviours for ECAs. While BML describes how behaviours are aligned, the burden of animation quality rests entirely on the realization system and animation techniques chosen by developers.

2.2.3 Techniques of interactive virtual human animation

There has been a huge growth in the field of 3D character animation in the last ten years. With more and more animated films and games coming up, the entertainment industry has a growing demand for realistic and exciting animations. Thus, various animation methods for designing, editing and synthesizing character movements have been invented. The proposed methods operate on animations which can be produced in three different ways:

1. *Manual approach* is similar to what has been used to produce animated movies in the past. Nowadays an animator’s workload is significantly reduced thanks to animation production tools like Autodesk 3ds Max or Maya which feature a variety of animation techniques, such as animation grouping, widgets for

manipulating the trajectories of the character's joints, sticking hands and feet to environment objects, inverse kinematics... The manual approach produces realistic and compelling animation examples only if a lot of time is invested. The reason is presence of many low-level parameters - one-second long animation of only one character can have more than ten thousand parameters that can be manipulated. Another drawback of this method is that an animator starts every animation afresh and for some animations he does a redundant work. Example is walking animation, which consists of a sequence of similar cycles.

2. *Procedural animation methods* refer to algorithms which combine approximations of different physic laws. Examples are fundamental approaches used in mechanics, such as kinematics and dynamics, which are standard for 3D character animation. They describe motion in space along a 3D line, which is also known as trajectory or path. In dynamics an animator provides initial conditions and adjusts rather abstract physical parameters, such as internal forces, external forces, and user applied forces, which control positions and motions of virtual humans during execution. The effect of changing parameter values is often unpredictable and to see the result, the animator has to run simulation. These improvisational results show virtual humans as "puppets that pull their own strings". Although procedural methods provide high control essential for interactive applications, the drawbacks are high computational cost and lack of realism.
3. *Motion Capture methods* have grown to be very popular for producing motion in the last few years. Since they are based on tracking movements of real humans, these methods produce realistic and compelling results. Motion capture uses tools such as Vicon BLADE [Vicon Blade] and Organic Motion [Organic Motion] with sensors attached to an actor's body and infrared cameras that track human movements. During recording session digital readings are fed into linked computers, where they are recorded and interpreted into motion sequences. In this way a large database of sequences can be recorded and later applied to a virtual character. However, most captured data is often noisy and contains errors resulting from the capture process.

While manual approach and motion capture methods produce examples of motions, procedural approach synthesises motions on the fly according to a specific set of parameters, providing a high level of control which is very important for interactive applications. However, it is very hard to obtain realism with procedural actions. On the other hand, motion examples look natural, but for real-time applications they usually require modification. For example, animation of human pointing is usable only if pointing direction meets the requirements of the application. Usually, it is difficult to modify and reuse examples in another context because even small changes to data might decrease motion realism. These are the reasons why example-based motion synthesis has become a topic of interest and in character animation field. Simple motion synthesis techniques are interpolation or weighted blending which combine two or more animations according to their weights. However, in recent years, efforts in this field have brought technological advancement which increases the potential for on the fly motion synthesis. Animation examples are also combined with the flexibility of procedural motion synthesis. This has resulted in several research streams in editing and processing data for later reuse. Three of these are chiefly relevant for interactive character animation. These are re-ordering motion clips using graphs [Kovar et al., 2002], [Arikan et al., 2002], interpolating motions between motion clips [Kovar and Gleicher, 2004] and constructing models of human motion [Li et al., 2002]. Besides, there are also hybrid approaches which are instrumental in combining some of the ideas above, such as motion transition graphs, as demonstrated in locomotive motion generation [Kwon, 2005] [Park et al., 2004] and rhythmic motion synthesis [Kim et al., 2003] (For further reading on this subject we refer to [Pejsa and Pandzic, 2010].). However, most motion synthesis approaches are applied to realizing actions in 3D space, e.g. combination of locomotion, running, or jumping over obstacles. There has been a little of work in combining advanced motion synthesis techniques to realize communicative or affective behaviours (e.g. [Pan et al., 2007]).

Procedural animations or animation examples control the visual appearance of a 3D character. Usually different approaches are required for controlling the face and the body. For the body they involve controlling body parts at the graphical level via joint transformations (bone-based) and for the face they control surface deformations (geometric-based). Exceptions are trends in the game industry which share bone-based principle for both modalities. Thus, animation of conversational behaviours on low-level can be viewed distinctively as face and body animation.

To animate surface deformation of the face, different approaches which control low-level face parameters (e.g. points, bones and/or muscles) have been proposed. Some of the approaches define key surface points, like MPEG-4 Face and Body Animation (MPEG-4 FBA) standards [Pandzic and Forchheimer, 2002] [Moving Pictures Expert group]. MPEG-4 FBA defines in total 68 Facial Animation Parameters (FAPs) for high-level expressions (visemes and facial expressions) and low-level movements of individual facial feature points. Other face animation approaches define regions of facial movements, like Ekman and Friesen's [Ekman and Friesen, 1978] Facial Action Coding System (FACS). FACS is an anatomically-oriented coding system, based on basic "Action Units" (AUs) that cause facial movements. Initially, it has been proposed for the purposes of systematic observation of facial expressions. Later, researchers have realized that FACS can be used in computer graphics and thus it has been utilized for facial motion synthesis. Some FACS-based face animation systems map Action Units to MPEG-4 FBA parameters [Raouzaiou et al., 2002] or even blend emotional expression with Action Units and FBA parameters [Mäkäräinen and Takala, 2009], [Malatesta et al., 2009]. Other systems define morph targets based on FACS. Finally, among other approaches to drive face movements a lot of attention has been given to muscle-based approaches. These methods try to model the physics of the skin, fat, bone and muscle tissue on the face [Platt and Badler, 1981]. In muscle-based models, animation is done by moving the face muscles in intervals. Such methods can be very powerful for creating realism but complexity of the face model increases computational expenses and causes difficulties in face modelling. This is the reason why this approach has not been popularized. To decrease the number of facial muscles there are also approaches which suggest integrating geometric deformation of a face mesh with muscle-based models [Noh, 2000].

Body animation approach differs a lot from facial animation principles. The control of body movement is done by moving the skeleton which consists of a hierarchical set of bones and joints (which can be defined according to e.g. H-Anim or MPEG-4

FBA standard). When one joint moves, rotations and transitions propagate down to all of its sub-joints, e.g. elbow joint rotates the whole arm, just exactly as it happens in reality. The pose of each skeleton joints is represented with a 4×4 transformation matrix that contains a translation and a rotation component, where the rotation is defined as a 3 × 3 sub-matrix. In computer graphics, a common way to represent rotations is using quaternions, which are an extension of complex numbers. Quaternions are very easy to manipulate and calculate rotations in a 3D space. They are appropriate for real-time applications, but difficult to visualize. Quaternions specify poses of a skeleton which affect a 3D mesh, representing a virtual character. This process is generally called skeleton-driven deformation.

For ECA developers it is important to control ECA motions on a high level, using descriptive communicative actions – speech and nonverbal behaviours. Multimodal behaviour realization system maps these descriptions to low-level parameters; skeletal joints and face parameters. The mapping is done by applying procedural or pre-recorded animations using techniques of motion synthesis, such as motion blending. The animation approach used for animation of a single physical behaviour depends on behaviour characteristics. For example, visual cues of a spoken utterance are very easy to animate. Existing TTS systems (e.g. Microsoft Speech Application Programming Interface [MSAPI]) provide on the fly phonemes/visemes which are then mapped to lip shapes, whereas lip-sync systems derive visemes from a speech signal using speech processing techniques. Lip shapes are usually morph targets modelled using commercial tools like Autodesk 3ds Max. During realization of speech, interpolation between different morph targets is done. Among other nonverbal cues, a lot of attention has been paid to animation of face movements, since face provides cue signals that accompany speech and stress emotions. In addition to that, body motions are harder to animate realistically, especially expressive hand movements and posture shifts. Gesture and posture shift examples from motion capture systems look compelling, but they are characterized by dynamics and 3D space of execution which depends on the inner state of an ECA. To change the expressiveness, motion synthesis techniques should be incorporated into animation systems.

2.2.4 Multimodal Behaviour Animation and Realization Systems

Up until a few years ago, several realization systems and representation languages at different levels of abstraction were proposed. These systems vary in their functionalities and supported behaviours: some of them operate from communicative planning to behaviour realization, others just realize physical behaviours; some of them are focused only on facial motions, others animate the full body. The most relevant examples of these systems we explain through three different categories: nonverbal behaviour planners, emotion generators and BML-compliant realizers.

2.2.4.1 Nonverbal Behaviour Generation

A lot of work has been done on systems which generate – i.e. plan and realize – nonverbal behaviours and speech using a variety of communication rules. Most of these systems work as black boxes - they use their internal behaviour representations and animations which cannot be isolated and used for other purposes. The systems are based on a theoretical background from linguistics, psychology, and anthropology, which has proven the great importance of nonverbal behaviours in face-to-face communication and its strong relationship with speech. For example, research findings show that eyebrow movements can emphasize words or may occur during word-searching pauses [Ekman, 1979], iconic gestures accompany certain lexical units [McNeill, 1992], co-expressive gesture and speech are shaped by dialectic between linguistic expressions and spatio-motor representations [McNeill, 1992], and synthetic displays are the most frequent facial gestures accompanying speech [Chovil, 1991]. These theoretical findings are coded and then used to construct behaviour generators. The proposed nonverbal behaviour generators focus on animating either facial expressions or full body motions, including hand gestures and posture shifts. In these systems expressivity of gestures is described with internal representations which however have not been widely accepted.

One of the earliest and probably best known examples of such systems is Behavior Animation Toolkit (BEAT) [Cassell et al., 2001], which has been utilized in the Rea agent. BEAT is designed as a plug-in system which applies lexical analysis to a spoken text, then looks for gestures from a gesture library and generates spoken utterance enriched with nonverbal behaviours - facial expressions, hand gestures and head movements. In BEAT system gesture selection is based on manually-written rules that can be modified and matched to gestures in the gesture library. BEAT focuses only on a few gesture examples which are generated procedurally, but which seem to be insufficiently realistic. In addition to that, its scheduler is unable to adapt gestures to scheduling constraints, nor refer to their inner structure. Thus, preparatory action and transitional behaviours must be independently scheduled or left to the animation engine.

To create believable characters Hartman et al. [Hartmann et al., 2006] identify expressivity parameters through selection of gestures in a planning phase (which gestures to use?) and on the execution level (how to display the gestures?). They build a system which generates gestures varying in activation, spatial and temporal extent, fluidity and repetition, and which employs AMPL language for behaviour representation [Hartmann et al., 2002]. For gesture synthesis the system uses kinematic algorithms of IKAN library [Tolani, 1998]. IKAN computes shoulder and elbow rotations of the character's body for given target position but with insufficient level of realism.

Another system which analyses lexical structure of spoken text and prepares nonverbal behaviours is based on architecture for human gesturing, HUGE [Smid et al., 2006]. This architecture however can accept various inducements as input, including speech and/or its acoustic features. While BEAT relies on manually-designed generation rules, HUGE system uses statistical data obtained from analysis of videos of real humans, primarily TV speakers. Speaker utterances are animated with procedural actions: eyebrow moments, head swings and nods [Brkic et al., 2008], eyelid and mouth movements. The applied animations are executed sequentially, so there is no possibility to run multiple actions simultaneously. Body motions (hand gestures and posture shifts) are not generated at the moment. Another study that has been conducted with HUGE is proposed in [Zoric et al., 2009]. The system processes prosodic features of speech and then finds matching nonverbal behaviours using Hidden Markov Models (HMMs). HMMs replace the statistical model and are trained to recognize whether a particular gesture occurs for a specific acoustic feature. Generated animations are the same as in the case of the system with a statistical model.

Another behaviour generation system that plans nonverbal behaviour in addition to speech is the open-source Nonverbal Behavior Generator [Lee and Marsella, 2006]. Unlike other systems, Nonverbal Behavior Generator has a layered architecture which has been designed to support FML and BML languages. The generator processes the syntactic and semantic structure and affective states in spoken text and annotates it with appropriate nonverbal behaviours by using statistical data. Developers base this system around affective states obtained from analysis of videos of emotional dialogues. The planned behaviours are realized using SmartBody system [Thibeaux et al., 2008], which is an open-source realizer for full body animation.

The most advanced system which generates realistic hand-gestures and posture shifts is the one developed by Neff et al. [Neff et al., 2008]. The system uses the same approach as BEAT system – it takes input text and plans nonverbal behaviours, but compared to BEAT it uses a character-specific gesture style. These styles are based on individual statistical models which are developed by carefully annotating videos of real humans. This approach is similar to the one used in HUGE, but with one main difference – hand gestures and posture shifts are taken into account. The system pays a lot of attention to planning both the gestures' internal structure and their macro-structure. For synthesis they use a dynamic, controller-based approach to physically-simulated animation with hand-tuned controllers [Faloutsos et al., 2001] and controllers to track motion capture data [Zordan and Hodgins, 2002]. The overall approach has been shown to yield impressive results. However, the system cannot be used for real-time purposes since it takes a lot of time to compute animation sequences - in a conducted study it took 14 minutes to produce 58 seconds of animation.

2.2.4.2 Emotion Expression Generation and Animation

Other realization systems for conversational characters focus on modelling and generating affective faces. Among those, there are generators based on computational models of emotions and moods, and low-level systems which concentrate on synthesizing facial expressions and their blends. On low-level, a lot of attention has been paid to facial expression synthesis using Ekman's FACS and MPEG-4 FBA standard. While Ekman's system can be used to synthesize an essentially infinite number of facial expressions, MPEG-4 has a strong correlation with computer vision methods which are commonly used for modelling realistic motions. The pre-recorded examples, however, require a multidisciplinary study on the physical characteristics and perceptual meaning of pre-recorded expressions before motion synthesis methods can be applied to derive new expressions.

In facial expression synthesis, various techniques focus on mathematical accuracy or efficiency of their algorithms and on analysis aspect of synthesized facial expressions. While early studies in the area have dealt with synthesizing 6 archetypal types of emotions, new trends address the importance of realizing intermediate expressions. Some facial animation systems compose intermediate expressions of archetypal types according to their weights [Deng and Ma, 2009][Ostermann, 1998]. Averaging is done on the level of low-level facial features and results are not necessarily anatomically correct expressions. Another approach to blending emotions displays two expressions on different parts of the face [Martin et al., 2006]. This approach encounters difficulties if two expressions affect the same region, for example mouth in case of fear and disgust. The system proposed by Mäkäräinen and Takala [Mäkäräinen and Takala, 2009] overcomes this issue by using parametric elements which combine FACS and MPEG-4 standard. It generates facial expressions by combining facial displays from AUs and finding the balance for skin tissue under forces which are acting on it. The system enables combination of up to three basic emotions, but does not focus on discovering the stimulus on human observers. Some of the researchers systematically address the mechanisms of emotion elicitation process and create an animated face in which intermediate expressions are shown. Example is work of Malatesta et al. [Melatesta et al., 2009], who build a facial expression generation system based around Scherer's appraisal theory [Scherer et al., 2001]. In Scherer's theory, the main idea is that emotions are extracted from individual evaluations of events that cause specific reactions in different people. The work aims to investigate this effect through the synthesis of elicited facial expressions by mapping Ekman's FACS to MPEG-4 FBA. Blends of emotions are created using MPEG-4 FAP parameters stored in profiles of basic and intermediate expressions. The system developed by Arellano et al. [Arellano et al, 2008] relies on affective model of emotions and personality. The model is based on Mehrabian's theory which connects PAD space with personality traits [Mehrabian, 1995]. The model generates emotional states based on environment events and previously experienced emotions. Emotional states are then mapped to facial expressions synthesized with MPEG-4 parameters. Animations are modelled upon videos of a 3D database of human facial expressions using computer vision methods.

In facial expression synthesis, there have been also some studies in discovering statistical rules for synthesis of different facial expressions. Nevertheless, most of these rules observe patterns insufficient to obtain perceptually realistic results. If expressions are modelled using computer vision methods, this issue can be solved by increasing the amount of obtained data. An example is the facial expression generation system offered by Mana et al. [Mana et al., 2006]. To create the rule-generation database, they use 28 markers to track humans and then train Hidden Markov Models (HMM) with the obtained patterns and corresponding expressions. Their generator produces facial expressions of different intensity which are synthesized with MPEG-4 FBA parameters. The approach is applied to 6 archetypal types of emotions. Another example of a face generator, FacePEM, uses a hybrid of region-based facial motion analysis and statistical learning techniques to generate facial expressions [Deng and Ma, 2008]. Statistical learning is done on data obtained from computer vision methods. The system can be applied to an animation sequence to drive emotional expression, including the conversational behaviour.

2.2.4.3 BML-compliant Behaviour Realizers

At the moment of writing this thesis, three BML-compliant animation engines have been developed; Articulated Communicator Engine (ACE), SmartBody [Thibeaux et al., 2008] and Embodied Agents Behavior Realizer, EMBR [Heloir and Kipp, 2009]. Articulated Communicator Engine is a development of an earlier engine based on multimodal utterance representation markup language (MURML) [Kopp and Wachsmuth, 2004]. To control movements, the system uses primitive controllers which operate on different parts of the character's body: arm, head, neck, etc...

Realization in this system is done using blending, but authors of this work do not provide generic and flexible reconnections and processing. ACE system supports facial expressions, lip synchronization, visual text-to-speech synthesis and hand gestures, and it can be integrated with different graphics engines. The animations are modelled using procedural methods and lack realism. The system has been utilized in two different projects - NUMACK and Max.

SmartBody system is based on a controller hierarchy which includes special meta-controllers for motion blending, time-warping and scheduling. Moreover, it automatically handles warping and blending operations to meet time constraints of the running actions, which are executed using keyframe motions (MEL scripts of Autodesk Maya) or procedural actions. The system supports facial expressions, speech, body postures, body gestures, gaze, head motions, feedback notifications and interruptions of running behaviours. It has been utilized in several projects and integrated with a variety of engines, including OGRE, GameBryo, Unreal and Source. SmartBody is engine-independent and interacts with the rendering engine via TCP.

The most recent of the BML compliant realizers, EMBR system, is architecturally similar to SmartBody. It is based on a modular architecture that combines skeletal animation, morph targets and shaders. Novel solution in this work is EMBRScript control language, a wrapper between the BML language and realizer. The language supports specification of temporal and spatial constraints on the resulting animation. Animations are produced on the fly, using motions segments, which are blocks of animations computed from recorded animations, key frames, morph targets and an inverse kinematics system. EMBR relies on the free Panda3D graphics engine and Blender modelling tool. Authors of this work also create a plug-in for the Anvil annotation tool [Kipp, 2001], which translates annotations into EMBRScript language.

2.2.5 Discussion

In recent years a significant progress has been made towards natural synthesis of conversational behaviours. First, a clear definition of representation of physical behaviours is given. Second, while early systems suffered from mechanical-looking, procedurally generated gestures, the newer generation of behaviour realization systems uses a combination of animation techniques to achieve believable and interpretable behaviours. Usually, pre-recorded examples are annotated with physical characteristics of behaviours and during realization they are combined with techniques of motion editing or procedural animations blended and combined with procedural animations. For handling motion synthesis, the importance of providing a framework of controllers responsible for motion scheduling which incorporates advanced motion synthesis methods has been stressed. Example of such a system is SmartBody.

However, there are still open problems in this area. Most of the proposed realization systems do not have a clear art pipeline and are tied to various graphics platforms, tools and formats for modelling virtual characters and animations. They also require installation of specific text-to-speech engines, and do not have a flexible architecture with clear integration interface and animation API. There is a problem of retargeting, i.e. applying animations to characters that have geometrically different body models. They do not support other behaviours than communicative ones. Examples are locomotion and other actions in which the character interacts with the environment, such as grasping.

3 RealActor: Multimodal Behaviour Realization System

In this chapter we present the main contribution of this thesis – RealActor, a multimodal behaviour realization system for Embodied Conversational Agents. First, we give an overview of Visage|SDK [Visage], animation framework used for the system implementation. We also explain how animation can be modelled using Visage|SDK. We then explain RealActor's motion control mechanism, i.e. how constraints are applied to the motion synthesis process and how the system takes multiple animation streams and mixes them in a meaningful way. We then give an overview of the overall system, explaining the realization process on an example of a BML representation script. The next section is dedicated to neural networks, which are integrated into the system to predict timings of synthesized speech and align behaviours to spoken words. This is a novel solution for synchronizing nonverbal behaviours and speech coming from TTS. Then we give an overview of animations used for motion synthesis, presenting a brief study on gestures and their parameters used to build an animation lexicon. Finally, we present system features and implementation approach.

3.1 Character Animation with Visage SDK

Visage|SDK is a commercial C++ framework for virtual character animation, which is a product of the company Visage Technologies. The framework consists of tools which provide support for both interactive and off-line animation with real time lip-sync [Zoric and Pandzic, 2006] [Axelsson and Björnhall, 2003], speech synthesis, facial features detection and tracking (facial motion capture [Ingemars, 2007]). It also includes export and import plug-ins for modelling and animation packages such as Autodesk 3ds Max [Autodesk] and Singular Inversions FaceGen [FaceGen], as well as automated facial morph target generation and parameterization for the models.

The toolkit synthesizes character motion using MPEG-4 Face and Body animation (FBA) standard [Pandzic and Forchheimer, 2002]. As for character and 3D virtual environment rendering it supports graphics engines like Ogre [Ogre3D], Horde3D [Horde3D] and Irrlicht [Irrlicht3D]. Integration of visage|SDK applications with a graphics engine is straightforward and amounts to providing an implementation of a scene graph wrapper.

3.1.1 Character representation with MPEG-4

MPEG-4 FBA is part of MPEG-4 standard (ISO14496) which provides tools for model-based encoding of video sequences containing human faces and bodies. It encodes a 3D synthetic model that encompasses realistic or cartoon-like 2D and 3D characters, which can be defined and animated by specific FBA parameters. MPEG-4 FBA defines two sets of parameters to define the geometry and animation of the face and body of the virtual character:

- Definition Parameters specify the geometrical shape definition of the FBA model by means of FDPs (Face Definition Parameters) and BDPs (Body Definition Parameters). These parameters allow the decoder to create an FBA model with specified shape and texture.
- Animation parameters define the animation of the face and body by means of FAPs (Face Animation Parameters) and BAPs (Body Animation Parameters). There are in total 66 low-level Face Animation Parameters (FAPs), two high-level FAPs, and 196 Body Animation Parameters (BAPs). Animations are applied to the character as streams of FBAP values.

FAPs are defined following the study on minimal face actions which are in close relation to muscle actions. The specification defines values of these parameters – the maximum values produce unnatural expressions for character face, but they can be used to animate cartoon-like faces. To allow interpretation of the FAPs on any face model, FAPs are expressed in terms of Face Animation Parameter Units (FAPUs), which define distances between key facial features (e.g. mouth-nose distance). There are also two high-level FAPs: expression and viseme. Expression FAP can specify intensity of two out of six basic facial expressions by Ekman. Viseme parameter can specify two visemes out of a predefined list of 14, and a blending factor to blend between them.

The MPEG-4 Body representation was defined simultaneously with Web3D Humanoid Animation 1.0 specification [H-ANIM]; therefore it is based on the same model to represent a standard humanoid body. MPEG-4 BAPs define angles of rotations for each joint in the virtual character's skeleton. In the skeleton each joint has degrees of freedom (DOF) which specify its pose. One DOF defines just one kind of movement, for example human ankle performs two kinds of movement, flexion/extension and inversion/eversion, allowing two degrees of freedom. MPEG-4 BAPs specify the values of DOFs and the unit of rotation (BAPU) is defined in radians. Character position in 3D space is defined using six DOFs for the global location. The unit of translation for translational BAPs is measured in millimetres.

MPEG-4 FBA specification inherits the principles for face and body animation - face parameters are generally manipulated in linear space and there are less feature points to control, whereas the body has more degrees of freedom (DOFs) and body motions can be described by joint rotations and translations.

3.1.2 Character and animation modelling and integration

Visage|SDK tool has a flexible pipeline which allows for character modelling in FaceGen or Autodesk 3ds max. The overall procedure is done is several steps which consist of manual and semi-automatic methods. Once characters are modelled they should be prepared for animation. FaceGen modelling tool allows for easy creation of detailed face models for use in visage|SDK applications. Moreover, it can automatically generate morph targets needed for facial animation, including all visemes and facial expressions. Rigging is done manually, placing the skeleton inside the character's mesh. Rigged characters can be exported into formats like COLLADA [Collada] (format supported by Horde3D engine) or VRML (Virtual

Reality Modelling Language) [VRML]. To prepare the character model for animation with visage|SDK, the designer needs to use FBAMapper, a tool which maps MPEG-4 FBAPs to joints and/or morph targets of the model. This high-level model is then serialized to a simple XML-based file format called VCM (Visage Character Model). VCM files are independent of the graphics engine and reusable, so once a character has been prepared for MPEG-4 FBA animation, it can usually be used in different graphics engines without necessitating modifications to FBAP mappings.

