Virtual Reality (2003) ■■: ■■■–■■■
DOI 10.1007/s10055-003-0113-x

ORIGINAL ARTICLE

Martin Klesen · Michael Kipp · Patrick Gebhard
Thomas Rist

Staging exhibitions: methods and tools for modeling narrative structure to produce interactive performances with virtual actors

Received: ■ / Accepted: ■

Abstract CrossTalk is a self-explaining virtual character exhibition for public spaces. This paper presents the CrossTalk system, including its authoring tool SceneMaker and the CarSales exhibit. CrossTalk extends the commonplace human-to-screen interaction to an interaction triangle. The user faces two separated screens inhabited with virtual characters and interacts through a frontal touch screen. One screen features the exhibition's hostess, an agent who explains exchangeable exhibits located in the opposing screen. The current exhibit is CarSales, a demonstration of automatically generated dialogue, performed by virtual actors. The physical presence of the characters is established through the separation of screens and intensified by inter-character conversations across screens, tying hostess and exhibit together. CrossTalk utilizes a combination of both automatically generated and pre-scripted scenes, and a context memory to adapt to the user and the environment. CrossTalk's authoring tool SceneMaker, in a strict separation of narrative structure and content, provides non-experts with a screenplay-like language to create installations for staging exhibitions.

Keywords Authoring · Believability · Embodied agents · User adaptivity · Virtual theater

1 Introduction

In the early days of TV commercials a human presenter promoted a product by enumerating advantageous features. Over the last few decades this presentation style has almost completely been replaced by formats that draw on short, entertaining episodes. Such performances embed product information in a narrative context acted out by human actors. These episodic formats do not only meet the commercial industry's high demand for originality, they can also anticipate typical questions and concerns of the customer.

In the field of human-computer interaction, where artificial characters act as virtual presenters or tutors, the single character format still prevails [1–3]. At DFKI, we have proposed a shift from single character settings towards presentation teams [4, 5]. Using teams for infotainment purposes bears a number of advantages. Product information can be conveyed in a less obtrusive and more effective way. Characters can utilize complementary personalities/roles to impose a visible, enacted structure on the presented information. For example, by personifying different attitudes towards the product (pro vs. con, technical vs. fun) or representing varying levels of expertise (expert vs. non-expert).

Infotainment is information *and* entertainment. Using multiple characters relieves the user to some extent from the pressure to interact. By watching the characters interact with each other, users becomes familiar with the system before initiating their own interactions. Reducing interaction inhibition is highly important for installations that are meant for public spaces that should attract passers-by.

CrossTalk is a 24-hour entertainment installation with virtual characters that are 'alive' around the clock. It provides visitors with a specially extended interaction experience by offering two virtual spaces on separate screens, one displaying Cyberella, the installation's hostess, the other displaying Tina and Ritchie, two virtual actors (Fig. 1). Together with the visitor console up front, it creates an interaction triangle. It was conversations between virtual characters that cross the limits of physical screens inspired the name 'CrossTalk'. It has been conceived as a generic exhibition framework to 'stage' technology in the field of embodied characters by embedding an exhibit seamlessly into an overall story/scenario. The installation needs no personnel – it is self-explanatory and runs in an endless loop. The carefully chosen overall scenario, the multitude of pre-scripted scenes (250 in English/German each), and the large

M. Klesen (✉) · M. Kipp · P. Gebhard · T. Rist
DFKI GmbH Saarbrücken, Germany
E-mail: {klesen, kipp, gebhard, rist}@dfki.de

1 0 0 5 5 | 1 1 3 | B
Journal number | Manuscript number
Dispatch: 9.9.2003 | Journal: Virtual Reality | No. of pages: 13
Author's disk received ☑ | Used ☑ | Corrupted ☐ | Mismatch ☐ | Keyed ☐

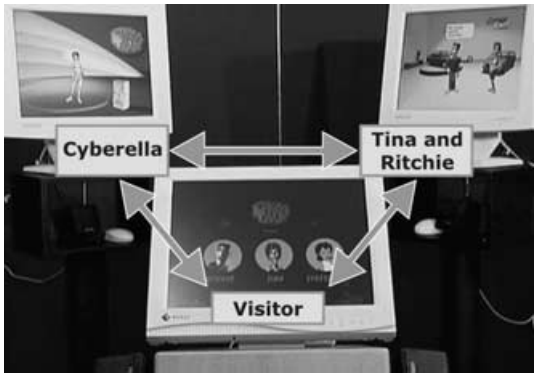

Fig. 1 Main components and spatial layout of the CrossTalk installation

repertoire of character gestures made this installation very successful, attracting crowds of people and letting visitors watch/interact for even half an hour without detecting repetitive patterns.

CrossTalk is interactive. It gives the impression that the characters are aware of their environment and respond appropriately to user input. A context memory stores situational and dialogue context (discourse history, location, time, etc.) and evaluates visitor interactions to trigger context-sensitive scenes (Sect. 3.4). Using short- and long-term context combined with carefully authored scenes is our approach towards emergent narrative [6].

Actors, whether they are virtual or human, need a script and directions telling them what to do. Writing a good script is hard, especially when having to include character actions, context knowledge and user interaction while maintaining story structure and creating an 'illusion of life'. Also, shifting the whole system to new scenarios or domains usually requires starting from scratch again. In CrossTalk we address these problems by providing the SceneMaker authoring suite.

SceneMaker is an authoring system for storytelling systems (Sect. 3). Its central paradigm is the separation of narrative structure from story content, which, from an author's viewpoint, facilitates controlling story structure and consistency. Interactive performances are authored in a two-step approach. First, the scene flow (i.e. the transitions between scenes) is defined. Secondly, the content of each scene must be provided, either by a human author using a screenplay-like scripting language (Sect. 3.1) or by automatically generating scenes at runtime using plan-based dialogue generation (Sect. 4.3). This hybrid approach has a number of advantages. It allows the use of pre-scripted scenes where automatic generation fails, for instance, in situations of humor, irony, or small talk. It also supports incremental development. Sample scenes, meant to be automatically generated at a later stage, can be pre-scripted and run as pre-test simulations, e.g. to check out different parameter settings, even before the domain model and the dialogue strategies have been fully implemented. Thus, SceneMaker supports rapid prototyping and serves as a modeling and preview tool.

The rest of this paper is organized as follows. Section 2 describes the CrossTalk framework. The SceneMaker tool for authoring CrossTalk is explained in Sect. 3. Section 4 presents CarSales, a CrossTalk exhibit that demonstrates our plan-based approach to automatic dialogue generation. In Sect. 5 we survey related work, systems and authoring tools, in the field of interactive narrative and close with a summary.

2 The CrossTalk framework

2.1 The experience

CrossTalk's triangular layout has two screens located in the background and a touch-screen in the middle for visitors (Fig. 1). The screens show Cyberella, the exhibition's hostess, and the two virtual actors, Tina and Ritchie, who were hired to act out a demo performance under Cyberella's direction. This demo performance represents the 'exhibit' of the installation (Sect. 4). CrossTalk runs in two modes. The OFF mode is assumed when no user is present, whereas a user starting the installation lets the system switch to ON mode.

2.1.1 No User: OFF

When no user is present the installation is in OFF mode. In this mode, passers-by are attracted by the ongoing 'life' in the background screens, while the third front screen features a large, pulsating 'start' button. In OFF mode, all three characters perform idle actions like breathing, looking around or shifting posture. They also chat with each other and even start rehearsing parts of the performance from time to time.

Tina: How are you gonna get home?
Ritchie: Found a pretty good Internet connection. Rates are a real bargain after 8pm. Two cents per minute.
Tina: That's still loads of money considering my luggage.
Cyberella: Do you have a lot?
Tina: You bet. At least twenty megabyte.
Cyberella: Wow.
Ritchie: Women!
Tina: Excuse me?
Ritchie: Women can't pack. Simple as that. Ever heard of WinZip?

2.1.2 User arrives: ON

As soon as a passer-by enters the installation by pressing the 'start' button, all characters interrupt their current activities. Cyberella welcomes the user and offers a demo performance of automatic presentation generation enacted by Tina and Ritchie. She guides the user through parameter selection menus where the user can choose

roles, personalities and issues for the following demo. The demo itself consists of a car sales dialogue acted out by Ritchie and Tina that is automatically generated based on the user-selected parameters (Sect. 4.3).