Integration of Visage|SDK with different graphics engines is straightforward. It amounts to providing an implementation of the scene wrapper which then for every frame of animation receives the current set of FBAP values and interprets the values depending on the character's FBAP mappings. Typically, BAPs are converted to Euler angles and applied to bone transformations, while FAPs are interpreted as morph target weights. Thus, a graphics engine only needs to support basic character animation techniques skeletal animation and morphing – and the engine's API should provide the ability to manually set joint transformations and morph target weights. There are currently implementations for Ogre, Horde3D and Irrlicht engines.

To model characters, Visage|SDK also provides stand-alone tools for production of face models and morph targets that use Visage|SDK's content production libraries. These include a face model generator which generates 3D face models from 2D photographs and facial motion cloner which can automatically create morph targets on face models originating from applications such as Singular Inversions FaceGen. These tools rely on intermediate file formats (FBX [Autodesk], COLLADA and VRML) for model import and export.

Animations are stored and applied to the character as streams of face and body animation parameter (FBAP) values. When multiple animations affect the same character, they are blended together using linear interpolation. At the moment, our research group is working on a new animation engine that will use techniques such as registration curves [Kovar and Gleicher, 2004] and motion graphs [Kovar et al., 2002] to achieve better and more natural blending and transitions. Animation streams can be created using a variety of animation production techniques which are presented in Chapter 2. Visage|SDK provides a plug-in for Autodesk 3ds Max which exports animations into FBA (Face and Body Animation) format employed by visage|SDK. The animations in 3ds Max can be manually crafted or created using motion capture. Nowadays, there are a few motion capture databases available on the Internet (e.g. Carnegie Mellon University Motion Capture Database [Carnegie]). Visage|SDK also provides real-time facial feature detection and tracking tool [Ingemars, 2007] which can be used to track facial motion in live or prerecorded video streams. Tracked features correspond to CANDIDE-3 model [Ahlberg, 2001], but they can be re-used and applied to MPEG-4 FAPs using an offline processing technique.

Animations for Visage|SDK applications can be also modelled with procedural techniques implemented in C++ programming language. Visage|SDK provides C++ programming API for applying stream of FBA values to character motions.

3.2 Control Mechanism

In Chapter 2 we introduced requirements of multimodal behaviour realization. Here, we will discuss specific requirements for behaviour realization based around BML. We introduce issues for control mechanisms and explain how meaningful behaviours are combined to generate realistic conversational behaviours. Finally, we present the system's design and behaviour realization process.

3.2.1 Control issues and requirements

Multimodal behaviour realization process with BML scripts requires fine tuning of BML primitive units (behaviours), which are realized through animations of nonverbal behaviours and/or speech. Since primitive behaviours may have relatively complex relations, we believe that a pre-processing step, in which time boundaries for behaviours are defined, is necessary for effective multimodal behaviour realization.

We explain this claim on the following example of a BML script:

```
<?BML version="1.0"?>
<act>
<bml id="bml2">
     <gesture id="g1" stroke="s1:tm" type="beat"
     hand="right" handlocation="outward;center;near"/>
     <speech id="s1">
          <text> Trust me, I know what I'm <sync id="tm"/>
doing. </text>
     </speech>
     <head id="h1" stroke="g1:start" action="rotation"
          rotation="nod" amount="1"/>
</bml>
</act>
```

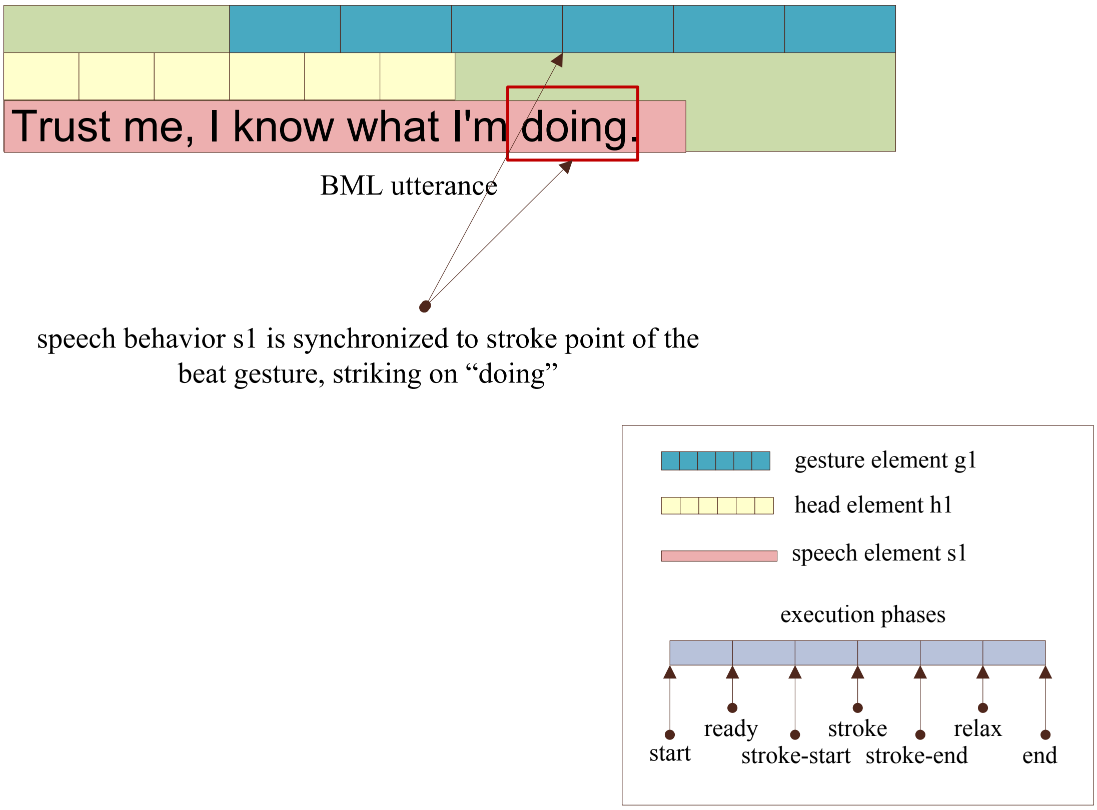


Figure 11: Example of behaviour synchronization in a BML utterance

In this example, the speaker utters the sentence "Trust me, I know what I'm doing." (specified in the speech element). As he pronounces the word "doing", the speaker makes a hand gesture (gesture element) and nods his head reassuringly (head element). More specifically; at the moment when hand gesture starts, stroke point of the nod needs to occur. Then, stroke point of the hand gesture needs to be aligned with the word "doing" (BML does not define whether it is a begging, middle or end of the alignment word, so we consider that stroke point of the hand gesture can occur in any moment when word “doing” is pronounced). Thus, hand gesture needs to starts before “doing” is pronounced, while nod needs to start before hand gesture as it stroke phase lasts (Figure 11). Relations between these primitive units require smart scheduling of animations and its coordination with speech during realization. (Physical features of the executed motions - hand gesture and nod are specified with BML’s attributes described in the RealActor reference manual, which is appendix to the thesis.)

A well-known issue of multimodal behaviour realization is timing of speech which is being generated with text-to-speech (TTS) engines. Most TTS engines do not provide a priori timing information for words and phonemes, except a few, such as free engines Mary TTS [Schröder and Trouvain, 2003] and Festival [Taylor et al., 1998]. In the BML example, hand gesture needs to start playing in advance, so that the onset of their stroke phase corresponds with the uttering of the word "doing". Existing BML realizers, SmartBody and EMBR, which use Festival and Mary TTS respectively, solve this issue with the *animation planning* approach, in which pre-processing and manual annotation of animation clips is done in order to plan animations in addition to the timings received from TTS engines. On the other hand, ACE engine uses *dynamic adaptation* and attempts to synthesize necessary synchronization data on the fly by contracting and prolonging procedural animations which are aligned to the spoken text. Microsoft SAPI, which we use in RealActor, only provides viseme, word and sentence boundaries in real-time. The animation planning approach cannot handle self-interrupting behaviours such as gaze and head motion, nor their synchronization with other behaviours. To address these issues, we use neural networks to determine timing data in real-time where it is not available a priori.

Finally, within multimodal behaviour realization process, the system needs to respond to events from the environment. We believe that a smart control of behaviour execution needs to be employed in the realization system. It includes animation tracking, stopping, pausing and merging of primitive behaviours. When doing these functions, realism of applied behaviours needs to be preserved.

3.2.2 Behaviour processing and synchronization in RealActor

RealActor is designed to overcome the realization issues we have described in subsection 3.2.1 and drive realistic responsive and interruptive behaviours for ECAs. Since the system is responsible for realization of behaviours send by the behaviour planner (see Figure 2, SAIBA framework), system interruptions are provided through external control functions, which are explained in subsection 3.2.3.

The system is based on animation planning and execution done through applying BML's primitive behaviours to distinct body controllers. The overall realization process is done by three components responsible for (Figure 12):

- parsing of BML scripts
- preparing elements for execution, and
- realizing planned behaviours

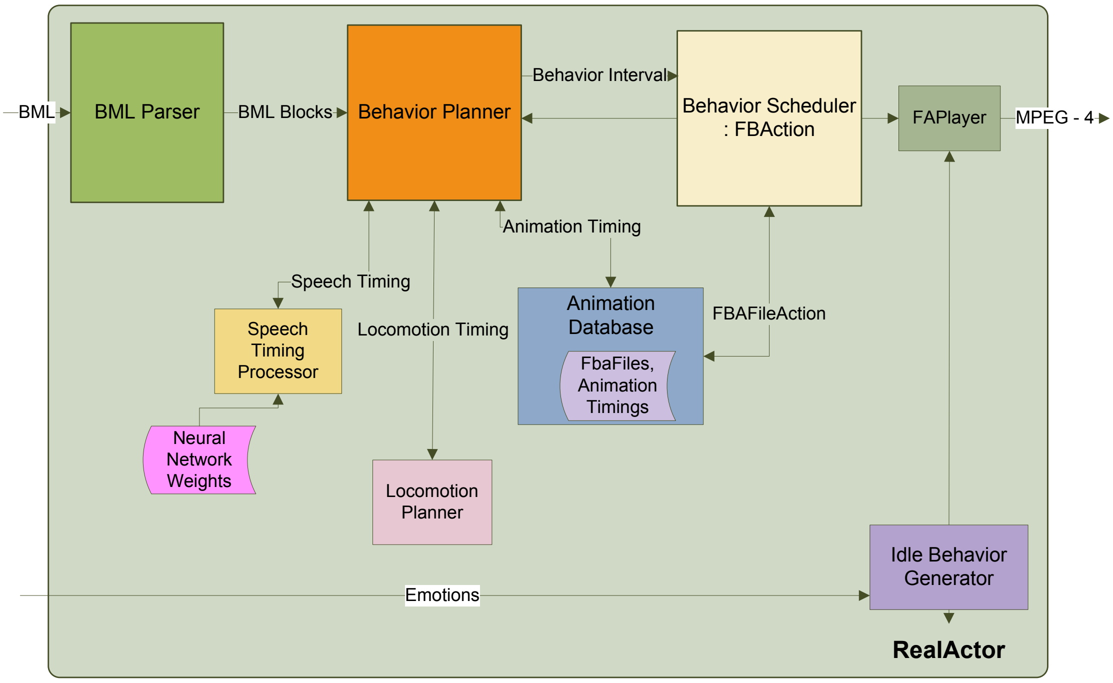


Figure 12: Design of RealActor

The first component, *BML Parser*, is responsible for reading BML scripts and creating objects - BML blocks – containing BML elements with element attributes. BML blocks are added to a list which is then passed to *Behavior Planner*.

Behavior Planner does animation planning. It reads the list of BML blocks and prepares each block for execution by adding timing information needed for behaviour synchronization. The planning is done using a recursive algorithm in which BML elements are processed using relative references to other BML elements and absolute timing information of BML elements which have been prepared for execution. This timing information is retrieved from the *Animation Database*, an animation lexicon which contains a list of all existing animations. In the Database each primitive animation is annotated with time constraints, type information and information about the motion synthesis process (example/procedural). If input BML script contains a walking action, Behavior Planner uses *Locomotion Planner,* a subcomponent which handles character locomotion. It uses the character's current position and specified destination to compute the duration of the walking action.

The character’s speech can be handled in two ways: using lip-sync or text-to-speech synthesis. If lip-sync is used, speech must be recorded in a pre-processing step and manually annotated with necessary timing information, which is then used for animation planning. Text-to-speech synthesis is more practical, because it can be performed concurrently with behaviour realization. However, most TTS systems (including Microsoft SAPI used in RealActor) do not provide a priori phoneme and word timing information necessary for synchronization with nonverbal behaviour. To address this issue we employ machine learning and neural networks, as described in the next subsection. The designed solution is implemented in the *Speech Timing Processor* component.

Finally, prepared BML elements are executed with TTS or lip-sync based speech and related animations (which are nonverbal behaviours) using a hierarchical architecture of animation controllers. The main controller, which we named Behavior Scheduler, uses timing information and time control to decide which behaviours will execute and when. By continually updating the execution phase of each behaviour it is able to synchronize BML elements and provide feedback about progress of behaviour execution. During realization, each BML block is realized sequentially. When one block is finished, the Scheduler notifies the Planner so a new BML block can be passed for execution.

The principle of realizing multimodal behaviours is based on applying BML's primitive units to distinct body controllers, such as head, gesture, face... Behaviour synchronization is handled by the Scheduler, which transmits motions to its low-level subcomponents responsible for realization of primitive behaviours (Figure 13):

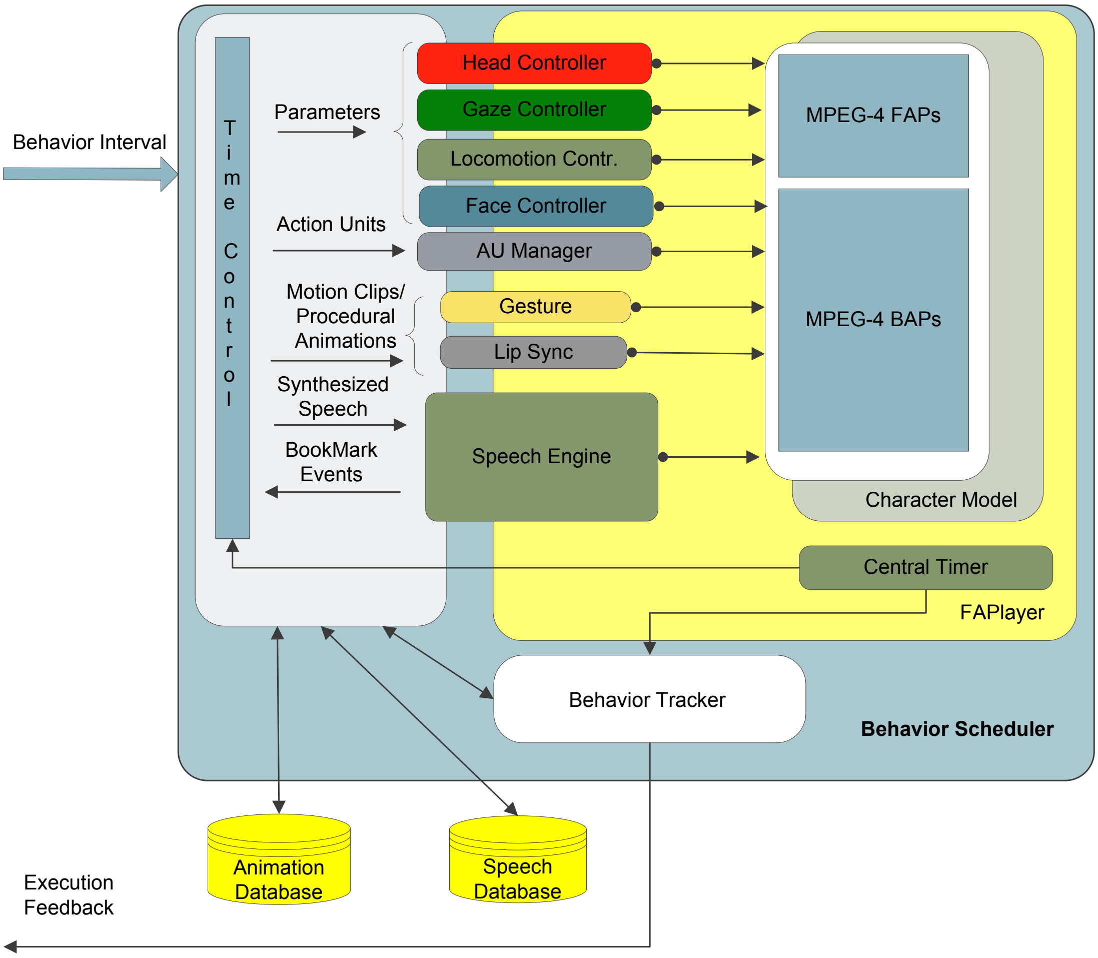


Figure 13: Controller-based architecture for animation execution

- Gazing and head movements are generated by two controllers, Gaze Controller and Head Controller, which run in parallel with the Scheduler. When new gaze direction or head movement occurs, the Scheduler sends to those two controllers parameter values which correspond to the new movement. Animation is executed depending on the current position of the character's head/eyes, target position and duration. The reason we have designed these control units is the nature of head and eyes movement. During conversation, a person can sometimes make gestures that continue after his utterance is finished (meaning the body has not returned to the neutral position). An example is gazing in multiuser communication; person A stops speaking and looks at person B. B then continues the conversation, which is then interrupted by person C. Person A, who was previously looking at person B, gazes at person C. This would mean that person A continues to look at person B after the speaker's utterance is finished. At the time when C begins to speak, a new gaze action occurs. This event directly affects the synthesis of new movements that happen on the same modalities (eyes, head, and body) because in RealActor, both head and gaze motions are designed as procedural animations which are generated on the fly using interpolation steps between the initial and target position.

- To generate facial expressions we use two distinct controllers. The first controller, Face Controller, is compliant with the BML element face which controls mouth, eyes and eyebrows separately. Alternatively, we design a face animation model based on Ekman's Facial Action Coding System. The model is controlled by AU Manager, which runs in parallel with the Scheduler, accepting and running Action Units (AUs). We have chosen Ekman's system because it is widely accepted in psychology and computer science and can be used to synthesize an essentially infinite number of facial expressions. Face animation model is introduced in Chapter 4.
- Locomotion Controller accepts target location and generates a walking action
- Most gestures are generated using motion clips from the animation database containing 52 types of gestures. The clips affect not only hand and arm movements, but also head movements which are blended with the result coming from the Head Controller.
- Speech can be generated either by applying Visage's lip sync system [Zoric and Pandzic, 2006] to audio clips from the Speech database or using text-to-speech synthesis engine. If developers want to use audio clips, they need to manually annotate timings of specific words that need to be aligned with gestures and store timing files in the animation database.

The body controllers are placed in *FAPlayer*, the character animation player in visage|SDK character animation framework. The player is responsible for blending streams of animation data in a meaningful way in order to generate responsive and believable behaviours. The player also contains a time processing unit which is essential for tracking behaviour realization progress and necessary to notify the Behavior Scheduler that it should start or stop execution of primitive units.

To drive realistic behaviours when the virtual character is not animated with BML's behaviours we implement *Idle Behavior Generator,* a unit which generates affective idle motions and sends the animation stream to FAPlayer.

3.2.3 Responses to unexpected events

In addition to the running behaviours the realization system needs to respond to world events, especially those of the participating interlocutors. Besides, an ECA may experience some unexpected events in his 3D virtual surrounding and which may affect its intentions and behaviors. These circumstances require the continuous control and interruption of realized behaviours. Since behaviour realization is a process which refers to realization of physical behaviours that are planned on by other components of ECA system (see 2.1.2 and Behavior Planning on Figure 2), by the term "interruption" we mean control functions and mechanisms which allow our system to immediately react to decisions made by the behaviour planner. For that purpose we design functions which control running behaviours and provide system interruptions and realization feedback. Control functions are implemented inside Behavior Scheduler and are reachable through RealActor API.

To control the behaviour realization progress we implement Behavior Tracker (Figure 13). Tracker is a component which provides realization feedback with syntax fully compatible with BML [BML Specification, feedback]. The component contains data about the realized animations and during realization process it uses timings of running animations and internal timing of the FAPlayer to create realization feedback. The feedback is then passed to the Scheduler and used to remove animations that are finished with execution. The realization feedback is also provided through RealActor API.

Furthermore, RealActor realizes multimodal utterances from BML scripts which are grouped into three different priorities: normal, merged and high priority. Normal behaviours have default priority, which is executed sequentially following the list of BML blocks. Merged behaviours are realized in addition to normal behaviours and can be used as responses to environment events, such as gazing at a person who entered the room during user-ECA interaction. High-priority behaviours interrupt running ones. When they are done executing, RealActor continues to run the interrupted behaviours. Examples of these behaviours are interruptions that do not necessitate behaviour re-planning.

In RealActor we also implement functions for pausing, stopping and deleting behaviours which are running or queued. The functions are implemented using mechanisms from visage|SDK character animation framework and they have purpose to provide the immediate interruptions of the behaviour realization, is so is required by the behaviour planner.

3.3 Neural Networks for Behaviour Alignment

The idea of using machine learning techniques to extract prosodic features from speech is not new. Existing TTS engines often use Bayesian networks, Hidden Markov Models (HMM) and Classification and Regression Trees (CART) to estimate phoneme durations in real-time. However, the drawback of most of these approaches is that they are language-dependent due to their reliance on phonemes. Our approach instead relies on words as input and is applicable to any language.

3.3.1 Training

In RealActor back-propagation neural networks (BNNs) [Rojas, 1996] are used to estimate word duration and align them with animation in real-time. For that purpose, we had to design the layout of our neural network system and train the BNNs with a database containing word-duration pairs in order to achieve the network configuration which yields optimal synchronization results. The main design challenge was definition of the optimal system configuration, such as; how many neural networks will be used, which will be input for those networks, and what will be configuration of those networks.

We specified the following requirements for our system:

- When determining word duration, TTS does word segmentation on phoneme level. Estimated durations of these phonemes depend on their positions inside the word. Segmentation and duration are language-dependent. Since we aim to provide a solution applicable to any language, we decided to use simple segmentation of words into letters and train networks to use correlations between letters to estimate word durations.
- As there are many combinations of letters within words and words have different lengths in letters, multiple BNNs would be defined, specializing in words of different lengths.
- BNNs produce output in 0-1 range. Therefore, normalization would be used to scale this output to word durations.
- Letters' ASCII code would be used as BNN input. Letter capitalization would be ignored.

To satisfy these requirements, we experimented with several system layouts. First, we designed a system of six BNNs, each specializing in words of a particular length. The networks were trained with a database containing 1.646 different words. Training is an iterative process, where letters of each word are encoded and processed by an NN. The output duration was compared with actual word duration and NN weights subsequently adjusted depending on the estimation error. The process continued until the error dropped below the specified threshold. Once the training was complete, we evaluated the resulting BNN configurations using a set of sample sentences and words. The set contained 359 words, which differed from those used in the training process.

While analyzing the results we observed that considerable estimation error occurred for words which were immediately followed by punctuation. This is due to the fact that these words take about 100 ms longer to pronounce than words without punctuation (which we henceforth refer to as *plain words*). To address this issue, we created another experimental layout, which contained 5 pairs of BNNs. Each BNN pair specialized in words of a specific length. However, the first BNN of a pair was trained to handle plain words, while the second one specialized in words followed by punctuation. Upon training and evaluating the BNNs, we found that they were indeed more effective in estimating word durations than the original setup. Therefore, we decided to use the latter approach in our final design. Figure 14 shows one part of the system design with pairs of neural networks processing the plain words and words which were immediately followed by punctuation.

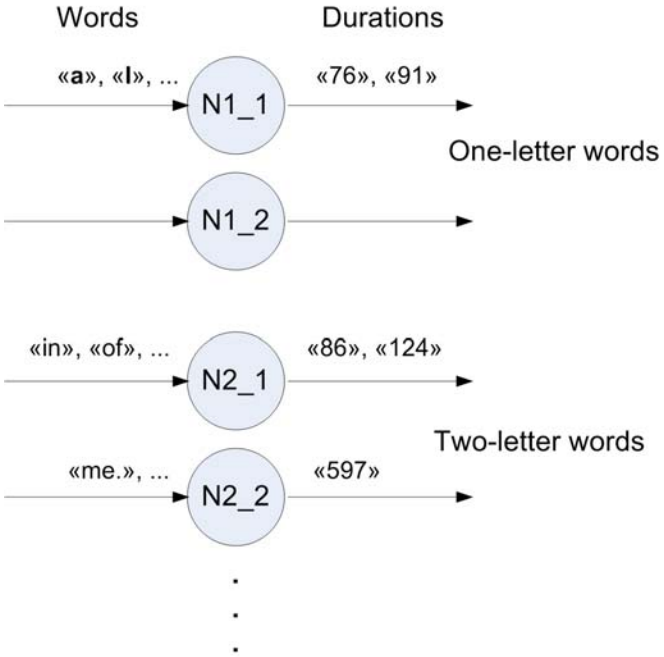


Figure 14: Determining word durations using neural networks

For our final system we prepared 19 pairs of neural networks (NNs) and trained them to estimate word durations. Each of the 16 pairs was trained to handle words of specific length (1-16), like in our experimental design. The remaining three pairs would handle long words of *varying* lengths (16-20 for the first pair, 20-25 for the second pair, 25-35 for the third pair). The words used for training were extracted from a variety of different contexts, including newspaper articles, film dialogue and literary fiction. The resulting database contained 9.541 words of varying lengths and meanings. The longest word encountered had 16 letters, so in the end we discarded the three pairs of NNs intended to handle long words, leaving a total of 32 BNNs in the system.