Ritchie: Welcome! My name is Smith. What can I do for you?
Tina: I'm thinking about buying this car.
Ritchie: This car is a real bargain.
Tina: Does it have airbags?
Ritchie: It certainly has.

During the demo performance, the user's screen shows three feedback buttons that read 'applause', 'boo' and '?'. If the user pushes one of these, an adequate reaction is interwoven into the running demo. For example, if the user presses the 'applause' button, the following may happen:

Ritchie: For the love of Pete!
Tina: What is it?
Ritchie: I can't do it. I just can't. It's driving me nuts!
Tina: What can't you do? What?
Ritchie: I I just can't keep people from admiring me.
Tina: Oh boy.
Cyberella: Go on.

After the demo, Cyberella suggests watching another demo and prompts the user to choose new parameters. Alternatively, the user can leave, an option that is available for the whole time of the interaction.

2.1.3 User leaves: OFF

When the user leaves, the installation switches back to OFF mode. 'Life' as usual resumes, the characters start waiting, chatting and rehearsing. Not only will new passers-by be attracted: by occasional reference to the past user interaction, the former user's attention might be caught again – the user is 'bound'. We call this: attract & bind.

2.2 The principles

CrossTalk has been presented on various occasions, most prominently at the world's largest computer fair, the CeBIT 2002 in Hannover, where it attracted crowds of people watching and interacting. In this section, we explore some issues that may have been responsible for CrossTalk's success.

2.2.1 Illusion of life

Like many other systems based on artificial characters, including motion pictures, a major aim of CrossTalk is to create the illusion of life [7]. Real people have a rich background of life experience that is referred to almost all the time. Although artificial characters lack this rich background, with a number of tricks we can deceive the user into believing that such a background exists. In CrossTalk, we suggest a rich technical, geographical and cultural background by different means.

The illusion that the agents live in different rooms or 'spheres' is strongly supported by the physical separation of screens. The user immediately feels that the two screens are windows to separate rooms which breaks the usual human vs. computer preconception and thus, feels more like a video conference or meeting in a group, both situations usually encountered in an all-human context.

To make the user feel that these characters are really alive, a whole bag of tricks is used. The most important one is that of role and meta-role. Nowadays, all users are aware that animated agents are programmed to do what they do. Thus, a concept of the agent as a puppet playing some preconceived role prevails in most users' heads. In CrossTalk, this preconception is systematically exploited. By building the meta-roles of 'actors' around the role we trick the user, who does not really believe in the role being real, into believing that the meta-role is real. The characters openly admit that they only play their role, that they are 'in reality' actors performing something a programmer has thought up. Of course, this meta-role of the agents being actors is just as well the product of programming. And although this is easily understood by users when thinking about it, the commonplace simple illusion of life is here 'padded' by another layer of life, the meta-role. Note that this technique has been employed in real theater as well to great effect and still is. Actors stepping out of their roles convey a feeling of authenticity, temporarily firing the attention of the audience by making them believe that the actor is speaking by and for himself – although this part is just as preconceived as the rest of the play, the actor just having switched from role to meta-role. The trick is to systematically counteract audience expectations so that 'life' and 'authenticity' are taken for granted on an intuitive level without deeper reflection. In CrossTalk, the character play out their meta-role when no user is present (OFF mode), creating a background of life. In this OFF mode, references to their actual roles are plentiful, most prominently so when they start rehearsing a scene. During rehearsal, characters are in their role creating a sphere of 'life in the role'. This illusion is immediately broken by the actors forgetting lines or asking for an Aspirin. Again, expectations have been countered. When in ON mode, the characters again assume their roles playing the 'performance' of a sales dialogue for the user. While the user is watching and giving feedback, the characters again step out of their roles from time to time to react to user feedback, for instance by bowing to the user or whispering confused remarks to each other.