Final configuration of the neural networks was determined experimentally. Following our previous experiences which have showed that the number of layers does not have much influence on net effectiveness [Zoric et al., 2008], so our neural nets are two-layered nets, containing input and the middle layer. Figure 15 shows a neural network which is estimating duration of seven-letter plain words. The network contains in total 105 nodes in the middle layer which could not be put on the figure.

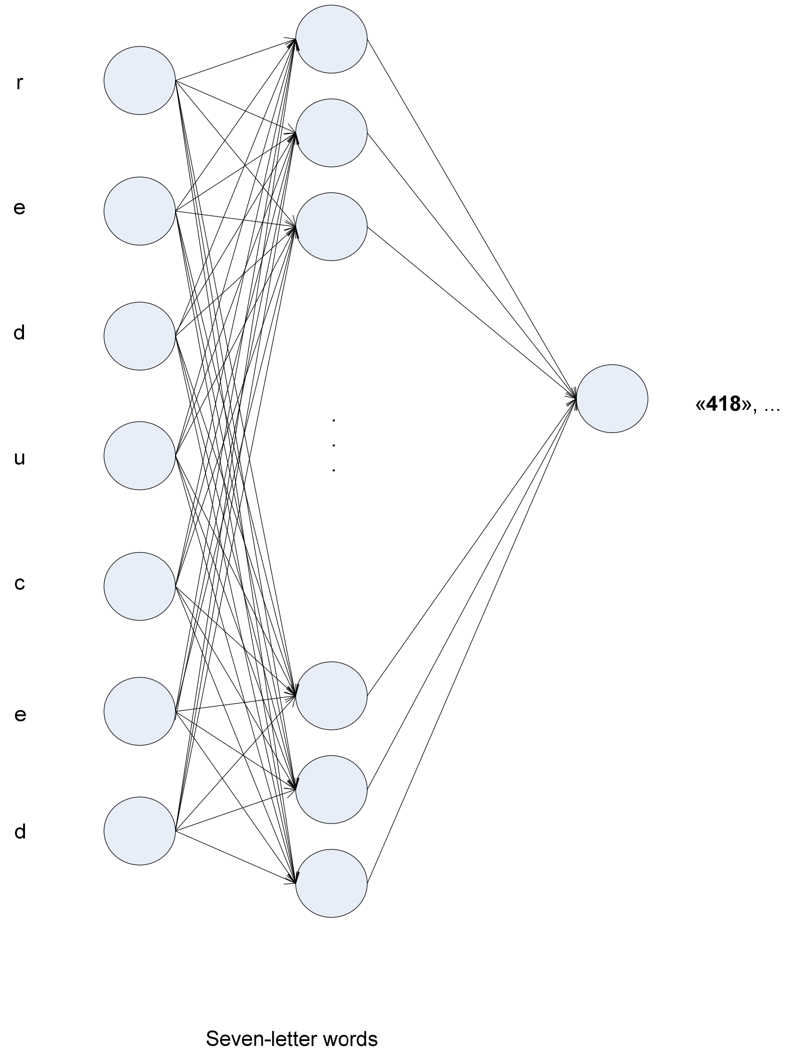


Figure 15: Configuration of neural network which estimates duration of seven-letter words

3.3.2 Evaluation

Once the BNNs were prepared, we evaluated the effectiveness of our method by using the BNNs to estimate the time when an animation should start so that its synchronization point is aligned to a specific word in a sentence. Sample sentences and words used for evaluation differed from those used in the training process. The evaluation was done by summing up the predicted durations of words preceding the synchronization word to compute the predicted time interval before synchronization point and comparing it with the correct time interval to determine estimation error. Time intervals used in the evaluation were 500 ms and 1500 ms, corresponding to shortest and longest start-stroke durations for hand gestures, respectively. During evaluation we also took into account the fact that the human eye can tolerate deviations between speech and image of up to 80 ms [Steinmetz, 1996], meaning that multimodal alignment error under 80 ms is unnoticeable to the viewer. The results of the evaluation are presented in Table 1:

Table 1: Neural network evaluation results

Time interval	500 ms	1500 ms
No. of words handled	310	258
No. of precisely aligned words	245	146
Alignment rate (error under 80 ms)	92,26%	73,26%
Alignment rate (no error)	79,03%	56,59%
Largest deviation	329 ms	463 ms
Smallest deviation	1 ms	2 ms
Average deviation	84.37 ms	126.00 ms

As shown in the table, our system was able to align 92,26% of words for the short time interval and 73,26% of words for the long interval with alignment error not exceeding the 80 ms threshold. Furthermore, the system achieved 79,03% and 56,59% alignment rates with no measurable alignment error.

Though the system uses a text-to-speech engine based on Microsoft SAPI by default, it can be integrated with any TTS engine with minimal effort on the user's part. For multimodal synchronization to work correctly, the neural networks need to be retrained with the new TTS engine, which is a largely automatic process that can take from several hours up to a day, depending on the user's system.

3.4 Idle Behavior Generator

To avoid unnatural motions when the character is standing still we created a component which generates idle behaviours [Egges, 2006]. The component, which we named *Idle Behavior Generator*, accepts emotional states of a character and animates its posture shifts, arm and head motions. These motions are generated even when a character is animated with BML content, i.e. when a character is speaking, gesturing or doing some other actions. In this situation, motions are blended by FAPlayer (Figure 12).

Idle Behavior Generator is not driven by behaviours described by BML. Instead, idle motions are triggered by emotional states of a virtual character (Figure 16).

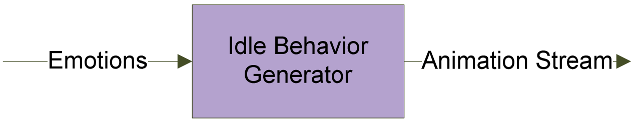


Figure 16: A schematic view of Idle Behavior Generator

We represent emotional state as a pair of numeric values placed inside the evaluation-activation space on the Emotion Disc [Schlosberg, 1954][Ruttkay et al., 2003] (Figure 17). In the Emotion Disc six basic expressions are spread evenly. The neutral expression is represented by the centre of the circle, while distance from the centre of the circle represents the intensity of expression. The advantage of this approach is that different discrete emotions can be placed on the disc, which provides for a possibility to link the animation system with different emotion models, such as the OCC emotions or Ekman’s expressions (Figure 17).

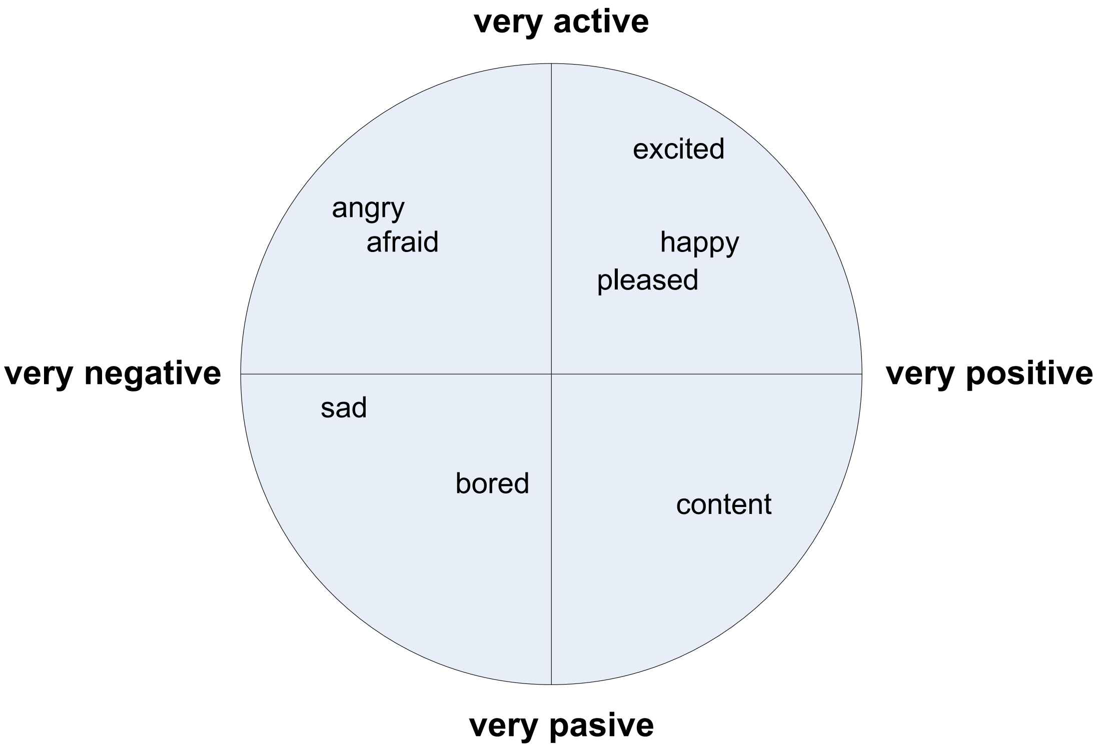


Figure 17: Activation-evaluation emotion disc

The input of Idle Behavior Generator, emotional state, is represented as a pair of numeric values. Those numeric values define a point on the Disc, which may be represented with either 2D coordinates or a vector defined by distance from the centre and its angle. Using those values, the Generator uses a heuristic algorithm to find an animation sequence which will be used to generate idle behaviours. Using 3ds max we have created 4 different animation sequences:

- (1) very passive/negative, characterized with very lowered shoulders, head, and very slow arm and head motions
- (2) passive/negative, characterized with slightly lowered head, posture and slow arm and head motions
- (3) positive/active, we modelled as “normal motions”
- (4) very positive/active, characterized with slightly heightened head, and fast arm and head motions

Each animation sequence is associated with one squared region on the Emotion Disc (Figure 18).

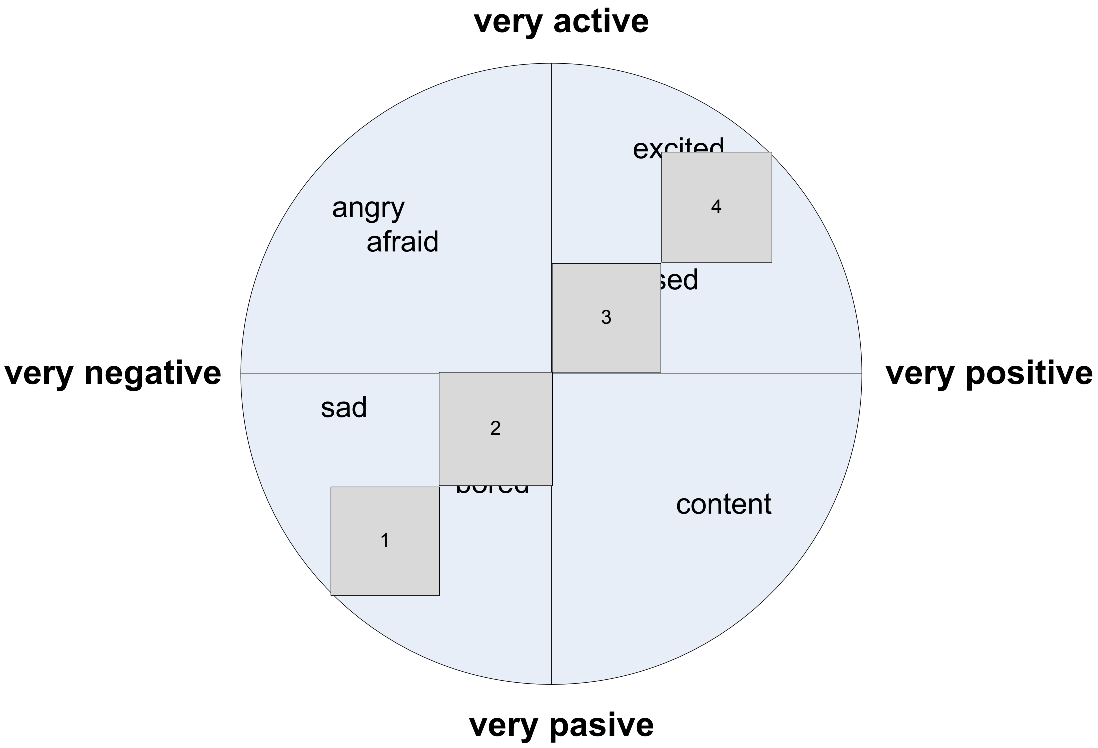


Figure 18: Animation sequences associated with regions on activation-evaluation emotion disc

In the idle motion behaviour generation process, the emotional state represented as a 2D point is mapped to the animation sequence using the following algorithm:

```
(1) Find the distance of 2d point from the each of 4 points that defines a squared region
(2) Find the minimum distance and the matching point
(3) Find the region to which the point maps
```

The selected region is then mapped to the corresponding animation sequence. Since some areas on the disc have not been covered with those regions, we use the minimum distance algorithm to find which animations will be used to generate affective idle behaviours. The final outcome of the generator, the animation sequence, is played in a loop until new emotional state has been specified as system input.

3.5 Animations

RealActor can employ three types of animation to visually realize BML behaviours: motion capture, key-frame and procedural. In absence of motion capture equipment and data necessary to generate communicative nonverbal behaviours, we rely mostly on the latter two animation types. While face and head movements are generated procedurally, most of the hand gestures are key-frame animations which are modelled in 3ds Max, following the findings of gesture study. All animations are organized in a large lexicon of parameterized gestures that should be of great help in modelling highly expressive characters. The lexicon also contains animation timings, which are not directed from BML scripts. We have tried to model and organize animations that will be applied to separate parts of the body with low-level controllers, but it was inevitable to model animations that affect all parts of the body simultaneously through one animation controller. Even BML defines these kinds of BML elements, such as gaze, which controls the position of the eyes, head and spine position simultaneously, even though head and spine position can be individually regulated by BML elements head and posture.

Head motions are implemented procedurally, using the work of Brkic et al. [Brkic et al., 2008]. The work proposes the facial gesture animation functions by fitting a mathematical model to the motion trajectories analyzed from recorded video clips of real, professional TV speakers. These mathematical functions use two attributes (maximum amplitude and duration of the gesture) to generate natural-looking head motion trajectories for head swings and head nods. We adapt these animations to our system, modifying control approach with parameters defined with the BML element head. For example, if BML head parameter “repetition” is set to value “2” for nodding, the animation of nod will repeat twice, the second time with smaller amplitude. Timing for nod of repetition “1”, which is stored in the animation lexicon, will increase by 75% of the initial value when repetition is set to “2”. Additionally, using the approach for head shake trajectories, we model head tilts and head shifts, which are very common for expressing affective states in nonverbal communication.

Face animations are applied through two types of controllers, depending on face parameter definitions in a BML script. If face is controlled through Ekman’s Action

Units (AUs), we use a dynamic face animation model based on FACS, which is introduced in the next chapter. The model also employs animations of facial expressions. For other facial movements defined with the BML face element, such as eyebrows raise or frown, we use simple procedural formulations which move meaningful FAPs. Maximum amplitudes of those FAPs are defined experimentally by performing series of test animations.

Speech can be handled in two ways: using lip-sync or text-to-speech synthesis. In both cases, lip animations are provided automatically – by using phoneme events coming from MS SAPI and their corresponding morph targets (TTS), or by mapping visemes, which are outputs of neural networks in the lip-sync system, to morph targets (lip-sync).

Gaze movements are implemented procedurally, based on theoretical findings in [Leigh and Zee, 2006]. In the work the authors identify several different functional classes of eye movement and derive descriptions which integrate eye movement and head movement during gaze shifts. For RealActor we implement head-eyes saccade movements. Head-eyes saccade movement describes gazing which rotates the eye from its initial position directly to the target, and it supports BML's gaze of type "at" (gaze directly at target). Saccade is very fast eye movement and its velocity is linearly related to amplitude - as the amplitude of saccade increases, so does its velocity and duration. We implement a procedural action which computes duration of gaze shifts on the fly, using initial position of gazing and its target location. During realization, if gaze angle exceeds +/-15 degrees, the head will start to follow. The RealActor system supports four directions of gazing – left, right, up and down.

Locomotion is generated with a function which repeats a single walk cycle until target location is reached. The function implements four important key poses (contact, recoil, passing, and high point) of the cycle which lasts 1 second. Duration of the whole walking sequence is computed in animation planning phase and depends on the initial and target position of the character.

The animation database employs several culturally codified Japanese and Croatian gestures generated procedurally, which are result of our prior work [Huang et al, 2007], whereas almost all gestures are key-frame animations modelled in 3ds max.

While creating hand-gesture animations we referred to lexicons proposed in [Poggi, 2007] and [Posner and Serenari, Blag], existing ECA systems [Neff et al., 2008], literature on human gesturing [Armstrong, 2003] and even comics such as the popular Asterix which provide an interesting repertoire of gestures. While selecting gestures that we modelled, we have tried to choose universal ones which are not culturally coded and which can be used in ECA systems regardless of their application. Thus, we chose a lot of beat gestures, which are small movements associated with speech and symbolic gestures which could be recognized in most of the cultures. For each gesture type we modelled variations based on dynamics and spatial characteristics.

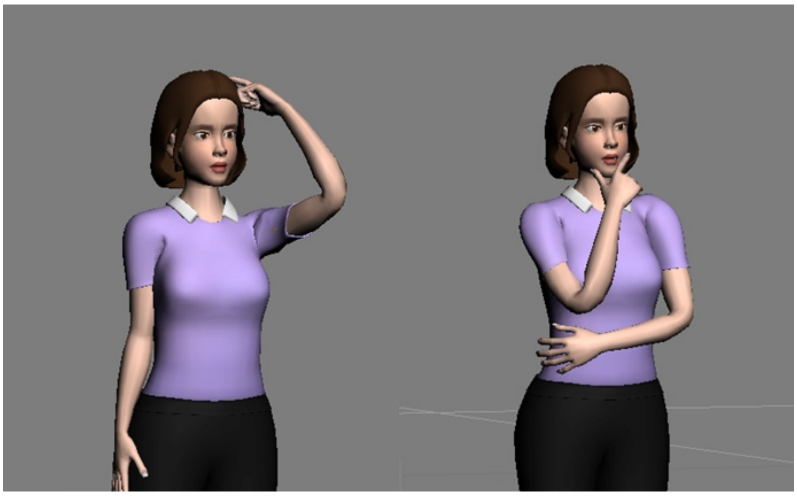


Figure 19: Variations of the symbolic “thinking” gesture

The system implements 52 different gestures: beats, lexemes, conduit and generic gestures, which are complex symbolic animations controlled through external namespace elements in a BML script (Figure 19). Whereas all animations are parameterized with corresponding attribute values of BML elements, the parameterization of gesture extends the BML attribute specification and is based on the work of Poggi [Poggi, 2007]. To provide simple control, we merged BML location attributes into a single vector attribute and add an attribute specifying which part of the head or arm is affected by the gesture. The parameterization also includes emblems, gestures which can be used interchangeably with specific words [Ekman, 1979]. In total, our animation parameterization has four dimensions:

- general - gesture type and literal meaning

- space-shape - hand shape and location at stroke-start point
- dynamics - intensity, amplitude, direction and movement trajectory during stroke phase
- timing information – specification of synchronization points for the BML scheduling component

Our animation lexicon also includes animations for idle behaviour to avoid characters standing perfectly still while inactive, and which are already introduced as key-frame animations, which are implemented in 3ds max and exported to FBA files.

To summarize, the system fully implements the following BML behaviours:

- face expressions (joy, sad, angry, fear, surprise)
- eyebrow, mouth and eyelid motion specified by the BML element "face"
- gaze at a specific target
- head nods, shakes, tilts and orientations specified by the BML element "head"
- 52 different types of gestures varying from beats to complex gestures which cannot be described with BML parameters (e.g. scratch head, cross arms)
- walking
- speech – synthesized from text or loaded from an annotated audio file

3.6 Features and Implementation

RealActor is implemented as part of visage|SDK character animation framework and has the following features (some of them are inherited from visage|SDK):

- specification of character behaviours using BML scripts
- start, stop, schedule or merge behaviours via high-level, BML-compliant API
- database of annotated animations which can be shared between multiple characters
- animation API
- visual text-to-speech synthesis based on industry-standard Microsoft SAPI
- lip synchronization
- supports animation using FACS
- supports integration with any engine or application via a minimal scene wrapper

RealActor has a layered architecture (Figure 20). At the top of the system sits the BML realizer, tasked with parsing BML scripts, planning, scheduling and executing behaviours. Its functionality is described in detail in the next section. Below it is the motion playback system. Based on MPEG-4 FBA, it plays and blends animations that correspond to various elementary behaviours. Finally, the bottom layer is the scene wrapper, which provides a common minimal interface for accessing and updating the underlying character model. This interface needs to be re-implemented for different rendering engines.

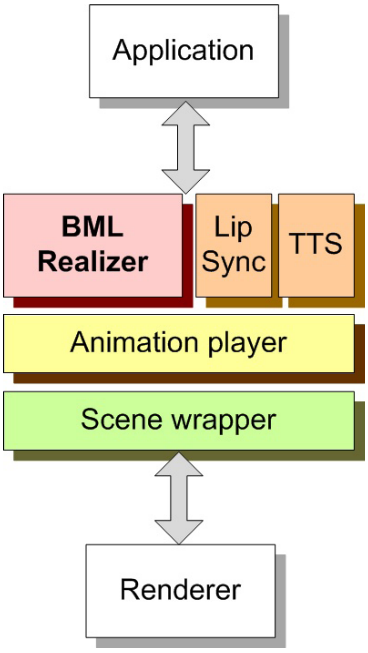


Figure 20: RealActor system architecture - simplified view

RealActor is implemented in Visual Studio 2005 in C++ using visage|SDK toolkit for character animation and its internal support to Microsoft Speech API [MSAPI] for speech synthesis.

We implement a class hierarchy which includes BML elements, blocks and internal processing functions. These are functions for parsing BML elements (implemented using MSXML library) and planning animation timings using recursive functions, algorithms and information retrieved from the animation lexicon, which are stored in Microsoft excel (XLS) files. Neural networks are designed by adapting the open-source code from the site [NN at your fingertips]. We also create a stand-alone application for training the neural networks which consists of two steps. While the first step retrieves data from the TTS engine that is required for training, the second step trains the nets, and when a certain threshold is reached, it stops the training process and saves data for RealActor.

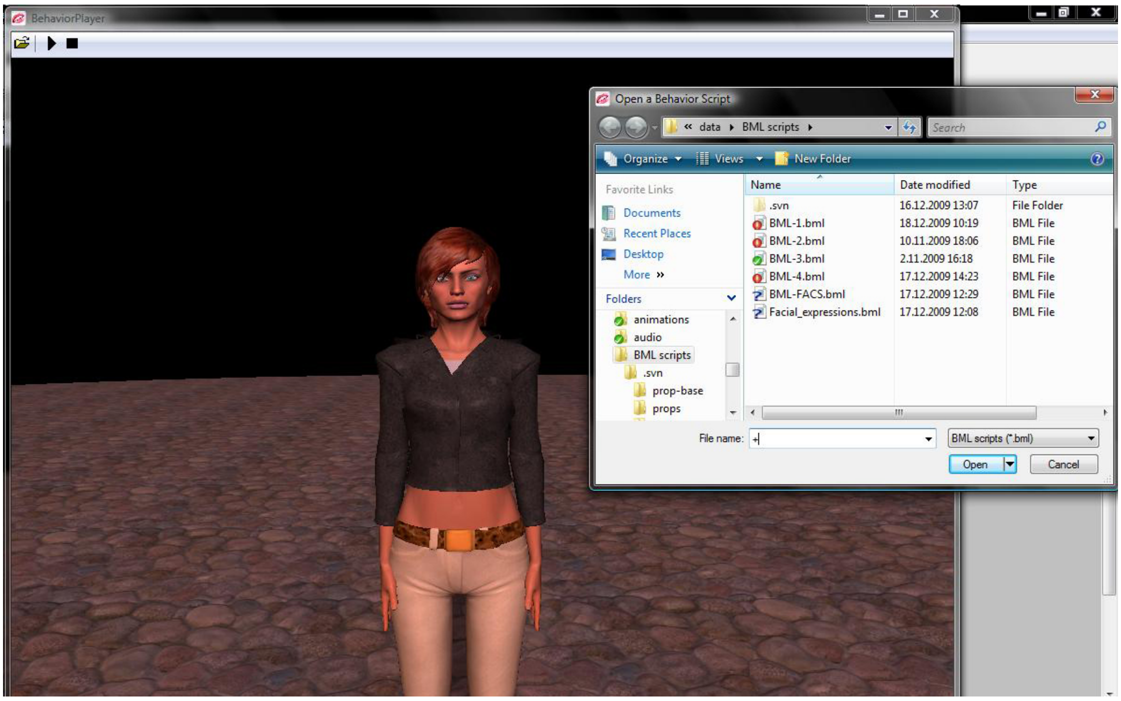
RealActor's animation mechanism uses Visage's central player, FAPlayer, which holds the model data and running animations and which implements the main thread responsible for generating the animation stream. FAPlayer is initialized by loading the virtual character as VisageCharModel, a class that aggregates model data important for character animation (skeleton, a list of all meshes, mappings of MPEG-4 face and body animation parameters (FBAPs) to skeletal joints and morph targets). After FAPlayer has been initialized, one or more animation tracks, which may be loaded from files or implemented procedurally, are added, and animation is started by calling the play method. The main thread's animation mechanism works by continually updating and mixing FBA streams (interpolation) coming from these animation tracks.

Behavior Scheduler is implemented as the main animation track for FAPlayer, an instance of FbaAction whose main function is called every 40 milliseconds. The Scheduler keeps references to low level controllers; it starts and stops different kinds of animations which are directed by a BML script or external control functions. For each animation frame, a central loop of Behavior Scheduler updates low-level controllers with the animations and removes those animations if they are finished with execution. It also keeps a reference to Behavior Planner to communicate realization progress.

RealActor is implemented as a static library and part of Visage|SDK. It provides an API which consists of functions which load the character model, read and realize BML scripts or accept them as a string, and virtual feedback functions which are called by Behavior Scheduler during the realization progress. Moreover, RealActor provides an animation API with parameterized BML elements. A developer can implement their own animation scheduler and use these functions to animate conversational behaviours for the virtual character.

3.7 Testing

To test RealActor we have built an application using the application framework for Horde3D engine from Visage. The application (which we call Behavior Player) is depicted in Figure 21. It renders the virtual environment and virtual character in neutral position, and an upper tab with controls which invoke RealActor functions. “Play” button opens a file dialog for choosing a BML script file to read and execute. “Stop” button stops running behaviours. BML scripts are saved as files with *.bml extension and can be edited using a simple text editor.



Name	Date modified	Type
.svn	16.12.2009 13:07	File Folder
BML-1.bml	18.12.2009 10:19	BML File
BML-2.bml	10.11.2009 18:06	BML File
BML-3.bml	2.11.2009 16:18	BML File
BML-4.bml	17.12.2009 14:23	BML File
BML-FACS.bml	17.12.2009 12:29	BML File
Facial_expressions.bml	17.12.2009 12:08	BML File

File name:
BML scripts (*.bml)

Figure 21: Interface of a test application for executing BML scripts

We have tested Behavior Player using two categories of BML scripts. The first one contains simple test units, which are primitive BML elements which contain different attribute values and have various sub-relations. For more serious testing, we have created the second category, which are several examples of BML scripts which contain various nonverbal behaviours in addition to speech. The snapshots of the execution are given in Figure 22. First row shows facial, gaze and head movements executed using BML’s gaze and head elements, and face Action Units generated with Face Animation model which we introduce in the next chapter. Second row depicts results of hand motion examples; beat gesture and lexeme gesture, which is aligned with the word “calm”.

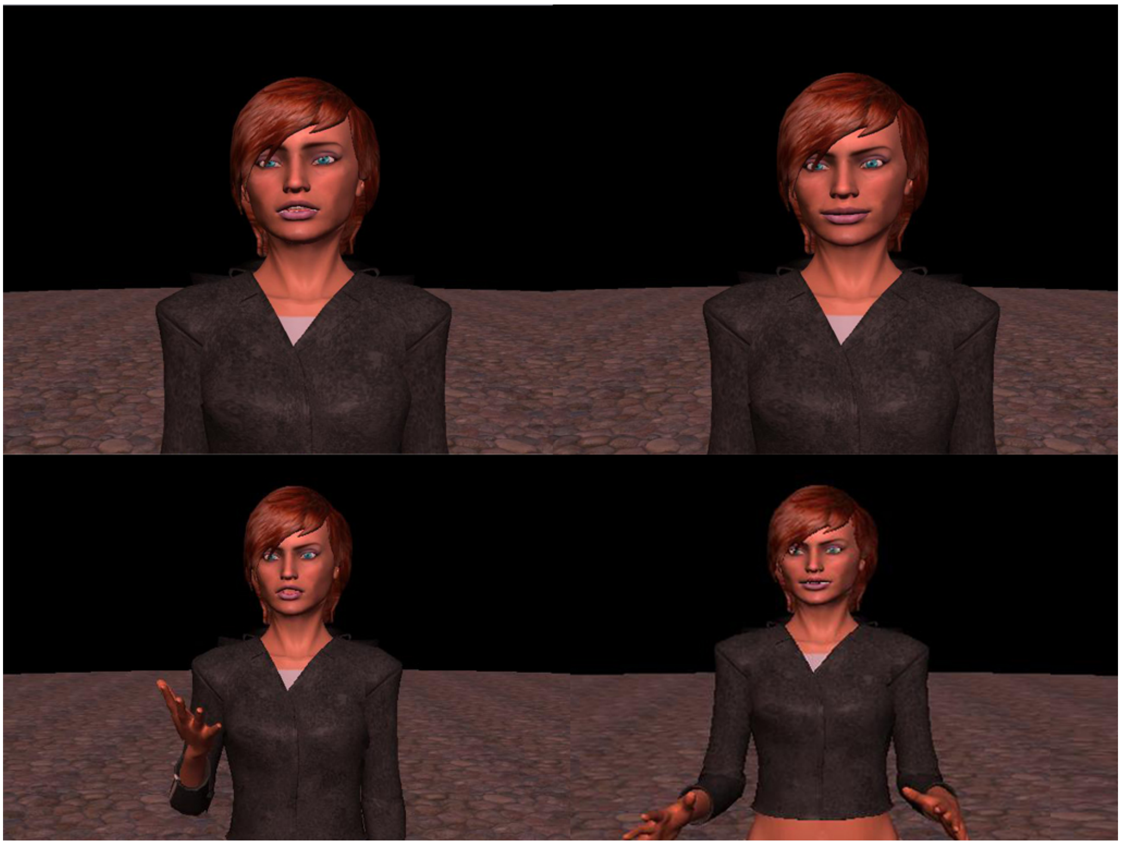


Figure 22: Snapshots of behaviour realization with RealActor

Behavior Player is also tested in a preliminary research which aims to re-create affective faces of real human speakers using FACS coding system. We have annotated video [Anvil] showing an angry person and then created a script using annotated elements, which are translated into BML elements. The script contains definitions of Action Units, head movements and references to the audio of the spoken utterance. It contains in total 1693 Action Units and 129 head movements which accompany 96 seconds of speech.

Although animation results are good, drawbacks of BML itself make our system unable to accurately reproduce annotated video. Annotation file specifies start and end point of each action, such as action unit or head movement, which is not defined in BML. For that purpose we have used absolute timings of Action Units, which are introduced in the next Chapter. This research is in an early stage, but it shows that

RealActor can serve to identify important features for creating affective ECAs and coding various expressions which may serve as data in behaviour generation process in ECA system.

3.8 Conclusion and Discussion

In this chapter we discuss multimodal synchronization issues for ECAs and present RealActor, an open-source character animation and behaviour system with support for modelling multimodal communication. The system is based on Behavior Markup Language (BML) and is freely available for exchange within the research community. We propose a universal, language- and engine-independent solution to the issue of synchronizing verbal and nonverbal expression by using neural networks to estimate word durations. Our system is designed in such a manner that it can be integrated with existing engines and application frameworks with minimal effort. In addition, it can be easily extended with new animations or motion synthesis approaches.

We have built an example application and tested the system with a variety of BML behaviour scripts. The final results seem very satisfactory; however, we have noticed some disadvantages of the system which mostly come from the animation side:

- Current animation of walking does not take into account collision-avoidance and interruptions in the walking cycle. Since the field of character locomotion research is vast, we have not addressed it in our thesis, but we propose it in future studies for improving RealActor's animations.
- Idle motion generator has a great impact on increasing the level of realism in the generated behaviours. However, animation sequences of affective idle behaviours are constructed by beginners in 3d animation. Hiring professional animators to build a greater database of idle motions (e.g. 8 sequences of animations for 8 different regions on the Emotions Disc) would undoubtedly have a great positive effect on visual appearance of the displayed character.
- In the system we provide an animation database which may be shared between virtual characters. However, there are about 20 modelled gesture

animations which may suffer from limb collision if applied to another character. Examples are scratching the head or clapping. For this purpose, motion retargeting techniques should be employed by RealActor system.

- We have noticed that postural shifts, such as leaning forward and back, or shape of the posture would greatly increase the impression of affective faces. Relationship of postural shifts and affective states has been also been proven in literature [Bianchi-Berthouze and Kleinsmith, 2003], therefore realistic animations of posture shapes would greatly improve the system's features.

Finally, while testing RealActor, we also observed that manual authoring of scripts can be a time-consuming and counter-intuitive process that requires a strong understanding of the BML specification and constant animation lexicon look-ups. One idea is to develop a graphical BML tool for more intuitive script authoring, while another is to provide a higher-level component for automatic behaviour generation. Considerable research has been done on methods of automatically deriving multimodal behaviours from text and our approach would build upon work presented in [Lee and Marsella, 2006] or an earlier example, the BEAT system [Cassell et al, 2001].

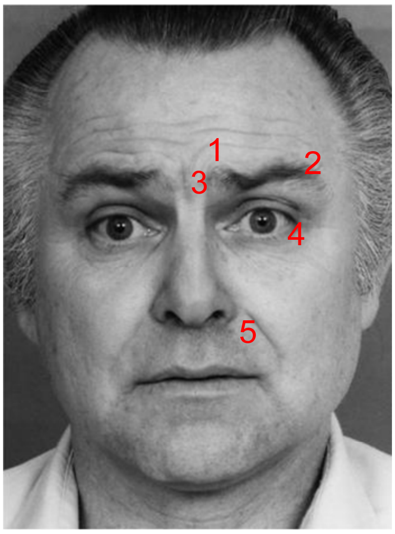
4 Dynamic Face Animation Model

This chapter is dedicated to the face animation model based on Ekman's Facial Action Coding System (FACS) [Ekman and Friesen, 1978]. The face animation model is integrated into RealActor and controlled by the BML element "face" and its attribute values. Alternatively, instead of (or in addition to) BML element timing references, developers can specify animation timings using absolute duration of each animation phase.

At the beginning of this chapter we are going to present FACS and explain the reason why it is widely used in computer graphics. In further reading, we are going to describe how we modelled animations of Action Units (AUs) using results of tracking videos of professional actors. The rest of this chapter presents evaluation and facial expression examples.

4.1 Facial Action Coding System (FACS)

Facial Action Coding System (FACS) is a common objective coding standard developed to systematically categorize and describe the richness and complexity of facial expressions. FACS defines in total 46 different Action Units (AUs), which are contractions or relaxations of one or more facial muscles, including head and eye movements. Using FACS a trained human expert can detect and decode any facial expression (Figure 23). Usually, it takes approximately 100 hours of self-instructional training for a person to make FACS judgments reliably with reliability determined by a standardized test. Once trained, it typically takes three hours to code one minute of videotaped behaviour. Similarly to decoding facial expressions using FACS, any facial expression can be synthesized as a combination of Action Units. This allows for enormous potential for facial expression animation.



Scores

1. 1B: Inner Brows Raise
2. 2C: Outer Brows Raise
3. 4C: Brow Lowerer
4. 5A: Upper Lid Raise
5. V11C: Nasolabial Furrow Deepener

Figure 23: Example of a facial expression decoded by FACS

While FACS has been used to verify the physiological presence of emotion in a number of studies (e.g. in the analysis of depression [Reed et al., 2007]), in last ten years it has also proven to be very useful for character animation. For example, FACS was used in the movies The Curious Case of Benjamin Button (2008) and Monster House (2006). The first movie enrolled real human actors who played the story characters. The main character was played by several actors and one digital head, which was created to replace the heads of the real actors who played the main character while he was of very old age. Monster House is an animated movie whose characters' motions are driven completely by motion capture data from real human actors. In both movies the approach for FACS-based synthesis is the same. One CG digital head is animated using a database containing animation sequences of Action Units collected by the same actor. The examples of compelling facial expressions are shown in Figure 24. Benjamin Button movie was awarded an Oscar for special effects.

Figure 24: Examples of faces animated with Facial Action Coding System. (left) Characters from animated movie Monster House (right) The Curious Case of Benjamin Button showing digital head placed on the body of another actor

4.2 Face Animation Modeling

In RealActor we employ facial expression synthesis using FACS. We rely on Visage's MPEG-4 FBA feature points and its morph targets and use groups of FAPs to implement each Action Unit. We model each AU as a procedural animation which controls the specific group of FAPS over a certain amount of time. Dynamics of each AU depends on its duration and intensity. During behaviour realization process in RealActor a central unit named AU Manager coordinates execution of several AUs using weighted blending and linear interpolation.

Each *Action Unit* is based on the Attack-Decay-Sustain-Release model (ADSR) [Dariouch et al., 2004], which specifies animation of linear facial features points though the following realization phases: *attack, sustain, decay, release*. While BML defines behaviour timings relatively using references to other elements, in our model a developer can alternatively specify duration of each phase absolutely. There are two reasons for this. First, the dynamics of face movement are highly important [Wehrle et al., 2000] and this approach provides greater control than BML itself. Second, using explicit specification of phase durations, videos showing facial expressions can be easily re-created (e.g. videos which are decoded with annotation tools, such as Anvil [Kipp, 2001])). Accordingly, we propose an additional face attribute for the BML face element which controls timings of Action Units.

```
<face id =”f1” type=”AU1” start=”s1: start” intensity=”A”
au_duration=”400;500;800”/>
```

The above example defines a BML element with id “f1”. F1 defines Action Unit 1 (which is inner eyebrow raise) of very low intensity defined with value “A” in *intensity* attribute (FACS defines 5 ranges of intensities of each AU, going from A to E). To control its dynamics we use *au_duration* attribute, which specifies that it performs very slowly, since AU1 amplitude is very low. The unit lasts 500 milliseconds and it takes 800 milliseconds until facial appearance goes back to neutral position, if there are no other facial actions. If not specified, default durations for attack, sustain, decay phases are 400, 1200, 350 respectively for all Action Units.

4.2.1 Tracking and Analyzing Action Units

A single Action Unit is modelled by analyzing results of face tracking. The tracker that we use has been developed at Linköping University [Ingemars, 2007] and comes as part of visage|SDK framework. It applies a manually adaptable textured Candide 3D model [Albrecht, 2001] to the actor's face and then tracks a set of significant feature points and extracts global parameters - rotation and translation – as well as dynamic facial parameters (expressions) for each frame of the tracked video. It tracks feature points using motion between frames and Candide model, and employs an extended Kalman filter to estimate the parameters from the tracked feature points.

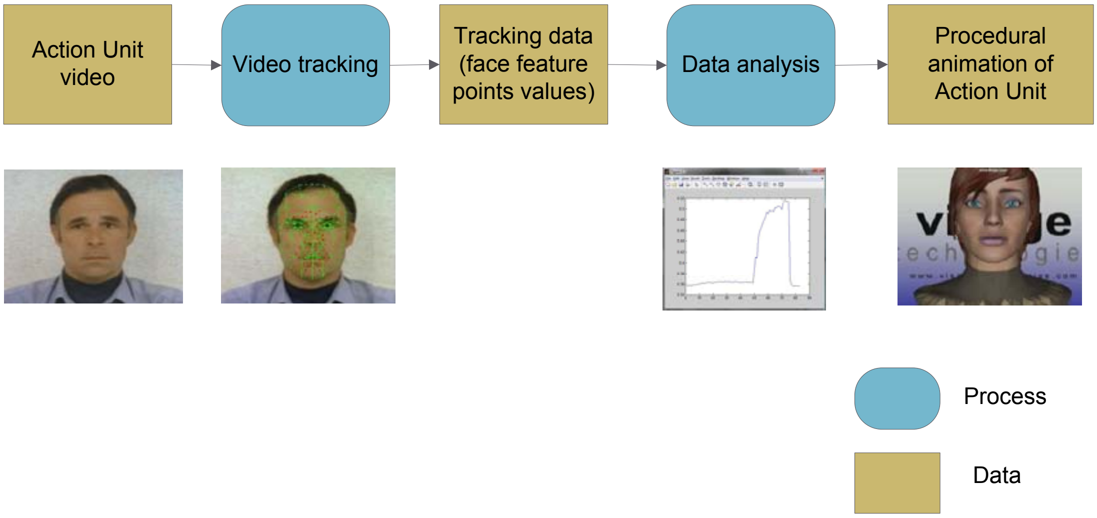

Action Unit video
Video tracking
Tracking data (face feature points values)
Data analysis
Procedural animation of Action Unit
Process
Data

Figure 25: Action Unit modelling

For Action Unit modelling we adapt the face tracker to obtain the results of videos that accompany the book “Facial Action Coding System” [Ekman and Friesen, 1978]. The idea is to analyze how movements of specific feature points occur and apply the findings to MPEG-4 FAPS (Figure 25). For that purpose, we have modified the tracker’s source code to save the global parameters of specific feature points in given files. We separately tracked each video showing a professional actor perform a single AU of certain intensity. First, texture of the Candide model with feature points (FPs) is applied to the actor’s face (Figure 26), and when the tracking is done, results are saved into a file. In that way we have saved results of videos of 34 different Action Units. Unfortunately, these videos display only the attack phase of the Action Unit.

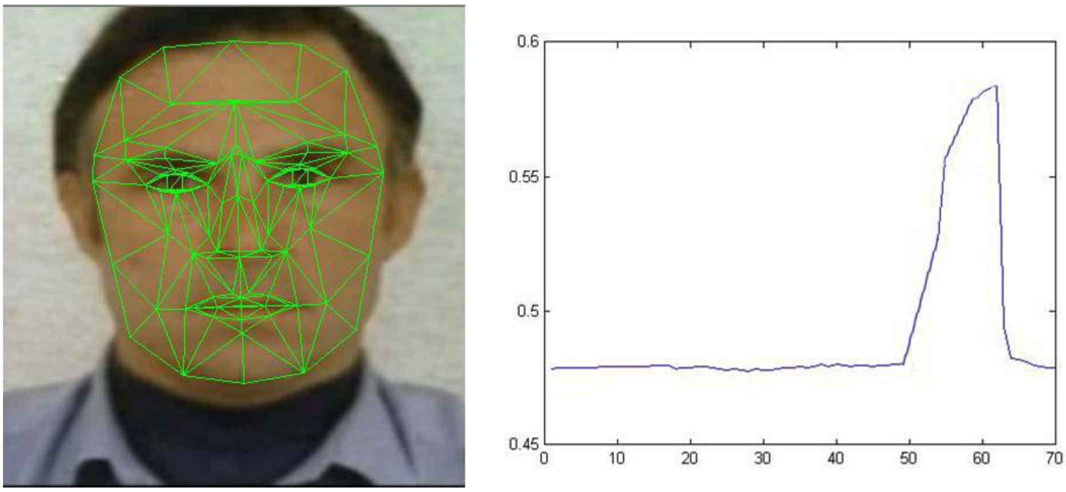


Figure 26: (left) Face tracking system with adaptive texture of Candide model. (right) A graph which shows movements of a left inner brow in 2D x-y plane.

Values of global parameters of specific feature points over time are then displayed using MATLAB simulation tool [Matlab]. Each graph displays the dynamics of movement of one feature point in 2D plane (usually X-Y space). For each Action Unit we create up to four graphs, depending on how complex the Action Unit is and what kind of movements it encompasses. For example, the lip movements move a group of several feature points (MPEG-4 FBA defines 12 FAPs that affect the lips) and to model the AU which affects lips we create and analyze four graphs.

With the help of MATLAB toolbox for linear curve fitting we then approximate the graphs with mathematical functions. This process is done manually. After viewing each graph, we estimate whether the movement will be approximated with linear functions, which is done using manual calculation, or we will use functions from MATLAB toolbox (functions *fit, fittype*, *fitoptions*). The first approach is chosen if a graph can be approximated with a straight line. Each approximation is then applied to a specific MPEG-4 FAP. The maximum FAP amplitude for each modelled AU is defined experimentally. While FACS defines five levels of intensity for each AU, in our model amplitude of FAPs is set linearly according to the maximum value and controlled intensity of a specific AU.

For example, we explain how we model AU1, described as “inner brow movement”. Figure 26 depicts how the feature point of left inner brow moves in a tracked video that shows an actor performing AU1. We approximate this function with the following equation (combination of linear and polynomial function):

$$0 < t < attack/2 \quad f(t) = \frac{2}{3} * A_{max} * t$$

$$\frac{attack}{2} < t < attack \quad f(t) = \frac{-2}{3 * attack^2} A_{max} * t^2 + \frac{4}{3 * attack} A_{max} * t$$

The attack phase is divided into two subphases; linear and polynomial functions affect the final position of the feature point depending on the realization time. The function is then applied to FAPs *raise_r_i_eyebrow* and *raise_l_i_eyebrow* in a procedural action which implements AU1. Maximum value of amplitude *A* is defined after series of experimental movements of FAPs *raise_r_i_eyebrow* and *raise_l_i_eyebrow* and is set to 200. Sustain phase keeps feature point values the same, while decay phase uses linear fall to the neutral face position.

If a developer wants an ECA to realize AU1, they can control the dynamics of execution using *intensity* of AU1 and duration of *attack, sustain* and *decay* phases. Default intensity is the smallest one - “A” - and for AU1 its value is 40. Default duration for attack, sustain, decay phases is 450, 800, 300 respectively for all Action Units.

4.2.2 Constructing the Face Animation Model

Using the modified face tracking system and results of analysis 34 Action Units were modelled in total. Most of the analyzed movements could be described with linear and polynomial functions. There were some graphs with spikes that probably come from bad video quality and these spikes were not taken into account in the approximation. In AUs which affect mouth movement, only key feature points (lip corners and lip top and bottom) were analyzed and applied to MPEG-4 parameters. Since the tracker we used does not track eye openness, we modelled AUs which affect the eyes manually. Head movements were not analyzed since RealActor implements a head controller with head movements modelled using a similar approach [Brkic et al., 2008].

After animation implementation, modelled Action Units were compared to original video examples. Subjective evaluation showed good results, however limitations of our face model and MPEG-4 FBA standard for some AUs gives animations which cannot be the same as in the original video. First, our face model has no wrinkles. Second, some AUs describe face movements that cannot be animated using MPEG-4 FAPs. Examples are: lip points on Z plane, the movement significant for AU29; inner lip points on X plane, the movement significant for AU30, and nose on Y plane, the movement significant for AU9.

In real time facial motion synthesis, modelled AUs need to be coordinated if two same AUs happen simultaneously or two AUs affect the same FAPs. Therefore we design *AUManager,* which is a central face animation controller for blending Action Units. For each frame of animation the blender iterates through running Action Units to determine the final value of each FAP. The final value is chosen depending on how many (same or other type) AUs affect the same FAP using weighted blending. This creates an effect of natural transitioning between Unit phases and continuous repeats of AU. For example, when AU1 is in *decay* phase and new AU1 appears, the results of *decay* and *attack* phases will be blended, second AU1 will continue without moving the brows back to neutral position. Visually, the ECA raises his/her inner eyebrows, and just as they start falling down they are raised back again smoothly.

4.3 Evaluation

After designing the facial animation model, we conducted an empirical study to figure out its effectiveness. The study consists of two cases:

1. Imitation of facial movements.
2. Emotion synthesis and recognition.

For both studies we recruited 28 subjects, 13 males and 15 females, ages between 23 and 35, none of who have any experience in character animation and emotion recognition. All subjects have higher education. The study included a subjective judgment of animation quality and recognition of synthesized face expressions.

4.3.1 Case Study 1

In the first study we wanted to estimate the accuracy of our face model. In an ideal situation we would recruit trained FACS coders to decode designed animations. Because of the lack of enough representatives, we chose the following methodology:

- We manually annotated 6 videos of a human who is performing combinations of different AUs. Each video lasted 2 seconds.
- Using time annotation to drive the face animation model we designed 6 canned animations of an ECA who is imitating the human.
- Pairs of videos were shown to human subjects. They had to report perceived quality of "imitation", meaning how well ECA's facial movements emulated the facial movements of the human. Videos were rated on a five-point Likert scale (0 – 4).

The annotation videos were selected to show as many different AUs as possible. Evaluation results are shown in

Table 2.

Video 1, which shows a combination of eyebrows movements and video 6, which shows lip corner stretching and falling were perceived as the best quality videos. Second video, which shows a subject wrinkling with the inner part of his eyebrows (AU4) and dilating his pupils (AU5) has a weak average rating. We suspect it is because participants were more affected by the wrinkles than by pupil dilation. Regarding the third video, combination of AU10 and AU12 in the original video makes a quadratic shape of the mouth. Using MPEG-4 FAPs we could not achieve that effect in animation.

Table 2: Case study 1 evaluation results

Combination of AU	Average Quality (max 4)	Standard deviation
Video pair 1 (AU1 + AU2 + AU4)	3	0,902
Video pair 2 (AU4 + AU5)	2,07	1,184
Video pair 3 (AU10 + AU12 + AU6)	2,11	0,916
Video pair 4 (AU12 + AU27)	2,89	0,916
Video pair 5 (AU14 + AU23)	2,64	1,311
Video pair 6 (AU15 + AU17)	3,17	0,819

4.3.2 Case Study 2

For the second study we constructed animations of basic emotions with FACS according to the theoretical specifications [Ekman and Friesen, 1978] (Table 3). In total we made 8 canned animations, six basic emotions and two repeats (disgust and sadness appear twice, but with different expression). Examples are shown in Figure 27. Each facial expression is scripted with BML and is further provided in RealActor script database.

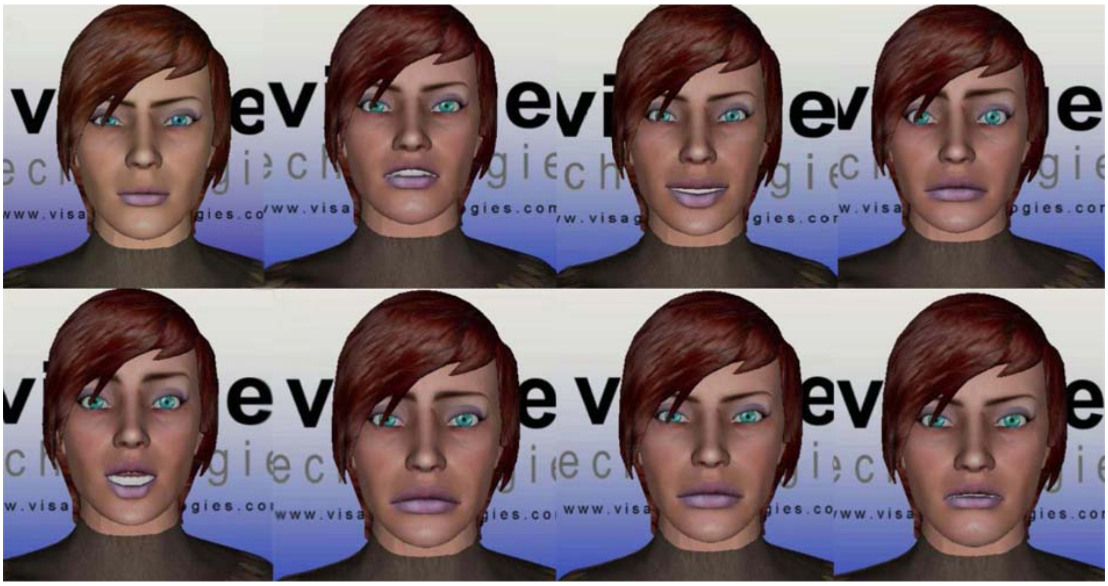


Figure 27: Facial expressions shown to evaluation subjects: anger, disgust1, joy, fear, surprise, sadness1, sadness2 and disgust2

Table 3: Combination of Action Units used to construct facial expressions

Animation	Emotion	Combination of AUs
1	Anger	4, 5, 7 and 27
2	Disgust	10 and 17
3	Joy	6 and 12
4	Fear	1, 2, 5, 4 and 15
5	Surprise	1, 2, 5 and 26
6	Sadness	1, 4, 11 and 15
7	Sadness	6 and 15
8	Disgust	11 and 17

Animations were shown to evaluation subjects. After viewing each video, they had to report perceived emotions. They could choose one of the following replies: anger, disgust, joy, fear, surprise, sadness and N/S (not sure). Evaluation results are shown in Table 4.

Table 4: Case study 2 evaluation results

Animations	Anger	Disgust	Joy	Fear	Surprise	Sadness	Not sure
Anger	**0,32**	0,07	0	0,07	0	0,20	0,32
Disgust1	0,29	**0,57**	0	0	0,04	0	0,10
Joy	0	0	**1**	0	0	0	0
Fear	0	0	0	**0,29**	0,46	0,21	0,04
Surprise	0	0	0	0,07	**0,89**	0	0,04
Sadness1	0	0,1	0	0	0	**0,79**	0,11
Sadness2	0	0,11	0	0,11	0	**0,57**	0,2
Disgust2	0,25	**0,46**	0,04	0,07	0	0,18	0

When discussing the evaluation results, we have to take into account cross-cultural variation of identifying emotions on a facial expression. Neuroscience findings say that the observation of higher recognition accuracy rates in cases where both the decoder and the encoder belong to the same cultural group (e.g. [Elfenbein and Ambady, 2002]). In our case, codes for facial expressions matching six basic emotions are defined by a psychologist from American culture (Ekman), while the evaluation subjects (decoders) are Croatian. Although Croatian and American cultural norms are similar, we may still assume that cross-cultural variations affected the evaluation results. This fact may be proved by investigating neuroscience findings, but this is not scope of the thesis. The goal of evaluation study was to determine whether our face animation model may be used in generic ECA system to synthesize believable facial expressions of emotions.

Study results (Table 4) have shown that joy, surprise, sadness were the best perceived emotions, which is similar to results of the work done by Gosselin and Kiroac [Gosselin and Kiroac, 1995]. Disgust and fear were perceived badly, as well as anger, which was also reported as sadness. We observed that fear was mistaken for surprise, which is a well-known phenomenon in psychology [Ekman, 1973].

In summary, the animations designed for joy, surprise, and sadness were well recognized. The results have shown that we have to improve animations of anger, fear and disgust. Weak perception of these emotions could be also because the evaluation subjects were confused by the fact that the same emotion may repeat in shown videos (which are in our case, emotions of disgust and sadness).

4.4 Conclusion and Discussion

In this section we have explained the way we implemented the facial animation model and integrated it into RealActor. The animation model applies Action Units from Facial Action Coding System (FACS) to MPEG-4 face animation model. While realizing animations, the animation model blends several action units and controls values of MPEG-4 FAPs depending on the realization phase of each Action Unit and its intensity. We have modelled 34 different Action Units which may be controlled using absolute phase durations in BML scripts. Finally, we have created sequences of facial animations which were evaluated by human subjects. The evaluation results indicate that our model is suitable for generating affective facial expressions for Embodied Conversational Agents. Finally, the coded facial expressions used in the evaluation (sad, fear, joy, surprise, disgust and anger) are saved in BML files and provided with RealActor.

Using absolute phase durations for controlling Action Units provides greater flexibility for character animation, but it has a negative impact when incorporated into BML code. Since BML script developers do not know how long each of BML's primitive behaviours lasts, absolute time control of action units in addition to relative time durations of BML behaviours may have negative impact of developers' productivity. Therefore, for further improvements, we suggest a kind of an intermediate animation layer which may be added underneath the BML code and the system, similar to EMBR [Heloir and Kipp, 2009].

5 A Multiuser Tour Guide ECA

In this chapter we present an experimental system for observing multiuser-ECA interaction. The multiuser system is developed using the Generic framework (GECA framework) for integrating Embodied Conversational Agent components [Huang et al., 2008], in cooperation with Graduate School Informatics at University of Kyoto and Faculty of Science and Technology at Seikei University in Tokyo. The GECA framework is presented in the next subsection. The rest of the chapter explains the multiparty conversation issues and the way we handle it in our system. We introduce a multiuser tour guide system in which an ECA interacts with up to two human users. The system combines different technologies to detect and address the system users and draw their attention. Experimental interaction with the system produces encouraging results - the system can detect and handle the user's appearance, departure, decreased level of interest and conversational role.

5.1 The Generic framework for Embodied Conversational Agents

Generic Embodied Conversational Agent (GECA) framework is designed to overcome the issues of integrating components which constitute an ECA system (introduced in Chapter 2). The framework is based on OpenAIR routing protocol and proposes specification of interfaces between the ECA system assemblies, the integration backbone (GECA Platform) and communication libraries (GECA Plugs).

GECA Platform adopts a blackboard model, which is a methodology widely used in distributed and large-scale expert systems. GECA allows multiple blackboards which allows data sharing between the components using a publish-subscribe message passing mechanism. Every message has a message type, and when a specific message is published to a blackboard, it is forwarded to the components which subscribed the corresponding message type. Those components then process the received message and publish the results as their contribution to the blackboard. To reduce the overhead of message forwarding, direct communication between components is allowed. Every blackboard has its own manager, while the server provides message subscription and naming services for the whole system.

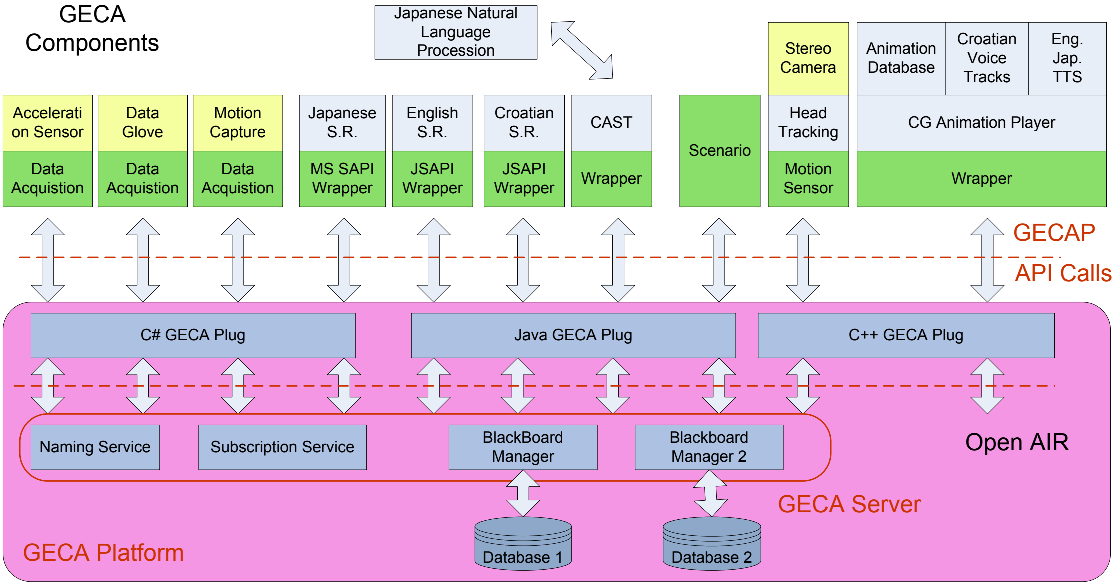


Figure 28: The conceptual diagram of GECA Framework and the configuration of a multimodal tour guide agent developed at the eNTERFACE '06 workshop

GECA Protocol (GECAP) is an XML-based, high-level, inter-component communications protocol. In GECAP, every message has a type; for example, ”*input.action.speech*” represents a speech recognition result, whereas ”*output.action.speech*” represents a text string to be synthesized by a Text-To-Speech (TTS) engine. Each message type has a specific set of elements and attributes, for example ”Intensity”, ”Duration” and ”Delay”. CGECAP message types are divided into three categories: input phase, output phase and system messages. Input and output messages are further categorized into three layers: raw parameter, primitive action and semantic interpretation. All data is represented as plain text and transferred using OpenAIR routing protocol through the GECA platform. GECAP is a specification of message format style and a set of core message types, the syntax is not fixed and can be easily extended to meet the demands of individual applications.

Components integrated with the GECA framework can be easily connected to the GECA server through C#, C++ or Java *GECA Plugs*. So far several different ECA applications have been implemented with standard GECA components such as Japanese spontaneous gesture generator, head tracker, hand shape recognizer, head pose recognizer, scenario interpreter, speech recognizer and behaviour realizer, RealActor. Among applications we note a tour guide system developed at the eNTERFACE '06 workshop in Dubrovnik [Huang et al., 2007], quiz agent and the extension of the tour guide agent with multiuser support which we present in this thesis.

5.2 Technical challenges

Most of the existing ECAs are incapable of participating in natural conversations in which the number of participants can change dynamically. They constrain themselves to interactions with a single user because dealing with multiple users is much more unpredictable and difficult. There are several challenges that multiuser interaction brings into dyadic systems, e.g. requirements to track and update the users' state, detect and resolve their requests in real-time, handle complex conversations… For these reasons multi-user ECA systems are experimental and scarce.

In the theory of multiparty human-agent communication there are several works which discuss multiparty dialogue issues. An excellent overview is given in the paper by Traum [Traum, 2001]. He explains the multiparty dialogue issues through three categories:

- *Participants' roles.* The issue refers to identification of participants' local roles and responsibilities which shift during interaction. In dyadic conversation it is always clear who is the speaker and who the addressee, but when we move to a multiparty domain, a participant might be an addressee, listener, or speaker. Some participants may also be uninvolved in the conversation. In addition to those issues, in a multiparty conversation participants may be related to tasks in a variety of ways, e.g. negotiation and team action, which may be also connected to participants' social roles and their effects on interaction behaviours (e.g. status, relationships).

- *Interaction management.* Managing the communications flow in a multiparty system is far more difficult than in a dyadic system. Some of the issues are how to give and recognize a turn and how to handle participants' channels (and backchannels). Besides, conversations can be easily split and merged together and attention can be paid to several persons. In some circumstances, participants can dynamically enter or leave the ongoing conversation. In complex scenarios, when there are lot of participants, several conversation threads may appear.
- *Grounding and obligations* are notions commonly used to model local state of dialogue. Grounding is a process of adding to the local ground between participants. By recognizing ground units, participant is able to model and participate in the grounding process. In multiparty communication usage of these models can become very complex; e.g. if there are multiple addressees, it can become unclear what a proper grounding should be.

To identify problems in the construction of multiuser systems, we consider the experimental system in which an ECA can converse with up to two users. The principle of maintaining multiuser interaction we define in the system scenario and conversational model. At the very beginning of system development we identify key features and then proceeded with several experiments with system equipment to determine its final configuration, which is shown in Figure 29:

- Speech processing. If the system users are standing next to each other the microphones should be localized to avoid an overlap between their verbal channels. Conducted experiments show that Loquendo ASR [Loquendo] meets our needs very adequately; it is stable in noisy environments, speaker-independent, there is no significant speech overlap at reasonable distances (0,5 meters approx.), and it has a keyword spotting feature absent from the free Microsoft SAPI engine. However, Loquendo sometimes reacts to voices which are not from the users. We think such errors can be prevented by using the Speaker Verification function, which we have not yet tested.
- Nonverbal data acquisition. During interaction it is extremely important to detect and locate the users so the ECA can gaze at them. For that purpose we installed

two cameras and used image processing techniques to detect the user's arrival, position and departure.

- To process image from one camera we decided to use Okao library [Okao] which provides accurate face detection and extra features such as face orientation, gaze direction, the positions and openness of eyes and mouth, gender detection, age identification and face identification from a single image. In preliminary tests with a 960x720@15fps web cam, accuracy of face detection was sufficiently high and undoubtedly usable, but most of the other functions were not reliable and could only be treated as an extra bonus. Since Okao Vision does not require stereo cameras, this result was acceptable for our system.
- The second camera identifies moving objects and detects the user's arrival and departure by computing differences between sequential images coming from the camera's input.
- The conversational model. The final conversational model we designed simplifies participants' conversational roles and is based on narration. In the system scenario, an ECA named Dubravka takes visitors on a tour of the city of Dubrovnik and talks about its cultural heritage, history, and monuments. She maintains the initiative in the conversation, though users can interrupt her and ask questions about the current topic. We predict topic-related questions and define it by using specific keywords in the speech recognition engine (such as "where, "when", "how"). To hold the users' attention during the session, Dubravka also asks the users simple "yes/no" questions.
- ECA appearance. The users should perceive the ECA's gaze direction, which has proven to be very important in regulation of conversation flow in multiparty communication. Since the system environment is displayed on a 2D screen, the ECA's size and an appropriate gaze model are important to avoid the Mona Lisa effect (an impression perceived with 2D pictures of humans, that "the eyes follow the observer across the room"). The ECA's final position on the screen was derived from initial experiments with the ECA's size and its attention towards the users. We concluded that users can distinguish the direction in which the ECA is gazing only if the ECA's size on the screen is very large.

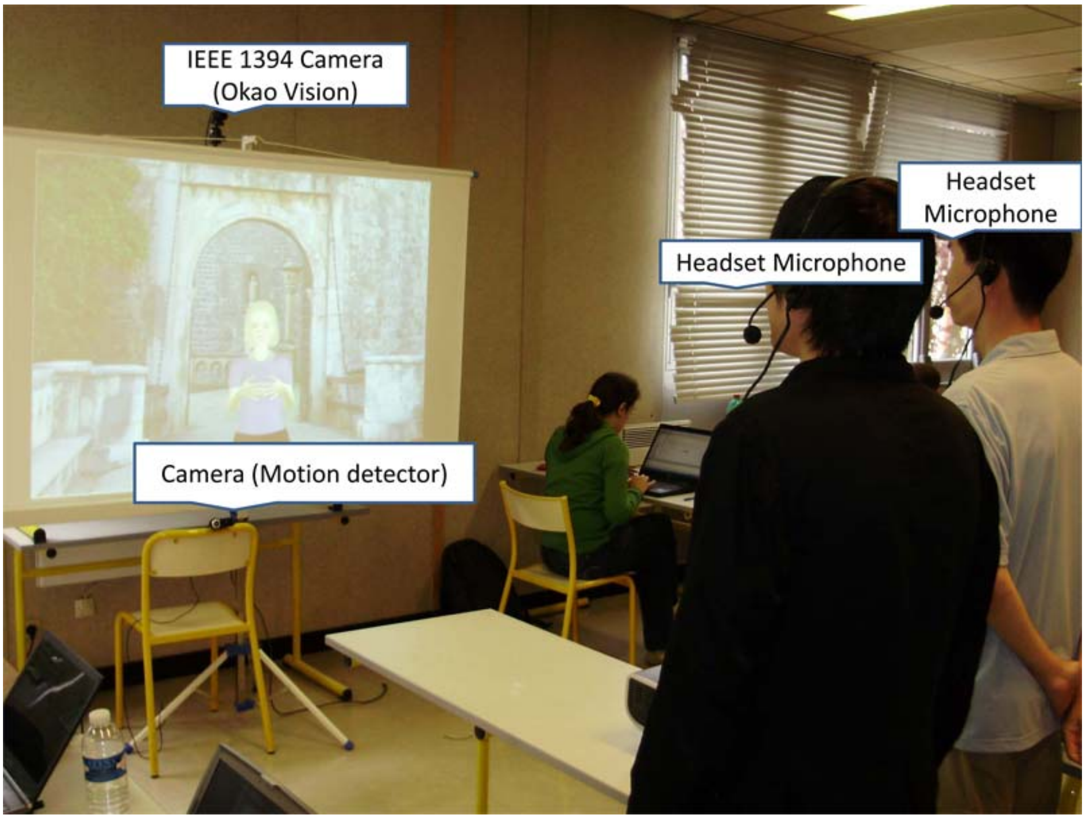


Figure 29: Final configuration of the Multiuser ECA system

5.3 System Overview

Results of experiments have given insight into the possibilities of handling and maintaining multiuser communication with the agent. On this basis we have developed the conversational model, in which we predicted specific multiuser situations, such as speech collision and mutual conversation between the users, and defined how the system responds to these situations. Afterwards, we designed and implemented the system components.

5.3.1 Multiuser aspect

The multiuser aspect is explained with regard to Traum's issues [Traum, 2001] and the conversational model:

- *Detecting the user’s appearance and the number of users.* During interaction, the system dynamically detects the user’s arrival, his position in front of the system and his departure. In its initial state, the agent waits for someone to appear. When that happens, she approaches and engages the potential user. In case that user departs, she turns back and reverts to the initial state. At most two users can stand in front of the system. If there are less than two users, Dubravka can also invite another person to join the conversation.
- *Channel management.* The system combines users’ verbal and nonverbal channels to resolve their utterances. Nonverbal behaviours taken into account are face orientation and gazing. By combining those modalities the system is able to define one of the following situations: decreased level of attention, making requests of the system, departure and speech collision.
- *Speech collision.* During interaction it might happen that the users ask questions simultaneously. The system handles this situation by having the ECA address one of the speakers and give him turn, e.g. the agent says “You can speak one by one (gesture “calm down”), I’ll respond to you both. You can be the first one (addressing one user by pointing with an open hand)”
- *Identification of conversational roles.* Our conversational model simplifies the rules of how participants’ local roles shift during the session. System scenario defines the communication workflow and several specific situations:
    1. When talking about cultural heritage of the city, the agent gazes at both users with the same frequency. In the case when one of the users asks a question, the system identifies him by using a localized microphone, and when the agent responds to him, he becomes an addressee, and the other user becomes a listener. Following the findings of studies on gazing [Vertegaal et al., 2001] [Rehm and André, 2005] we developed a computational model of gaze in which the agent gazes at the addressee more than he does at the listener.
    2. During the session, the agent may ask the users simple questions, e.g. “Is this your first time in Dubrovnik?” In this case she waits for both users' reactions and responds to each user separately. The reason why we added a questionnaire into the story is to hold the users’ interest in the session.

3. As described, most of the time the agent is the speaker and one or both users are addressees. Unfortunately, natural conversation between three parties in which each individual can become speaker, listener or addressee is not handled in a natural way. Since our agent is capable of understanding only simple questions related to the current topic, conversation between users is not welcome.

We are also concerned with features which can make interaction with the system more fluid. It is important to recover the system from failures, such as failure of speech recognition. In case a user's speech is not recognized, we propose two-stage recovery. First the agent asks the user to repeat his question. If his speech is not identified the second time around, she replies to him: "I'm sorry. I don't know a response to your question".

We also use Okao vision to recognize a situation in which the user is paying less attention to the session. If this occurs, the agent speaks until the end of the planned utterance, turns to the user and says "Seems you are not interested in this topic. Would you like to hear something else?"

5.3.2 Data Acquisition and Processing

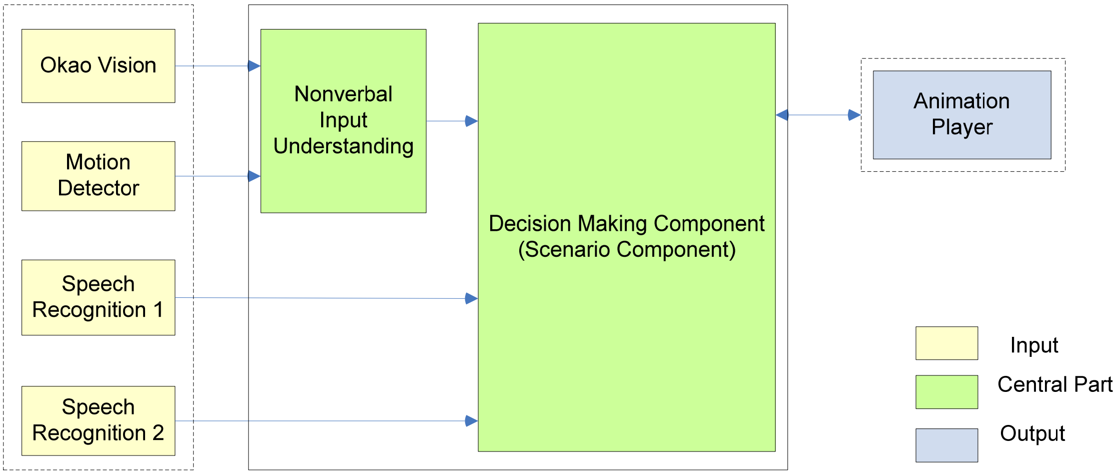


Figure 30: Design of the multiuser ECA system

Figure 30 depicts the software components connected to the GECA platform/server, which comprise our system. During a session with the system, the components communicate through the server using GECA messages. As part of the message content we introduce a variable system which describes interaction between the agent and humans. For example, `SpeechInput` represents the most recent result from one of the speech recognition components, `Speaker` represents the id of the speech recognition component, `UserNumber` represents the number of users who are standing in the user area, `UserStatus` represents availability of the users, `UserAttention` represents how much the users are paying attention to the system, `Addressee` specifies the addressee of the agent's next utterance etc. Depending on the situation in the environment, the system components update the values of variables and exchange data to define a final response.

Input components, Okao Vision, Motion Detector and two Speech Recognition components, detect and process human speech, appearance and face orientation. Combination of this data maps to one of several distinct situations in the system environment (Table 5).

Table 5: Situations in the multiuser ECA system environment and explanations how each situation is identified

Situation	**Situation Description**	**Components which detect situation**	**Explanation how components work**
User's arrival	System is in idle state until it detects potential users	Motion Detector	Motion Detector detects movements in the area which activate the system.
Number of users	During interaction system tracks the number of users	Okao Vision & Motion Detector	Okao vision determines number of faces. If it fails, Motion Detector processes users' areas in the image and detects motions in each area.
Interruptions	User interrupts Dubravka to make a request	Speech Recognition	For each utterance we predict topic-related questions defined with keywords such as "how", "where"…
Unrecognized speech	Subtype of interruption; request is not recognized	Speech Recognition	Speech is detected, but no keywords are triggered. Loquendo SAPI has additional features for this purpose
Decreased level of attention	Users are not paying attention to the system, e.g. they are gazing about	Okao Vision & Motion Detector	Okao Vision processes user's facial orientation and Motion Detector handles their movements.
Leaving the system	User(s) is(are) leaving the system	Speech Recognition, Okao Vision, Motion Detector	User can say "bye, bye" or "I'm leaving" which triggers speech recognition component. If there are no motions or faces on the input images, users have departed.

Once the S*peech recognition* component is started, it awaits users' speech input. When it receives a speech start signal, it starts recognizing the speech. The results of recognition, timing and speaker's id, are then sent to the server and passed to *Decision Making Component*. To detect user's nonverbal behaviours we use two input components: *Okao Vision* and *Motion Detector*. Okao Vision detects face orientation which is used to approximately determine the users' gaze directions. This is not very accurate but should be sufficient when we only need to distinguish rough directions like that of the screen or another user. Motion Detector divides the viewed input region in two distinct areas (which we refer to as user areas) and detects motion within each area. Detection results are sent to *Nonverbal Input Understanding* component through the server.

Nonverbal Input Understanding component uses information received from users' nonverbal channels to detect some of the situations defined in the system scenario (Table 5). First, it combines input data coming from Motion Detector and Okao Vision and uses simple heuristic methods to resolve the number of users. Since Okao Vision fails in detection when users rotate their head beyond 60 degrees, it is not sufficient to track the number of users. Therefore we additionally use Motion Detector to determine the user's presence. For example, during the system session two users listen to the agent and the left user turns his head to see who entered the room behind him. In this situation Okao Vision sets `UserNumber` variable to one, as if there is just the right user in the system, and sends it to the Understanding component. At the same time, Motion Detector detects motions in the left area and notifies the Understanding component, which then realizes that there are still two users present.

Nonverbal input understanding component also tracks users' attention. We identify two meaningful patterns the users tend to look at for a significant proportion of time – the agent and the other user. Okao Vision is used to detect face orientation which we assume is the user's gaze direction. Since two patterns (agent, other user) are placed in different directions this approach is sufficient to efficiently track the user's attention. By combining results of Okao Vision and Motion Detector, the system can smoothly respond to situations in which a user is not interested in the system anymore.

This *Decision Making Component* is responsible for defining the ECA's behaviours as responses to the system input. Its implementation is based on Information State Theory [Traum et al., 1999] [Larsson et al., 2001], which is a general theory of human-agent dialogues. Information State Theory defines a dialogue as a set of variables (or information) that describe the current state of the dialogue. The implemented prototype is information state dialogue move engine [Midiki, 2005], which is capable of handling multi-modal, multiparty conversations, dynamically changing behaviours accompanying the emotion dynamics simulating component, etc.

To support the concepts proposed in the theory, we have developed a script language [Huang et al., 2008] based on AIML [AIML]. Compared to AIML which merely matches recognized speech inputs and nonverbal inputs with predefined patterns, we introduce a variable system from Information State Theory which describes interaction between the agent and humans. Values of these variables are updated with the agent system's internal status and perception events sent from the speech recognition and nonverbal input interpretation components. Additionally, script designers can also specify variable update logic as effects of particular input patterns. During the session, the agent or the script execution engine update their internal status variables based on perception of the outside world or users and pick for execution the first valid template for which all conditions (predicates) evaluate as true. Therefore, rules such as the one that specifies what the agent should do when a user appears in or departs from a user area can be specified in scripts.

States limit possible patterns that will be used in matching in the current conversational situation and thus isolate interference from other states which may happen to have the same triggering patterns. Due to absence of a context management mechanism in the agent's behaviour control, there is no way to determine whether a user's answer is related to the last question asked by the agent. However, for example, when the agent is going to ask a yes/no question such as "Do you need a tour guide?", transition to a specific state corresponding to the question can isolate it from other yes/no questions.

Within the decision making component we introduce the variable `GlobalState` for error and interruption handling. When a failed or unknown recognition occurs, appropriate response will be sought among the categories defined in the global state. When interruptions from the user such as "excuse me" or "pardon" occur, they are also matched against the patterns defined in this state. The disadvantage of this approach is that, in absence of a full dialogue-managing central component, the agent does not conduct a plan that contains multiple steps to achieve a certain goal. The agent's behaviours are driven by the events that occurred in the outside world. The management mechanism for information like grounding or topics is not included in the script execution kernel.

To control the gaze direction of the agent, which is essential in three-party dialogue, we use `Addressee` attribute introduced in the `Utterance` element to specify if the addressee of an utterance is located to the left, right or both. Addressee specification is done by the rules in the scenario script by the script programmer; e.g. a specific utterance of the agent can be directed at the last speaker, and so on.

Finally, responses in the system output are handled by *Animation Player*, which is developed Visage's virtual environment. Animation Player communicates with the Decision Making Component and depending on the information it plays and stops behaviours, or merges new behaviours which serve as intermediate responses to the events from the environment.

5.4 Implementation

The system is constructed from several computers and hardware equipment which communicates through TCP/IP-based OpenAIR routing protocol [OpenAIR] implemented in the GECA Framework. For each component we have implemented C++ or Java wrappers which are then connected to the GECA server through GECA plugs. Speech recognition component is based on Loquendo ASR [Loquendo] and Okao Vision component is based on Okao Vision library [Okao]. From Okao the system uses the following features: face detection, face orientation tracking and eye-mouth openness. Motion Detector, the component which we use to detect moving objects in distinct areas, is based on OpenCV library [OpenCV]. The component divides the viewed image region coming from the camera into two distinct areas (which we refer to as user areas) and detects motion within each area. The user areas are surrounded with a blue and red square, respectively. To recognize moving objects we calculate a sum of pixels in one frame and compare this value to the value from the previous frame. If threshold is exceeded, moving objects are detected.

Decision Making component is implemented in Java programming language, and Animation Player in C++ programming language and libraries from Visage|SDK.

5.5 Interaction example

After constructing and integrating the system, we tested the functionality with two human subjects. Before proceeding to interact with the agent, they were informed that they could only ask questions about the current topic and they cannot move outside the user's area. We describe the interaction example:

1. User detection and session of initialization

There are no system users and the agent is standing still at Dubrovnik city gates (Figure 31). The system detects a potential user. The agent approaches and welcomes the user, instructing him to pick up the headset with a microphone. It then starts speaking, asking the user whether it is his first time to visit the city of Dubrovnik (Figure 32).

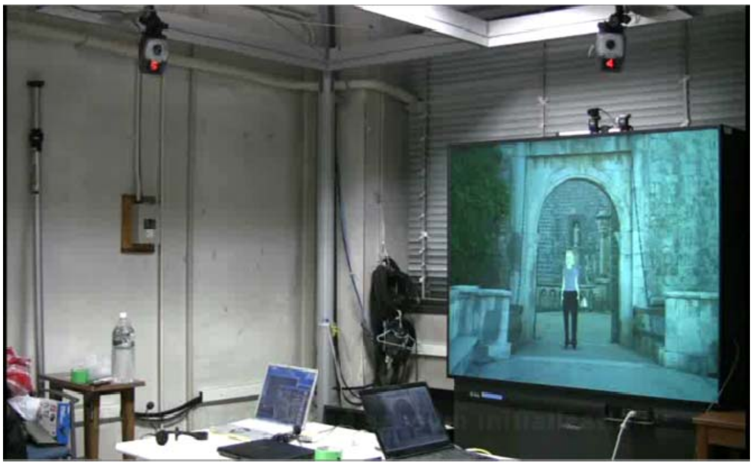


Figure 31: Initial state of the multiuser system

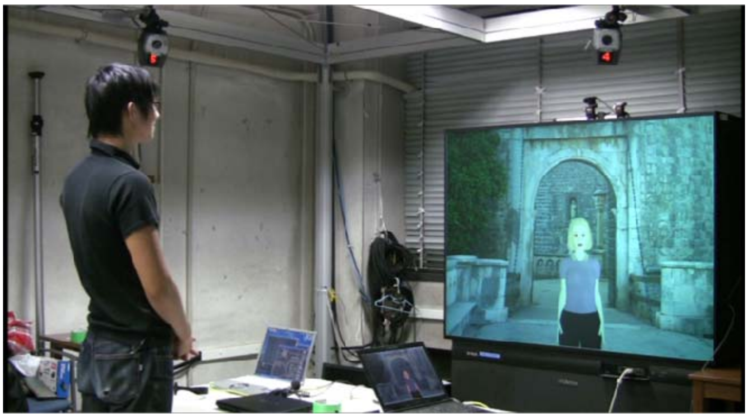


Figure 32: The agent approaches and welcomes the user

2. Interruption and welcome to the new guest

A new user is approaching while the agent is speaking. The agent stops with the speech and welcomes the new arrival. He then takes a microphone, and the agent continues with the talk about the city.

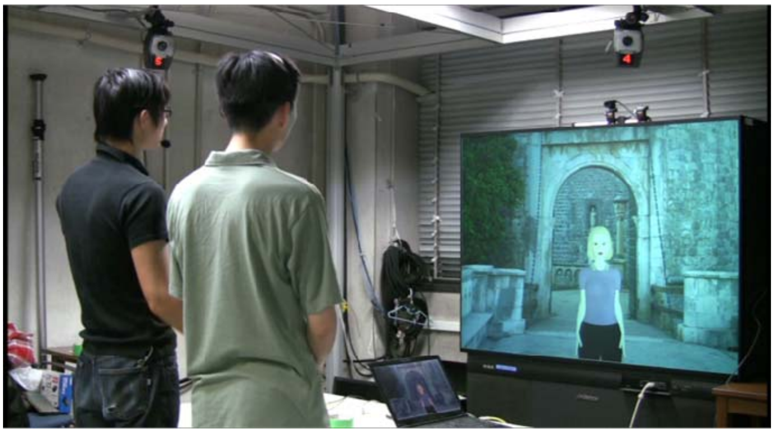


Figure 33: The agent welcomes another user

3. Handling a request and error recovery

One of the users asks a question while the agent is telling a story about the city walls. He is interested in how long it takes to walk around the city walls. The agent stops and listens to the user. At first, speech recognition fails and the agent asks the user to repeat his question. Afterwards, the agent answers the question and continues with narration.

4. Detecting decreased users' attention

After a certain time, two users start to talk to each other. The system detects this situation and interprets it as decreased level of attention. The agent stops speaking and says, "Seems you are not interested in this topic. Would you like to hear something else?" One of the users then asks the agent to tell him something about the history of the city.

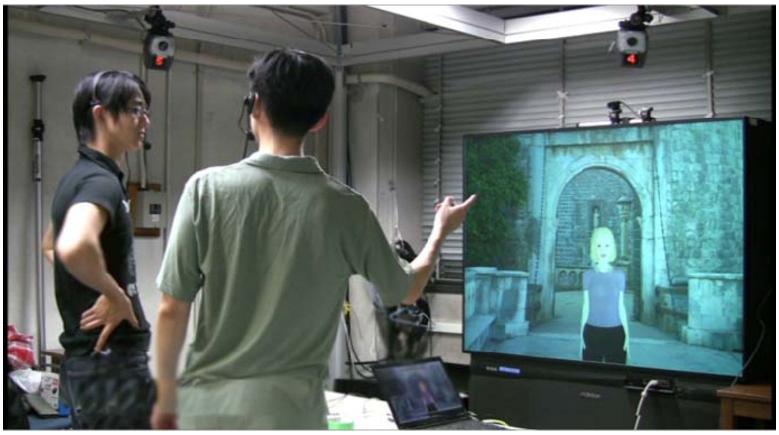


Figure 34: Detecting the situation in which users start speaking to each other

5. One of the users is leaving

While the agent is talking about the city's history, one of the users gets bored and leaves the session. The agent stops speaking and says (with approximately 2 sec delay) "Bye, bye, see you next time." Then she continues with her speech

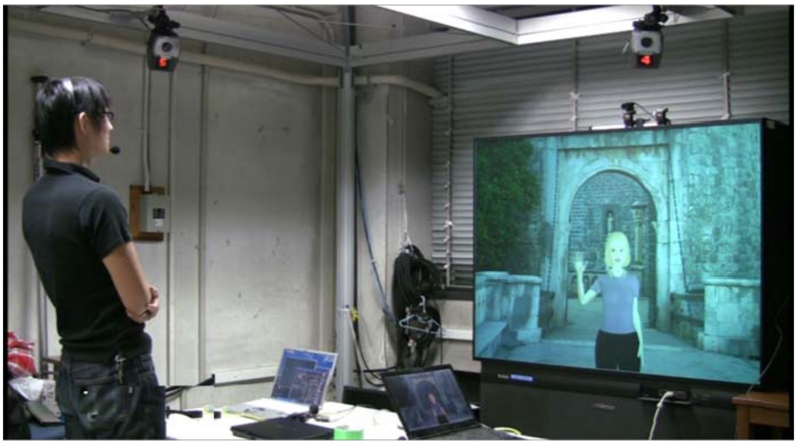


Figure 35: The agent says a goodbye to the user

6. Another user is leaving

The agent stops speaking about the history, and then asks the user if he wants to hear something else about the city. The user says "No, I have to leave". The agent then says goodbye and goes back to its initial position.

5.6 Conclusion and discussion

In this chapter we presented an implementation approach of a prototype system in which an Embodied Conversational Agent communicates with two human users. Although no exhaustive system evaluation has yet been performed, the initial results are encouraging. The system can correctly detect the user's arrival and departure, distinguish local roles of conversation participants and use ECA's gazing direction to specify an addressee. It can react to interruptions such as user requests and it can shut down automatically after users' departure. The implemented features execute smoothly, except when the user is leaving, when a certain delay is observed.

In multiuser ECA systems we are particularly interested in the level of interaction naturalness and how it can be increased. Therefore, as future directions we plan to address implementation additions which should make interaction with the system more fluent. Limitations of Speech Recognition component make the agent capable of understanding only topic-related questions. Furthermore, we aim to extend the system to dynamically invite observers to join the session, which should make the interaction interesting. Example is the situation when one user leaves and there is one observer standing in the background. In this situation, the agent will look at the observer and say "We have one free place, please come closer and join the tour". This feature demands more complex image processing than what we currently perform on our camera input.

The weakest point of the system is the rather simple conversational model which limits the ECA's ability to respond to questions we did not take into account. Nevertheless, we find this work useful for further studies and experiments in the field of multiparty interaction between ECAs and humans.

6 Conclusion

In the previous chapters we discussed our research on Embodied Conversational Agents. We addressed many different topics and issues in those fields and proposed contributions to it. In this chapter we summarize those contributions and discuss limitations of our work, as well as directions for future research.

6.1 Contributions

Synchronization of synthesized speech and nonverbal behaviours. We have presented a novel solution to synchronize nonverbal behaviours and speech which is synthesized by Text to Speech (TTS) engines which do not provide a priori timings. We employ a machine learning approach to predict timings of words coming from the TTS engine. The approach relies on neural networks, which are trained to estimate durations of those words with a database containing pairs of words and their durations. Evaluation of those networks has showed that our system is able to align 92,26% of words for the short time interval and 73,26% of words for the long interval with alignment error not exceeding the 80 ms threshold (human eye can tolerate deviations between speech and image of up to 80 ms [Steinmetz, 1996]). The networks are integrated into RealActor, a multimodal behaviour realization system, to predict timings of MSSAPI [MSSAPI] words. With RealActor system we provide instructions how developers may easily adapt those networks to different TTS engines. The package contains an application which extracts timings from the TTS engine and an application which trains neural networks to estimate extracted timings.

Face animation model based on Facial Action Coding System (FACS). We have developed a controllable facial animation model which synthesizes FACS Action Units using type, intensity and duration of animation phases: attack, sustain, decay (ASDR model [Dariouch et al., 2004]). Alternatively, the developer may specify total duration of Action Units. The animations were designed upon analysis of videos of real humans performing different Action Units. Combinations of different Action Units are handled by AU Manager, a component which blends them to form a composite display. We have evaluated the animation model in two case studies. While the first study re-creates videos of real humans who perform combinations of different Action Units, the second study synthesizes different facial expressions based on Ekman's investigation guide [Ekman and Friesen, 1978]. The evaluation results have shown that the model can be used to synthesize realistic facial motions as combinations of different Action Units.

RealActor, a multimodal behaviour realization system for ECAs. The system is designed to overcome the realization issues and drive realistic responsive and interruptive multimodal behaviours for ECAs. It uses behaviour representation described with Behavior Markup Language (BML) which defines multimodal utterances as combinations of primitive behaviours – speech, facial expression, hand gestures... During behaviour realization, the system first plans animations according to BML element references and timings of animations employed by the system. The execution is done by scheduling animations and applying them to distinct body controllers. Using control mechanisms, ECA developers may interrupt animation execution and/or add new motions in addition to the running ones.

A multiuser ECA system for observing multiuser-ECA interaction. We devise an experimental system for observing multiuser-ECA interaction. The multiuser system is developed using the Generic framework (GECA framework) for integrating Embodied Conversational Agent components [Huang et al., 2008], in cooperation with Graduate School Informatics at University of Kyoto and Faculty of Science and Technology at Seikei University in Tokyo. The system combines different technologies to detect and address the system users and draw their attention. Experimental interaction with the system produces encouraging results - the system can address the user's appearance, departure, decreased level of interest and identify his conversational role.

RealActor is implemented as a C++ static library. It provides functions which realize behaviours described in BML and control functions which stop, delete and merge running behaviours. As described in Chapter 2 and Chapter 3, these control functions are essential for realizing interruptive and responsive behaviours for ECAs. To test RealActor, we have developed an application which loads the virtual character and applies chosen BML scripts to the character. The application serves for ECA developers to produce their own behaviour scripts which may be used to drive ECA responses in real-world applications.

6.2 Limitations and Future Work

The behaviour realization system resulting from this thesis provides intuitive and continuous control of generated behaviours for Embodied Conversational Agents. It generates speech and nonverbal body behaviours which have close correlation to speech. However, some aspects of the system could be improved.

The generated behaviours are synthesized on the fly using procedural animations, except hand gestures, which are modelled using 3ds Max and then played back during realization. Although the execution results look compelling, more realistic results could be achieved by using and processing motion capture data. Therefore, on low level, we plan to introduce a new animation system based on parametric motion graphs. The new system will support example-based parametric motion (achieved by blending multiple motion capture animations with the help of registration curves) [Kovar and Gleicher, 2004] and employ parametric motion graphs for transitioning between motions [Kovar et al., 2002].

Locomotion in the system is implemented using a simple procedural method that computes walk duration from origin and destination and, depending on the computed duration, repeats the pattern of the walk cycle. Locomotion constitutes a vast area of research since it assumes interaction with the environment, path-finding and collision avoidance. It has been studied for the last fifteen years, especially for the purposes of interactive humans in video games and simulation applications. Consequently, many solutions in the area, such as solutions which use motion capture data combined with

Utilization of RealActor in real ECA systems may be tough. Manual authoring of BML content should be replaced with automatic systems, as explained in Section

2.2.4.1., which use statistical models or machine-learning techniques to learn how nonverbal actions should be assigned to spoken utterances. Besides, BML does not allow an absolute time control of executed animations. Therefore, we suggest an additional animation layer which would be added below the BML realizer, similar to EMBR [Heloir and Kipp, 2009]

The prototype of a multiuser system we developed, and experiments with equipment integration have led us to valuable insights, however interaction with an agent in the system prototype is still far from natural. For that purpose it is essential to model the multiuser conversation scenario. One idea is to set up an experimental Wizard of Oz system and according to findings, develop a computational model which responds to behaviours of conversation participants.

References

[1] AIML, A.L.I.C.E. AI Foundation, AIML, Artificial Intelligence Markup Language, http://www.alicebot.org/TR/2005/WD-AIML, 2005

[2] Alice webpage. http://www.alicebot.org.

[3] Annosoft, http://www.annosoft.com/

[4] Ahlberg, J. CANDIDE-3 -- an updated parameterized face, Report No. LiTH-ISY-R-2326, Dept. of Electrical Engineering, Linköping University, Sweden, 2001.

[5] Albrecht, I., Haber, J., Seidel, H.: Automatic generation of nonverbal facial expressions from speech. In: In Proc. Computer Graphics International 2002. (2002) 283-293

[6] Arellano, D., Varona, J., and Perales, F. J. 2008. Generation and visualization of emotional states in virtual characters. Comput. Animat. Virtual Worlds 19, 3-4 (Sep. 2008), 259-270.

[7] Arikan, O., Forsyth D. A.: Interactive Motion Generation from Examples. ACM Transactions on Graphics (Proc. SIGGRAPH 2002) 21, pp. 483-490., July 2002

[8] Armstrong, N.: Field Guide to Gestures: How to Identify and Interpret Virtually Every Gesture Known to Man. Quirk Books (2003)

[9] Axelsson, A. and Björnhall, E., “Real time speech driven face animation”, Master Thesis at The Image Coding Group, Department of Electrical Engineering at Linköping, Linköping, 2003

[10] Autodesk Software, Maya and 3ds Max http://usa.autodesk.com/

[11] Badler, N. I., Bindiganavale, R., Allbeck, J., Schuler, W., Zhao, L., and Palmer, M. 2000. Parameterized action representation for virtual human agents. In J. Casselll, et al. (eds.), Embodied Conversational Agents MIT Press, Cambridge, MA, 256-284.

[12] Becker, C., Kopp, S., Wachsmuth, I.: "Simulating the Emotion Dynamics of a Multimodal Conversational Agent", ADS 2004, pp. 154-165, 2004.

[13] Bianchi-Berthouze, N. and Kleinsmith, A.: A categorical approach to affective gesture recognition. Connection Science 15(4) (2003) 259–269

[14] Bevacqua, E., Mancini, M., Niewiadomski, R., Pelachaud, C. An expressive ECA showing complex emotions. AISB'07 Annual convention, workshop ``Language, Speech and Gesture for Expressive Characters", Newcastle UK, pp. 208-216, April 2007.

[15] Bickmore, T., Pfeifer, L., and Paasche-Orlow, M. (2007) Health Document Explanation by Virtual Agents. In Proceedings of Intelligent Virtual Agents '07. Paris, France

[16] Bickmore, T. and Casselll, J. (2001) Relational Agents: A Model and Implementation of Building User Trust. Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI), pp. 396-403. March 31 - April 5. Seattle, Washington.

[17] Blumberg, B. 1997. Multi-level Control for Animated Autonomous Agents: Do the Right Thing… Oh, Not That… In Creating Personalities for Synthetic Actors, R. Trappl and P. Petta (Eds.), Springer-Verlag, New York.

[18] BML Specification, http://wiki.mindmakers.org/projects:bml:draft1.0

[19] Brandherm, B., Prendinger, H., and Ishizuka, M. 2008. Dynamic Bayesian network based interest estimation for visual attentive presentation agents. In Proceedings of the 7th international Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1 (Estoril, Portugal, May 12 - 16, 2008). International Conference on Autonomous Agents. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 191-198.

[20] Brkic, M., Smid, K., Pejsa, T., and Pandzic, I. S., Towards Natural Head Movement of Autonomous Speaker Agent, Proceedings of the 12th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems KES 2008, Zagreb, Croatia, 2008.

[21] Carnegie, Carnegie Mellon University Motion Capture Database http://mocap.cs.cmu.edu/

[22] Casselll, J "Towards a Model of Technology and Literacy Development: Story Listening System" (2004) Journal of Applied Developmental Psychology 25 (1): 75-105.

[23] Casselll, J., Sullivan, J., Prevost, S., and Churchill, E., 2000, Embodied Conversational Agents. Cambridge, MA: MIT Press.

[24] Casselll, J., Bickmore, T., Billinghurst, M., Campbell, L. Chang, K., Vilhjalmsson, H., and Yan, H., “Embodiment in Conversational Interfaces: Rea,” presented at CHI 99, Pittsburgh, PA, 1999.

[25] Casselll, J., Vilhjalmsson, H.H., Bickmore, T.: Beat: the behaviour expression animation toolkit. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM (2001) 477-486

[26] Chovil, N.: Discourse-oriented facial displays in conversation. Research on Language and Social Interaction 25 (1991) 163-194

[27] Churcher, E., Atwell, E. S., and Souter, C. 1997 Dialogue management systems: a survey and overview. Technical Report 97.6, School of Computer Studies, University of Leeds, February 1997

[28] Collada, https://collada.org/

[29] Costantini, E., Pianesi, F. and Cosi. P. Evaluation of Synthetic Faces: Human Recognition of Emotional Facial Displays. In E. Andrè, L. Dybkiaer, W. Minker, and P. Heisterkamp, editors, Affective Dialogue Systems ADS '04, Springer-Verlag

[30] Dariouch, N. Ech Chafai, M. Mancini, C. Pelachaud, "Tools to Create Individual ECAs, Workshop Humaine, Santorini, September (2004)

[31] Deng, Z. and Ma, X. 2008. Perceptually guided expressive facial animation. In Proceedings of the 2008 ACM Siggraph, Eurographics, Symposium on Computer Animation (Dublin, Ireland, July 07 - 09, 2008), Eurographics Association, Aire-la-Ville, Switzerland, 67-76.

[32] Egges, A. Real-time Animation of Interactive Virtual Humans. PhD Thesis, University of Geneva, Geneva, Switzerland. August 2006.

[33] Ekman, P. (1973) Cross-cultural studies of facial expression, pp. 169-222 in P. Ekman (ed.) Darwin and Facial Expression.

[34] Ekman, P. In: About brows: Emotional and conversational signals. Cambridge University Press, Cambridge (1979) 169-202

[35] Ekman P. and Friesen W., Facial Action Coding System: A Technique for the Measurement of Facial Movement. Consulting Psychologists Press, Palo Alto, 1978.

[36] Ekman P. And Friesen W., The repertoire of nonverbal behaviours: Categories, origins, usage and coding

[37] Ekman, P. and Rosenberg E. L. (1997). “What the face reveals: Basic and applied studies of spontaneous expression using the facial action coding system”, Oxford University Press

[38] Elfenbein, H. A., & Ambady, N. (2002a). On the universality and cultural specificity of emotion recognition: A meta-analysis. *Psychological Bulletin*, *128*, 203-235.

[39] Elliot, C.; Brzezinski, J.; Sheth, S.; and Salvatoriello, R. 1997. Story-morphing in the Affective Reasoning paradigm: Generating stories automatically for use with “emotionally intelligent” multimedia agents. Institute for Applied Artificial Intelligence, School of Computer Science, Telecommunications, and Information Systems, DePaul University.

[40] FaceGen, http://www.facegen.com/

[41] Faloutsos, P., van de Panne, M., Terzopoulos, D.: The virtual stuntman: Dynamic characters with a repertoire of autonomous motor skills. Computers & Graphics 25(6) (2001) 933–953

[42] Fratarcangeli, M., Adolfi, M., Stankovic, K., Pandzic, I.S.: Animatable face models from uncalibrated input features. In: Proceedings of the 10th International Conference on Telecommunications ConTEL 2009. (2009)

[43] Gosselin, Pierre; Kirouac, Gilles, Le decodage de prototypes emotionnels faciaux, Canadian Journal of Experimental Psychology, (1995)., pp. 313-329

[44] Greta Documentation, http://www.tsi.enst.fr/~pelachau/Greta/mediawiki/index.php/Main_Page

[45] H-ANIM, Humanoid Animation Working Group, http://www.h-anim.org/

[46] Hartmann, B., Mancini, M., and Pelachaud, C.: Formational Parameters and Adaptive Prototype Instantiation for MPEG-4 Compliant Gesture Synthesis. Proceedings of Computer Animation 2002, IEEE Computer Society Press, 2002.

[47] Hartmann, B., Mancini, M., and Pelachaud, C. 2006. Implementing expressive gesture synthesis for embodied conversational agents. In Proceedings of Gesture Workshop. Lecture Notes Artificial Intelligence, vol. 3881. Springer. 45–55.

[48] Hayes-Roth, B.; van Gent, R.; and Huber, D. 1997. Acting in Character. In Creating Personalities for Synthetic Actors, R. Trappl and P. Petta (Eds.), Springer-Verlag, New York.

[49] Heloir, A. and Kipp, M.: EMBR - A Realtime Animation Engine for Interactive Embodied Agents. IVA 2009, 393-404

[50] Heylen, D.K.J. and Kopp, S. and Marsella, S.C. and Pelachaud, C. and Vilhjálmson, H. (2008) The Next Step towards a Function Markup Language. In: Intelligent Virtual Agents 2008, 1-3 Sep 2008, Tokyo, Japan.

[51] Hill, R., Gratch, J., Marsella, S., Rickel, J., Swartout, W., and Traum, D. Virtual Humans in the Mission Rehearsal Exercise System. Künstliche Intelligenz (KI Journal), Special issue on Embodied Conversational Agents, 2003.

[52] Horde3D, http://www.horde3d.org/

[53] Huang, H.-H.; Čereković, A.; Nakano, Y.; Nishida, T.; Pandžić, I. S. The Design of a Generic Framework for Integrating ECA components, Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, Padgham, Lin ; Parkes, David ; Müller, Jörg ; Parsons, Simon (ur.). Estoril, Portugal: Academic Press, 2008. 128-135

[54] Huang, H.-H., Čereković, A., Tarasenko, K.; Zorić, G., Levačić, V., Pandžić, S. I.; Nakano, Y.; Nishida, T., An Agent Based Multicultural Tour Guide System With Nonverbal user Interface, Journal on Multimodal User Interfaces (JMUI). 1 (2007) , 1; pp. 41-48

[55] Humanoid Animation Working Group, http://www.h-anim.org/

[56] Ingemars, N.: A feature based face tracker using extended Kalman filtering, 2007, University essay from Linköping University

[57] Irrlicht3D, http://irrlicht.sourceforge.net/

[58] Jonsdottir, G. R., K. R Thórisson and Nivel, E. (2008). Learning Smooth,. Human-Like Turntaking in Realtime Dialogue. Intelligent Virtual Agents (IVA'08), Tokyo

[59] Kenny, P., Parsons, T. D., Gratch, J., Leuski, A., and Rizzo, A. A. 2007. Virtual Patients for Clinical Therapist Skills Training. In Proceedings of the 7th international Conference on intelligent Virtual Agents (Paris, France, September 17 - 19, 2007). C. Pelachaud, J. Martin, E. André, G. Chollet, K. Karpouzis, and D. Pelé, Eds. Lecture Notes In Artificial Intelligence, vol. 4722. Springer-Verlag, Berlin, Heidelberg, 197-210.

[60] Kenny, P., Parsons, T., Gratch, J., and Rizzo, A. 2008. Virtual humans for assisted health care. In Proceedings of the 1st international Conference on Pervasive Technologies Related To Assistive Environments (Athens, Greece, July 16 - 18, 2008). F. Makedon, L. Baillie, G. Pantziou, and I. Maglogiannis, Eds. PETRA '08, vol. 282. ACM, New York, NY, 1-4.

[61] Kim T. H., Park S. I., Shin S. Y.: Rhythmic-motion synthesis based on motion-beat analysis. ACM Transactions on Graphics (Proc. SIGGRAPH 2003) 22, 3, pp. 392-401., July 2003

[62] Kipp, M. Anvil – a generic annotation tool for multimodal dialogue. In: Proc. Of Eurospeech, pp. 1367 – 1370 (2001)

[63] Kipp, M., Gebhard. P, (2008). IGaze: Studying Reactive Gaze Behaviour in Semi-immersive Human-Avatar Interactions. Proc.8th International Conference on Intelligent Virtual Agents, Tokyo, pp. 191-199

[64] Kopp, S., Becker, C., Wachsmuth, I. (2006). The Virtual Human Max – Modeling Embodied Conversation. In: KI 2006 – Demo Presentation, Extended Abstracts, pp. 21-24

[65] Kopp, S., Wachsmuth, I.: Synthesizing multimodal utterances for conversational agents. Computer Animation and Virtual Worlds 15 (2004) 39-52

[66] Kopp, S., Krenn, B., Marsella, S., Marshall, A., Pelachaud, C., Pirker, H., Thórisson, K., Vilhjalmsson, H. 2006. Towards a Common Framework for Multimodal Generation in ECAs: The Behaviour Markup Language. Proc.6th International Conference on Intelligent Virtual Agents, Marina del Rey

[67] Kovar, L., and Gleicher, M. 2004. Automated extraction and parameterization of motions in large data sets. In ACM Transactions on Graphics. 23(3):559-568.

[68] Kovar, L., Gleicher, M., and Pighin, F. 2002. Motion graphs. In ACM Transactions on Graphics. 21(3):473-482.

[69] KQML, Knowledge Query and Manipulation Language, UMBC, http://www.csee.umbc.edu/kqml/

[70] Kwon, Shin: Motion Modeling for On-Line Locomotion Synthesis, Eurographics, ACM SIGGRAPH Symposium on Computer Animation, 2005

[71] Larsson, S., Berman, A., Gronqvist, L., and Kronlid, F. (2002). TRINDIKIT 3.0 Manual. Trindi Deliverable D6.4

[72] Lee, J., Marsella, S.: Nonverbal behaviour generator for embodied conversational agents. In: Intelligent Virtual Agents. (2006) 243-255

[73] Leigh, R., Zee, D.: The Neurology of Eye Movements. New York: Oxford University Press. 2006

[74] Li, Y., Wang, T., and Shum, H.-Y. 2002. Motion texture: A two-level statistical model for character synthesis. In ACM Transactions on Graphics. 21(3):465-472.

[75] List, T., J. Bins, R. B. Fisher, D. Tweed, K. R. Thórisson (2005). Two Approaches to a Plug-and-Play Vision Architecture - CAVIAR and Psyclone. In K. R. Thórisson, H. Vilhjalmsson, S. Marsella (eds.), AAAI-05 Workshop on Modular Construction of Human-Like Intelligence, Pittsburgh, Pennsylvania, July 10, 16-23. Menlo Park, CA: American Association for Artificial Intelligence.

[76] Loquendo, http://www.loquendo.com/en/

[77] Ljunglöf, P. Dialogue management as interactive tree building. In Proc. DiaHolmia'09, 13th Workshop on the Semantics and Pragmatics of Dialogue, Stockholm, Sweden, 2009.

[78] Malatesta, L., Raouzaiou, A., Karpouzis, K., Kollias, S., MPEG-4 facial expression synthesis, in Personal and Ubiquitous Computing, Volume 13, Number 1, January 2009 , pp. 77-83(7)

[79] Mana, N. and Pianesi, F. 2006. HMM-based synthesis of emotional facial expressions during speech in synthetic talking heads. In Proceedings of the 8th international Conference on Multimodal interfaces (Banff, Alberta, Canada, November 02 - 04, 2006). ICMI '06. ACM, New York, NY, 380-387

[80] Mancini, M. and Pelachaud, C.: Dynamic behaviour qualifiers for conversational agents. Intelligent Virtual Agents, IVA'07, Paris, September 2007.

[81] Martin, J., Niewiadomski, R., Devillers, L., Buisine, S., & Pelachaud, C. (2006). Multimodal complex emotions: gesture expressivity and blended facial expressions. International Journal of Humanoid Robotics (IJHR), special issue Achieving Human-Like Qualities in Interactive Virtual and Physical Humanoids: 3 (3), 269-291.

[82] Matlab, http://www.mathworks.com

[83] Mäkäräinen, M. and Takala, T. (2009) An Approach for Creating and Blending Synthetic Facial Expressions of Emotion, in Intelligent Virtual Agents, 243- 249, 2009

[84] McCrae, R. and John, O. (1992). “An Introduction to the Five-Factor Model and its Applications”, Journal of Personality, Routledge, 60 (2), 175–215.

[85] McNeill, D.: Hand and Mind: What Gestures Reveal about Thought. University of Chicago Press (1992)

[86] Mehrabian, A. (1995). “Framework for a Comprehensive Description and Measurement of Emotional States”, Genetic, Social, and General Psychology Monographs, Heldref Publishing, 121 (3), 339-361.

[87] Metta, G., Fitzpatrick, P. , Natale. L., YARP: yet another robot platform. In the International Journal on Advanced Robotics Systems, Special Issue on Software Development and Integration in Robotics. March 2006.

[88] Midiki, The MITRE Corporation (2005). Midiki User’s Manual, version 0.1.3 beta edition.

[89] MindMakers.org, http://www.mindmakers.org

[90] MSAPI, Microsoft Speech API, speech API: http://www.microsoft.com/speech/speech2007/default.mspx

[91] Moving Pictures Expert group, MPEG-4 International standard, ISO/IEC 14496, http://www.cselt.it/mpeg/

[92] Neff, M., Kipp, M., Albrecht, I., Seidel, H.P.: Gesture Modelling and Animation Based on a Probabilistic Recreation of Speaker Style. Transactions on Graphics (2008), 27, 1, Article 5

[93] NN at your fingertips, Neural networks at your fingertips, http://www.neural-networks-at-your-fingertips.com/

[94] Noh, J. Y., Fidaleo, D. and Neumann, U. Animated deformations with radial basis functions. In Proceedings of the ACM Symposium on Virtual Reality Software and Technology, pages 166–174. ACM Press, 2000.

[95] Object Management Group CORBA/IIOP Specification, http://www.omg.org/technology/documents/formal/corba_iiop.htm

[96] Ochs, M., Niewiadomski, R., Pelachaud, C., & Sadek, D. (2005). Intelligent expressions of emotions. 1st International Conference on Affective Computing and Intelligent Interaction ACII. China.

[97] Ogre3D, http://www.ogre3d.org/

[98] Okao, Okao library, http://www.omron.com/r_d/coretech/vision/okao.html

[99] OpenAIR 1.0, http://www.mindmakers.org/openair/airPage.jsp

[100] OpenCV, http://sourceforge.net/projects/opencvlibrary/

[101] Organic Motion, http://www.organicmotion.com/

[102] Ortony, A., Clore, G. L. and Foss, M. A. (1987). “The referential structure of the affective lexicon”, Cognitive Science, Cognitive Science Society, Inc., 11 (3), 341-364.

[103] Ortony A, Clore G L, Collins A (1988). The cognitive structure of emotions. Cambridge University Press, Cambridge, MA.

[104] Ostermann, J. Animation of Synthetic Faces in MPEG-4, In CA 1998, Proceedings of the Computer Animation, p. 49, IEEE Computer Society

[105] Pan, X., Gillies, M., Sezgin, T. M. & Loscos, C. (2007) Expressing Complex Mental States Through Facial Expressions. In the Second International Conference on Affective Computing and Intelligent Interaction, Lisbon, Portugal

[106] Pandzic, I.S, Forchheimer, R. (editors), MPEG-4 Facial Animation - The standard, implementations and applications, John Wiley & Sons, 2002, ISBN 0-470-84465-5

[107] Park S. I., Shin H. J., Shin S. Y.: On-line motion blending for real-time locomotion generation. Computer Animation and Virtual Worlds 15 , pp.125-138, Sept 2004

[108] Parke, F. I.: Computer generated animation of faces. In Proceedings ACM Annual Conference, pages 451–457. ACM Press, 1972.

[109] Parsons, T. D., Kenny, P., Ntuen, C., Pataki, C. S., Pato, M., Rizzo, A. A,-George, C.St. and Sugar, J. (2008), Objective Structured Clinical Interview Training using a Virtual Human Patient, Studies in Health Technology and Informatics, 132, 357-362.

[110] Pelachaud, C.: Studies on gesture expressivity for a virtual agent, Speech Communication, Volume 51, Issue 7, Research Challenges in Speech Technology: A Special Issue in Honour of Rolf Carlson and Bjorn Granstrom, July 2009, 630-639

[111] Perlin, K. Real-time responsive animation with personality. 1995. IEEE Transactions on Visualization and Computer Graphics.

[112] Platt, S. M. and Badler, N.. Animating facial expression. Computer Graphics, 15(3):245–252, 1981.

[113] Poggi, I. Mind, hands, face and body: a goal and belief view of multimodal communication, Weidler (2007)

[114] Posner, R., Serenari, M.: Blag: Berlin dictionary of everyday gestures

[115] Raouzaiou A, Tsapatsoulis N, Karpouzis K, Kollias S (2002) Parameterized facial expression synthesis based on MPEG-4. EURASIP J Appl Signal Process 2002(10):1021–1038

[116] Reed L., Sayette M., Cohn J. (2007). "Impact of depression on response to comedy: A dynamic facial coding analysis". Journal of abnormal psychology 116 (4): 804–9.

[117] Rehm, M., André, E., and Wissner, M. 2005. Gamble v2.0: social interactions with multiple users. In Proceedings of the Fourth international Joint

Conference on Autonomous Agents and Multiagent Systems (The Netherlands, July 25 - 29, 2005). AAMAS '05.

[118] Rehm, M. and André, E. Where do they look? Gaze Behaviours of Multiple Users Interacting with an Embodied Conversational Agent. In Proceedings of Intelligent Virtual Agents (IVA), 2005.

[119] Rojas, R.: Neural Networks - A Systematic Introduction. Springer-Verlag (1996)

[120] Ruttkay, Z., Noot, H. and Hagen, P. Ten. Emotion disc and emotion squares: tools to explore the facial expression face', Computer Graphics forum, 22, 49–53, (March 2003).

[121] Schlosberg, H. A scale for judgement of facial expressions. Journal of Experimental Psychology, 29:497–510, 1954.

[122] Schröder, M. and Trouvain, J. (2003). The German Text-to-Speech Synthesis System MARY: A Tool for Research, Development and Teaching. International Journal of Speech Technology, 6, pp. 365-377.

[123] Scherer, K. R., Shorr, A., Johnstone, T. (Ed.). (2001). Appraisal processes in emotion: theory, methods, research. Canary, NC: Oxford University Press.

[124] Smid, K., Zoric, G., Pandzic, I.S., [HUGE]: Universal architecture for statistically based human gesturing. In: Proceedings of the 6th International Conference on Intelligent Virtual Agents IVA 2006. (2006) 256-269

[125] Steinmetz, R.: Human perception of jitter and media synchronization. IEEE Journal on Selected Areas in Communications 14(1) (1996)

[126] Taylor, P.A., Black, A., Caley, R.: The architecture of the festival speech synthesis system. In: The Third ESCA Workshop in Speech Synthesis. (1998) 147-151

[127] Thiebaux, M., Marshall, A., Marsella, S., Kallmann, M.: Smartbody: Behaviour realization for embodied conversational agents. In: Proceedings of Autonomous Agents and Multi-Agent Systems AAMAS. (2008)

[128] Thoroughman, K.A., Shadmehr, R.: Learning of action through combination of motor primitives. Nature 407(6805) (2000) 742- 747

[129] Tolani, D., Inverse kinematics Methods for Human Modelling and Simulation. PhD Thesis, University of Pennsylvania, 1998.

[130] Traum, D., Issues in multiparty dialogues. (2003) In Advances in Agent Communication (F. Dignum, ed.). Springer-Verlag LNCS.

[131] Traum, D., Talking to Virtual Humans: Dialogue Models and Methodologies for Embodied Conversational Agents" In I Wachsmuth and G Knoblich (Ed.), Modeling Communication with Robots and Virtual Humans, pp. 296-309, 2008

[132] Traum, D., Bos, J., Cooper, R., Larsson, S., Lewin, I., Matheson, C., and Poesio, M. (1999). A model of dialogue moves and information state revision.

[133] Traum, D., Marsella, S., Gratch, J., Emotion and Dialogue in the MRE Virtual Humans, 2004, In Affective Conversational Agents and Dialogue Simulation, 117 – 127

[134] Traum, D., Marsella, S. C., Gratch, J., Lee, J., and Hartholt, A. 2008. Multi-party, Multi-issue, Multi-strategy Negotiation for Multi-modal Virtual Agents. In Proceedings of the 8th international Conference on intelligent Virtual Agents (Tokyo, Japan, September 01 - 03, 2008). H. Prendinger, J. Lester, and M. Ishizuka, Eds. Lecture Notes in Artificial Intelligence, vol. 5208. Springer-Verlag, Berlin, Heidelberg, 117-130.

[135] Traum, D. and Rickel, J. 2002. Embodied agents for multiparty dialogue in immersive virtual worlds. In Proceedings of the First international Joint Conference on Autonomous Agents and Multiagent Systems: Part 2 (Bologna, Italy, July 15 - 19, 2002). AAMAS '02. ACM, New York, NY, 766-773

[136] Vertegaal, R., Slagter, R., van der Veer, G., and Nijholt, A. 2001. Eye gaze patterns in conversations: there is more to conversational agents than meets the eyes. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Seattle, Washington, United States). CHI '01. ACM, New York, NY, 301-308

[137] Vicon Blade, http://www.mocap.lt/motion-capture/software/vicon-blade.html

[138] Vilhjalmsson, H., Cantelmo, N., Casselll, J., Chafai, N. E., Kipp, M., Kopp, S., Mancini, M., Marsella, S., Marshall, A. N., Pelachaud, C., Ruttkay, Z., Thórisson, K. R., van Welbergen, H. und van der Werf, R. J. 2007. The

Behaviour Markup Language: Recent Developments and Challenges. In: Pelachaud et al. (eds.) Proceedings of the 7th International Conference on Intelligent Virtual Agents, LNAI 4722, Springer, pp. 99-111. Paris

[139] Vinayagamoorthy V., Gillies, M., Steed, A., Tanguy, E., Pan, X., Loscos, C. and Slater, M., "Building Expression into Virtual Characters," Proc. Eurographics State of the Art Reports, 2006.

[140] Visage Technologies, http://www.visagetechnologies.com

[141] Virtual Reality Modeling Language VRML, http://www.web3d.org/x3d/specifications/vrml/

[142] Web Services Activity, http://www.w3.org/2002/ws/

[143] Wehrle T., Kaiser S., Schmidt S., and Scherer K. (2000) Studying the dynamics of emotional expression using synthesized facial muscle movements. J Pers Soc Psychol 78(1):105–119

[144] Zordan, V.B., Hodgins, J.K.: Motion capture-driven simulations that hit and react. In: Proc. ACM SIGGRAPH Symposium on Computer Animation. (2002) 89–96

[145] Zoric, G.; Pandzic, I., Real-time language independent lip synchronization method using a genetic algorithm. Signal Processing. 86 (2006), 12; 3644-3656

[146] Zoric, G., Smid, K., Pandzic, I.S. Towards facial gestures generation by speech signal analysis using huge architecture. In: Multimodal Signals: Cognitive and Algorithmic Issues: COST Action 2102 and euCognition International School Vietri sul Mare, Italy, April 21-26, 2008 Revised Selected and Invited Papers, Berlin, Heidelberg, Springer-Verlag (2009) 112-120

[147] Zoric, G., Cerekovic, A., Pandzic, I.S. Automatic Lip Synchronization by Speech signal analysis, Proceedings of Interspeech 2008 incorporating SST 2008. 2323-2323

Acronyms

ECA	Embodied Conversational Agent
BML	Behavior Markup Language
SAIBA	Situation, Agent, Intention, Behavior and Animation Framework
NN	Neural Network
FML	Function Markup Language
BDI	Belief-Desire-Intention Model
PAD	Pleasure, Arousal and Dominance mood space
CG	Computer-generated
HMM	Hidden Markov Model
FACS	Facial Action Coding System
AU	Action Unit
TTS	Text-to-speech
SAPI	Speech Application Programming Interface
MPEG	Moving Picture Experts Group
MPEG-4 FA	MPEG-4 Facial Animation
MPEG-4 FBA	MPEG-4 Face and Body Animation
FPs	Feature Points
FAPs	Face Animation Parameters
VRML	Virtual Reality Modelling Language
BEAT	Behavior Animation Toolkit
DM	Dialogue Management
NLP	Natural Language Processing
NLG	Natural Language Generation
SSML	Speech Synthesis Markup Language
CART	Classification and Regression Trees
BNN	Backpropagation Neural Network

Index

RealActor Reference Manual

RealActor has been developed as a static library which comes as a part of Visage|SDK animation toolkit. It is built using Microsoft Visual Studio 2005 and currently available only on Microsoft Windows Vista and XP.

The SDK is distributed with a sample application which uses OpenGL or Horde3D engine for graphics rendering. The application is named Behavior Player and is used to show basic features of RealActor. In the future we plan to provide support for OGRE rendering engine as well.

Using RealActor library, developers may develop their own applications according to their needs. It amounts to creating an instance of RealActor class and a virtual character, associating RealActor with the character and adding the character into the rendering engine scene graph (the principle is shown in Behavior Player introduced in Chapter 3). Then they can execute behaviours by calling different functions defined in RealActor's class. If developers want to use text-to-speech (TTS) engine which differs from MS SAPI, they will need to perform neural network training to achieve optimal results during execution of scripted behaviours.

Following are API command summary and explanation how developers may adapt RealActor to their TTS engine. In the end, we give a list of BML elements supported in RealActor, explain how developers may extend the library with their animation examples, and in the end we give examples of BML scripts which may be used in the system.

API Command summary

The following outlines the different API commands.

Player commands

- *runBMLDocument.* This function reads a BML document and realizes character behaviours as described in the document. It can be called multiple times while BML behaviours are still running. The content that is read is sequentially applied to the virtual character.
- *runBMLContent.* Does the same as runBMLDocument, except it accepts a string as the input argument
- *stopDelete.* This function stops and deletes all BML behaviours.
- *pause.* This function pauses (stops) all BML behaviours. The behaviours are not erased and can be resumed with the resume function.
- *resume.* This function resumes paused behaviors.
- *runPriorityBMLDocument.* This function pauses any running behaviours, reads the BML document and plays character behaviours as described in the document. After the function is finished, behaviours can either be resumed using the resume() function or deleted. It can be called only once while behaviours are running. Priority behaviours cannot be stopped, deleted or paused.
- *runPriorityBMLContent.* Does the same as runPriorityBMLDocument, except it accepts string as the input argument
- *mergeBMLDocument.* This function reads a BML document and merges behaviors with running behaviors.
- *mergeBMLContent.* Does the same as mergeBMLDocument, except it accepts string as the input argument
- *playBMLElement.* This function plays a BML element. BML element should be annotated with an element type and its attributes. When finished, the BML element is removed and deleted. By using this function, it is possible to run animation content without writing the BML code. Since the BML element is not placed inside any BML block, no feedback is provided.

- *setIdleBehavior*. Changes idle motions of a character according the values on activation-evaluation emotion disc

Feedback and Event commands

- *attachObserver*. Attaches an observer implementation to RealActor. In order to get events and feedback notifications, an instance of RealActorObserver must be registered using attachObserver() method. After this, the observer will be called when events occur. Up to 100 observers can be attached. They will be called in the same order as they were attached.
- *eventEmiter*. Virtual function which is called by RealActorObserver when a specific event described in the BML content occurs. The purpose of this function is to synchronize specific events in the application.
- *getFeedback*. Virtual function which is called by Real Actor during the realization phase. It is used to check progress of behaviour realization. BML Specification offers five different types of feedback: start, progress, stop, exception, warning. Implementation of those functions can be used to send feedback to a behaviour planning phase of SAIBA framework (see BML specification [BML specification]).

Additional commands

- *getCharacterModel*. Function provides a reference to VisageCharModel, which represents the virtual character model.
- *getCharacterName*. Function provides the name of the virtual character used in RealActor.

Neural network training

RealActor has been trained to support voices from MS SAPI engine which are used for speech synthesis. If a developer wants to use another TTS engine he will have to proceed with neural network training to achieve optimal results of executed behaviours. For this purpose we provide a package with instructions how to adapt RealActor to another TTS engine.

The package contains the following applications:

- Speech Synthesis - extracts time durations of words coming from TTS. The application reads files containing sentences and for each sentence it calculates how long each words lasts. Words are categorized into two different groups – group with plain words and group with words which are followed by punctuation.
- Train Network – application which trains neural networks for RealActor. There are two variants of this application, the first one trains networks responsible for estimation of duration of plain words, the second is responsible for words followed by punctuation. Overall the training process may last up to one day, depending on the size of the word database. For example, for MS SAPI we used a database containing 9.541 words and the networks took 22 hours to train.

BML Support

In RealActor we have implemented a draft 1.0 version of BML, which is the latest version of the specification [BML specification] proposed by BML community. RealActor supports most of the BML elements in Core BML. However, the parameterization of gesture extends the BML attribute specification. To provide simple control, we merged BML location attributes into a single vector attribute and added an attribute specifying which part of the head or arm is affected by the gesture. The parameterization also includes emblems, gestures which can be used interchangeably with specific words. In total, our animation parameterization has four dimensions:

- general - gesture type and literal meaning
- space-shape - hand shape and location at stroke-start point
- dynamics - intensity, amplitude, direction and movement trajectory during stroke phase
- timing information – specification of synchronization points for the BML scheduling component

In further reading we provide a list of BML elements with their parameterization which is supported in RealActor. For a more detailed explanation of their purpose, please read the BML specification.

Attribute type values

- **ID:** A unique ID in the character's world
- **name:** A string from a closed list of words
- **lexicalized:** A string from an open list of words (BML core specifies a few common ones)
- **bool:** A truth value, either *true* or *false*
- **int:** A whole number
- **float:** A number with decimals
- **angle:** A float number specifying angle in degrees
- **string:** An arbitrary string
- **direction:** A particular **name** type containing the words LEFT, RIGHT, UP, DOWN, FRONT, BACK, UPRIGHT, UPLEFT, DOWNLEFT, DOWNRIGHT

<locomotion>

Move the body of the character from one location to another. There are two distinctive types of usage: target-oriented and velocity-oriented. In RealActor, only target-oriented type is supported.

Attribute	Type	Use	Default	Description
Type	name	required		The type of locomotion specification (TARGET, VECTOR). VECTOR designates locomotion specifications using velocity vectors, TARGET designates locomotion specifications relative to a target object. Only TARGET is supported
target	vector, ID	required		In target mode: A named object, person or point in space that represents the end location of the locomotive behavior. The locomotion should end at a default “arrival location” relative to the target, which typically would be the “front” of the target.
facing	angle	optional	FRONT	In target mode, once the locomotion terminates, the character's body should

				be facing in this direction relative to the target. i.e. front facing would have the body completely face the target, but left facing would have the body be turned to the left (so that the right shoulder is now nearer the target).
manner	lexicalized	optional	walk	In target mode or velocity mode. The general manner of locomotion such as whether the character walks or runs. This can be extended through libraries while BML will include a few typical ones

Table 6: Locomotion attributes supported in RealActor

<gesture>

Coordinated movement of arms and hands, including simple gestures like beats and more complex parameterized ones.

Attribute	Type	Use	Default	Description
type	name	required		Category of the movement [POINT, BEAT, CONDUIT, GENERIC, LEXICALIZED]
hand	Name	required		What hand/arm is being used [LEFT, RIGHT, BOTH]
twoHanded	lexicalized	optional		How the two hands/arms of a two-handed gesture should be coordinated [mirror, alternate, parallel,…]
lexeme	lexicalized		required with LEXICALIZED	Refers to a group of animations or a method to procedurally realize this particular gesture
Shape				
handLocation	vector	optional		Location of the hand at stroke-start phase [(High, Center, Low),(Center, Inward, outward),(Contact, Near, Far)]
palmOrientation	direction	optional		The palm comes from this direction relative to the target, if one is specified. Otherwise, it's relative to the character. Values: forward (toward the hearer), backward

				(toward the speaker), inward (direction opposite of the hand which is used), upward, downward
handShape	lexicalized	optional		The shape of the hand [INDEXFINGER, THUMB, OPENHAND, FIST, FINGER-RING, CUP, PURSE]
fingerDirection	direction	optional		Refers to the orientation of the wrist by specifying where the finger is pointing, it's relative to the character.
Movement				
movementPart	lexicalized	optional		Which part of the arm/hand is involved finger[1-5], wrist, hand, forearm, arm
movementDirection	direction	optional		The general direction of the overall gesture movement relative to the character
trajectory	lexicalized	optional		The characteristic shape of the gesture movement through space [straight, curved, circular, rectangular, triangular, wave-like, zigzag,…]
amplitude	size	optional		Extra large, large, normal, short
power	power	optional		Refers to how powerfully the gesture is executed [WEAK, NORMAL, FORCEFUL]
repetition	integer	optional		Number of times the stroke phase is repeated

Table 7: Gesture attributes supported in RealActor

<head>

Movement of the head offset from posture and gaze direction. Types include nodding, shaking, tilting. Supports rhythmic movement.

Attribute	**Type**	**Use**	**Default**	**Description**
action	name	required	ROTATION	[ROTATION, ORIENTATION]
rotation	name	*required*		Category of head movement [X (NOD), Y (SHAKE), Z (TILT)]. Need to decide

				whether to use X, NOD or either. In core, we assume only 1 axis in an element.
amount	float	optional	0.5	The extent of the movement here 1.0 is fully extended and 0.0 is the least extended, and -1.0 is fully extended starting in the opposite direction. Need to define what positive direction is.
orientatio n	name	*require d*	FRONT	[FRONT,RIGHT,LEFT,UP,DOWN]

Table 8: Head attributes supported in RealActor

<face>

Movement of facial muscles to form certain expressions. Types include eyebrow, eyelid and expressive mouth movements.

Attribute	**Type**	**Use**	**Default**	**Description**
type	name	required	ROTATI ON	The part of the face being controlled. [EYEBROW, MOUTH, BLINK, FACS]. BLINK only requires amplitude, with negative opening the eye more, while EYEBROW requires the part and side of the face being specified.
amount	float	optional		[-1 to +1], Amount of movement, where 0 is no movement (or closed position), +1 is the highest (or most open) position), and -1 is the greatest possible movement in the opposite direction, if applicable.
side	float	optional	BOTH	[LEFT,RIGHT,BOTH] Specifies the side of the face being controlled, so for example left and right eyebrows can be controlled separately
direction	char	required		[X,Y,Z] X is sideways, Y is up/down and Z is away/inwards from the ideal *plane* of the feature.
Eyebrow part				
eyebrow_ part	name	optional	BOTH	[INNER,OUTER,BOTH] The part of the eyebrow. Note outer can only move in Y direction (up/down), while inner

				can move in X (sideways) and Y directions. See picture below.
Mouth part				
mouth_part	name	optional	CORNERS	CORNERS refers to the side corners of the mouth. CORNERS can move in both Y direction (raise/lower), and X direction (e.g. stretching the mouth).
FACS				
au	int	optional	0	(FACS only) The Action Unit (AU) reference number for a Facial Action Coding System (FACS) expression. The following AU are supported: AU1, AU2, AU4-AU32, AU38, AU39, AU43, AU45, AU46
au_duration	Vector, int	optional		Duration of action units. It may be a vector (attack, sustain, release), or overall duration NOTE: This does not exist in BML spec.

Table 9: Face attributes supported in RealActor

<gaze>

Coordinated multimodal movement indicating where the character is looking.

Attribute	**Type**	**Use**	**Default**	**Description**
type	name	required		Type of gaze behavior, either AT or AVERT (direct towards or avoid looking at the target respectively). AVERT is not supported
offsetangle	angle	optional	0.0	Adds an `angle` degrees offset to gaze direction relative to the `target` in the direction specified in the `offsetdirection`
offsetdirection	direction	optional	RIGHT	Direction of the offsetdirection angle [RIGHT, LEFT, UP, DOWN]

Table 10: Gaze attributes supported in RealActor

<speech>

To generate ECA speech, two methods can be used: speech synthesis with TTS engine and lip sync, which uses audio files and generates on-thy-fly viseme events using speech processing techniques. In both modes, a developer can define time mark points and align animations with them. Whereas in TTS mode time mark points are scheduled on the fly (animations are aligned with the word which comes after time mark point), in lip sync mode, the developer has to manually add timings of alignment points. The following are examples in BML code:

TTS mode

```
(1)
<speech id="s1">
<text>Hello, <sync id="tm"/> my name is  Dubravka. </text>
</speech>
```

Lip Sync mode

```
(2)
<speech id="s1">
<file ref="Intro.wav">
<text> Bok, moje ime je  Dubravka. Ja cu biti vas turisticki vodic
po  Dubrovniku.
<sync id="tm" time="2.2"/>
</text>
</file>
</speech>
```

In the first example, animations will be aligned with “tm” point on word “my”, and in the second example animations will be aligned at 2200 millisecond-mark of the audio file.

<namespace>

Our system supports external animations which could not be described with Core BML. These are reasonably complex behaviours performed by several body modalities simultaneously. The animations are modelled in 3ds Max and they can be called using BML namespaces. For the system, we define two namespace tags:

```
<ra:face category="expression" id="ex2" type="joy" intensity="0.8"/>
<ra:body category="gestures" id="g2" type="pray"/>
```

The first one represents complex facial expressions which are not defined in Core BML: The second namespace calls body motions, mostly gestures which may be even culturally codified (e.g. Japanese banzai).

Attribute	Type	Use	Default	Description
		RA: Face, Body		
category	name	required		Category of action, for Face[Expression], for Body[Gesture]
type	name	optional		Type of action Face[Expression(joy, wink…)], Gesture[bow, pray, wave, thinking, scratchHead, invite, warning, armsExtension, banzai…]
intensity	int	optional		Intesity of action [1,2]

Table 11: Definition of namespaces in RealActor

<emit> and <feedback>

Emit tags serve to send a message when a particular event happens, e.g. at the end of animation execution:

```
<bml id="bml1">
   <gesture id="g1" type="POINT" target="chair1"/>
   <emit id="emitter1" stroke="g1:end">
      <event id="event1">
         Optional information.
      </event>
   </emit>
</bml>
```

Events are generated through a virtual function the purpose of which is to synchronize specific events in the application.

Feedbacks notify about any BML performance requests and warnings. Feedbacks are useful to track progress of behavior realization. RealActor architecture supports two types of feedback: Start feedback and Stop feedback.

Animation extension and adding audio files

Developers may introduce their own example animations through RealActor library. The animations need to be in FBA file format, which can be exported from 3ds Max using appropriate plug-ins. In order to add the new animations into the system, the developer should modify RealActor data folder which contains all animations and other RealActor data. The modification is done in three steps:

1. Define animation category, which corresponds to BML element (gesture, head, extern animation). Also define values of that element.
2. Create a new subdirectory in *data/RealActor/animation* directory and place the animation there. We suggest a meaningful name for the directory, such as “clapping” for clapping animations.
3. Locate the animation timing file (it is in *data/RealActor/animation/timings* directory) and add new values (BML parameters and path to the animation file, which was created in the second step)

To realize the newly-added animation, the developer should create a BML script and insert a tag with BML values which were defined in the first step (element and its parameters). The animations can by synchronized with other behaviours using standard reference points.

To realize speech using audio files (through lip-synching), a developer should perform similar actions:

1. Place the audio file into *data/RealActor/audio/speech* directory.
2. Modify the speech timing file ins *data/RealActor/audio/* directory and add the name of the new audio file and its duration

To animate a character using the audio file, a developer needs to create a BML script with a tag which defines the name of that script (see <speech> tag specification in this manual)

BML Script Examples

The following are two examples of scripts which explain how BML elements are coordinated with one another.

1. A script that contains two BML blocks which describe two utterances: in the first one the character's voice is generated using TTS engine, and in the second one using an audio file.

```
<?BML version="1.0"?>
<act>
<!--Specification of nametags-->
<ra:animation xmlns:bm="extern">
<!—-the first bml element-->
<bml id="bml5">
      <!—-walk to the point [0.7,0,4.8] in 3D space, facing front-->
      <locomotion id="l1" type="target" manner="walk"
      target="0.7;0.0;4.8" facing="front"/>
      <!—-move head (nodding)-->
      <head id="h2" action="rotation" rotation="nod" amount="0.1"/>
      <!–start speaking (using tts)-->
      <speech id="s1"><text>Hello, my name is <sync id="tm"/>
      Dubravka. </text></speech>
      <!—-start gesturing, stroke phase of gesture is on word
      „Dubravka“-->
      <gesture id="g1" stroke="s1:tm" type="beat" hand="both"
      handshape="cup" movementdirection="forward"/>
      <head id="h1" start="s1:tm" action="rotation" rotation="nod"
      amount="1"/>
      <!—-blink when speech is finished-->
      <face id="f2" start="s1:end" type="blink" amount="0"/>
      <!—-emit event when gesture is finished-->
      <emit id="emitter1" start="g1:end">
            <event id="event1">
            Optional information.
            </event>
      </emit>
</bml>
<!—-the second bml element-->
<bml id="bml8">
      <!—-move head down when ex2 is started-->
      <head id="h1" start="ex2:start" action="orientation"
      orientation="down" amount="1"/>
      <!—start speaking (from audio intro.wav)-->
      <speech id="s1">
            <file ref="Intro.wav">
            <text> Bok, moje ime je Dubravka. Ja cu biti vas
            turisticki <sync id="tm" time="2.3"/> vodic po
            Dubrovniku. </text>
            </file>
      </speech>
      <!—-move eyebrows when speech is started-->
      <face id="f3" start="s1:start" type="eyebrow" direction="y"
      amount="0.7"/>
<!—-make an expression of joy when speech is started-->
      <bm:face start="s1:tm" category="expression" id="ex2"
      type="joy" intensity="0.8"/>
</bml>
</ra:animation>
</act>
```

2. A script that does not contain any speech. It only explains the usage of Action Units in BML code.

```
<?BML version="1.0"?>
<act>
<bml id="bml5">
      <face id="f2" type="facs" au="1" amount="0.5"
      au_duration="400;600;400"/>
      <face id="f3" type="facs" au="6" amount="0.7"
      au_duration="200;2200;500"/>
      <face id="f4" start="f2:end" type="facs" au="12" amount="0.6"
      au_duration="1000"/>
</bml>
</act>
```

Selected Publications

Journal Papers

- Čereković, Aleksandra; Pandžić, Igor. Multimodal Behavior Realization for Embodied Conversational Agents., Multimedia tools and applications. (2010) (accepted)
- Čereković, Aleksandra; Pejša, Tomislav; Pandžić, Igor. A Controller-based Animation System for Synchronizing and Realizing Human-like Conversational Behaviors., Development of Multimodal Interfaces: Active Listening and Synchrony, Lecture Notes in Artificial Intelligence. (2010), 1, 80-91
- Huang, Hsuan-Hung; Čereković, Aleksandra; Pandžić, Igor; Nakano, Yukiko; Nishida, Toyoaki. Toward a multi-culture adaptive virtual tour guide agent with a modular approach., AI & Society, Journal of Knowledge, Culture and Communication. 24 (2009), 3; 255-235

Conference Papers

- Čereković, Aleksandra; Huang, Hsuan-Hung; Furukawa, Takuya; Yamaoka, Yuji; Pandžić, I; Nishida, Toyoaki; Nakano, Yukiko. Implementing a Multi-user Tour Guide System with an Embodied Conversational Agent, the Proceedings of International Conference on Active Media Technology, Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen (ur.). Bejing : Springer, 2009. 7-18
- Huang, Hsuan-Hung; Furukawa, Takuya; Ohashi, Hiroki; Čereković, Aleksandra; Yamaoka, Yuji; Pandžić, Igor; Nakano, Yukiko; Nishida, Toyoaki. The Lessons Learned in Developing Multi-user Attentive Quiz Agents, Proceedings of 9th International Conference, Intelligent Virtual Agents 2009, Ruttkay, Zsofia ; Kipp, Michael ; Nijholt, Anton ; Vilhjalmsson, Hannes Hogni (ur.). Amsterdam, Netherlands : Springer, 2009. 166-173

- Huang, Hung-Hsuan; Čereković, Aleksandra; Nakano, Yukiko; Nishida, Toyoaki; Pandžić, Igor S. The Design of a Generic Framework for Integrating ECA components, the Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, Padgham, Lin ; Parkes, David ; Müller, Jörg ; Parsons, Simon (ur.). Estoril, Portugal : Academic Press, 2008. 128-135
- Čereković, Aleksandra; Huang, Hung - Hsuan; Zorić, Goranka; Levačić, Vjekoslav; Pandžić, Igor; Nakano, Yukiko; Nishida, Toyoaki. Towards an Embodied Conversational Agent Talking in Croatian, Proceedings of the 9th Interantional Conference of Telecommunications ConTEL 2007

Summary

This thesis studies Embodied Conversational Agents (ECAs) and the components which comprise the ECA system.

ECAs are computer-generated virtual characters who express human-like abilities in interaction with humans: they speak, make gestures, reason about ongoing conversation and adapt it to their conversational goal. The process of making ECAs is challenging because it is multidisciplinary and requires fundamental advances in speech recognition, natural language understanding and generation, dialogue management, cognitive modelling and reasoning, virtual human architectures and computer graphics and animation. Therefore, present research evidences how research laboratories cooperate in order to create effective and realistic ECAs. Besides, ECA community attempts to set specifications and encourage researchers to exchange their ideas and ECA system components.

We contribute to the vast area of ECA research by devising RealActor, an open-source character animation and behaviour system with support for modelling multimodal communication using Behaviour Markup Language (BML). The system incorporates synching mechanisms which realize multimodal utterances in real-time using animation examples, procedural animations, and techniques of motion blending. We present a novel solution for aligning animations with synthesized speech and explain our Facial Action Coding system (FACS)-based animation model modelled using techniques of face tracking.

Due to scarcity of research on multiuser-ECA interaction we propose new approaches to development of multiuser support in an ECA system, experiment with different equipment for human-machine interaction and study impressions of interacting with such a system.

Keywords: Embodied Conversational Agents (ECAs), multimodal behaviours, realization of communicative utterances, multiuser interaction

Sažetak

Ova doktorska disertacija se bavi sustavima utjelovljenih razgovornih agenata (engl. *Embodied Conversational Agents*) i komponentama koji čine te sustavu.

Utjelovljeni razgovorni agenti naziv je za računalno generirane virtualne likove koji mogu komunicirati sa korisnikom. Pri komunikaciji oni pričaju, gestikuliraju i rasuđuju o tijeku konverzacije te je adaptiraju svojim konverzacijskim ciljevima. Proces stvaranja razgovornih agenata je zahtjevan jer je to multidisciplinarno područje i zahtjeva temeljna znanja o obradi govora razumijevanju i stvaranju jezika, vođenja dijaloga, modeliranju svijesti i rasuđivanju, arhitekturama virtualnih ljudi i računalnoj grafici i animaciji. Stoga, istraživanja koja su u tijeku se najčešće izvode u suradnji nekoliko istraživačkih laboratorija kako bi se stvorili vjerodostojni i realistični razgovorni agenti.

Ova disertacija doprinosi širokom području utjelovljenih razgovornih agenata sa sustavom RealActor, otvorenim sustavom za ostvarivanje ponašanja koji ima ugrađenu podršku za modeliranje ponašanja uz pomoć jezika BML (engl. *Behaviour Markup Language*). Sustav se sastoji od mehanizama za sinkronizaciju koji ostvaruju multimodalna ponašanja u stvarnom vremenu i to koristeći primjere animacija, proceduralne animacije i tehnike miješanja animacija. U disertaciji je predstavljeno rješenje za poravnanje animacija sa sintetiziranim govorom i animacijski sustav lica temeljen na sustavu FACS (engl. *Facial Action Coding System*), a koji je modeliran uz pomoć tehnika praćenja lica.

Budući da je to nedovoljno istraženo područje, u disertaciji također predlažemo novi pristup izgradnji podrške za višekorisničku komunikaciju u sustavima sa razgovornim agentima, eksperimentiramo sa opremom koja se koristi u interakciji sa računalom te proučavamo ponašanje takvih sustava u interakciji sa korisnicima.

Ključne riječi: utjelovljeni razgovorni agenti, multimodalna ponašanja, ostvarivanje komunikacijskih ponašanja, višekorisnička interakcija

Curriculum Vitae

Aleksandra Čereković was born in Mostar, Bosnia and Herzegovina, on 25th of April in year 1982. She finished "Nova Gimnazija" high-school in Mostar in year 2001. The same year she continued her education at the Faculty of Electrical Engineering and Computing (FER), University of Zagreb, Croatia. In year 2005 she has joined NG-ECA project, which is a collaborative research work of Department of Telecommunications (FER), Graduate School of Informatics (University of Kyoto) and Faculty of Science and Technology (Seikei University, Tokyo). She graduated in 2006, receiving a diploma of B. Sc. E.E. for her research-oriented thesis. The same year she co-organized eNTERFACE ’06 workshop on Multimodal Interfaces in Dubrovnik (17/07 – 11/08/2006, Dubrovnik) where she participated in the project “An Agent Based Multicultural Customer Service Application”, working with NG-ECA researchers. In October 2006 she has started working as a research assistant at Department of Telecommunications (FER) and she has started PhD studies. As PhD student she led the project “Multimodal communication with a tour guide ECA” at eNTERFACE ‘08 workshop in Paris (04/08 – 29/08/2009, Paris) and she participated in two international schools, COST 2102 spring school in Dublin (04/08 – 29/08/2008, Dublin) and PEACH summer school in Dubrovnik (09/07 – 11/07/2008, Dubrovnik).

She works as a researcher at project “Embodied conversational agents as interface for network and mobile systems” which is funded by the Croatian Ministry of Science, Education and Sports (MZOS) and at COST project, funded by European committee (COST 2102:„Cross-Modal Analysis of Verbal and Non-verbal Communication“). She has co-authored over 20 research papers. Her research interests are multimodal communication, virtual character animation and embodied conversational agents.

Životopis

Aleksandra Čereković rođena je 25.4.1982. u Mostaru. Srednju školu završila je u Novoj gimnaziji u Mostaru 2001. godine. Iste godine upisuje Fakultet elektrotehnike i računarstva u Zagrebu. Kao student 2005. godine uključuje se u istraživački projekt o utjelovljenim razgovornim agentima, NG-ECA, koji su izvodi u suradnji Zavoda za telekomunikacije na Fakultetu Elektrotehnike i računarstva, Graduate School of Informatics na Sveučilištu u Kyotu i Faculty of Science and Technology na Seikei sveučilištu u Tokyu. Diplomira 2006. godine sa naglaskom na znanstveno-istraživački rad. Iste godine kao student i koorganizator sudjeluje u međunarodnoj radionici eNTERFACE '06 (17.07. – 11.8.2006., Dubrovnik) na kojoj radi sa istraživačima iz projekta NG-ECA. U listopadu 2006. godine zapošljava se na Zavodu za telekomunikacije na Fakultetu elektrotehnike i računarstva i upisuje poslijediplomski doktorski studij elektrotehnike. Doktorsku disertaciju obranila je 8.7.2010. Kao student doktorskog studija sudjelovala je u projektu radionice eNTERFACE '08 (04.08. – 29.08.2008. u Parizu, vođa projekta) te u dvije međunarodne škole, projekta COST 2102 (23.03.-27.03 2009., Dublin), i projekta PEACH (09.07 – 11.07.2008., Dubrovnik).

Surađuje na istraživanjima projekta Ministarstva znanosti, obrazovanja i športa Republike Hrvatske, kao i na COST projektu Europske komisije (COST 2102: „Cross-Modal Analysis of Verbal and Non-verbal Communication“, od rujna 2006. do studenog 2010.). Koautor je preko 20 znanstvenih radova. Njeni interesi su višemodalna komunikacija, animacija virtualnih ljudi i utjelovljeni razgovorni agenti.