2.2.2 Agent-world relations

Beyond the meta-theatrical framework of CrossTalk the illusion of life can be fuelled by referring to the user's

cultural background like well-known movies (*Star Wars, The Godfather*) or famous pop stars (Britney Spears, Madonna). These references as well as knowledge on situational context, i.e. the current time of day, year, season and the location (city, country, building), that is occasionally sewn in the dialogue, make the agents inhabitants of the user's world. The illusion is that of cohesion between user context and agent context. At the same time, to make them believable entities, some humorous technical remarks make the agents beings of a technical world where people live on hard disk sectors, travel via Internet connections and measure their time in CPU cycles. The illusion is that of a 'world inside the computer' where things are just a little different. While the two concepts, agents as beings of 'this world'' and agents as beings of a 'world inside the computer', seem contradictory as first, they work extremely well together, both metaphors supporting different illusions.

2.2.3 Agent-installation relations

CrossTalk now extends the illusion beyond the sphere of single agents to encompass the whole installation, which seems to be under full control of the agents. Not only is the installation presented and explained by an agent, also, different technical commands like ''could you fix the lights, Cyberella?'' (after the screens went black) or ''we have a wrong background picture on our screen, Cyberella'' suggest that the agents can themselves fix technical problems. In one highly dramatic scene, error messages appear on all screens while Ritchie in an act of selfless heroism announces that he will 'hack into the system' and disappears. A second later the error messages disappear and Ritchie re-appears. Everything is, again, under control.

2.2.4 User-system relations

While the issues touched so far concerned solely the agents and the installation, one hugely important aspect is the user-system relationship. The user must have the illusion that the characters are aware of him/her and, moreover, that s/he has an impact on the future course of things, as in any bidirectional relationship. The illusion of user awareness in CrossTalk is strongly fostered by the physical layout. Cyberella addresses not only the user, but also another physically present entity: the screen of the actor agents. The impression that Cyberella really communicates with something outside her screen is thus given twice. In the interaction with the user Cyberella also refers to user selections ('good choice', 'interesting selection') to make the parameter selection part of a social interaction. Whereas such reactions are immediate and quite common in human-agent systems, more subtle, still short-term interactions give an even stronger impression of user-agent relationship. When the user gives feedback like applause or booing during a demo performance, the actor agents (Tina and Ritchie) systematically step out of character, whispering consolations to each other, addressing the user (''thank you!'') or asking Cyberella for advice. Sometimes, responses to the user interaction (a smile, a wink, extra lines) are seamlessly integrated into the demo, the actors remaining in their roles, so that the user will take a while noticing them. On a even larger time scales, references to user behavior, how often s/he pressed 'applause' and 'boo' are made by actors and Cyberella either in ON mode or later on, while chatting. These are long-term user-system effects that are unusual and strongly binding. This conforms to the claim that user-system interactions should occur on a range of various *interaction time scales* [8].

2.3 The system

In this section we give an overview of both the offline and online processes that make a CrossTalk presentation run. We first present an author's 'work flow' for creating presentations in CrossTalk. Then, we give a concise description of CrossTalk's distributed architecture, the underlying modules and the control flow.

2.3.1 Workflow

Staging exhibitions in CrossTalk consists of two major phases, one offline, one online. In the offline phase a human author creates the presentation with the help of the SceneMaker authoring suite. In the online phase the presentation is run by the CrossTalk system.

In the offline phase a complex presentation is created using the SceneMaker tool. The most basic unit in authoring is a scene. Transitions between scenes are defined in a scene flow graph. A performance is defined as a number of scenes plus corresponding scene flow.

The creation of a performance proceeds in four steps. In step one the narrative structure is defined using scene flow graphs, a formal language in XML syntax. In step two dialogue content is written in a screenplay-like language (see Sect. 3 for a detailed account on authoring). In step three the pluggable exhibit is created by defining a scene generation package that contains dialogue strategies and domain model (see Sect. 4 for CarSales as a sample exhibit). In step four the results of all previous steps are compiled to a plan-based programming language by the SceneMaker tool. The final program is processed by the running CrossTalk system, described in the next section.

2.3.2 Architecture

CrossTalk's three separated screens are driven by three computers that are integrated by a distributed software architecture. Different CrossTalk modules (Fig. 2) communicate via typed messages using the InfoRouter

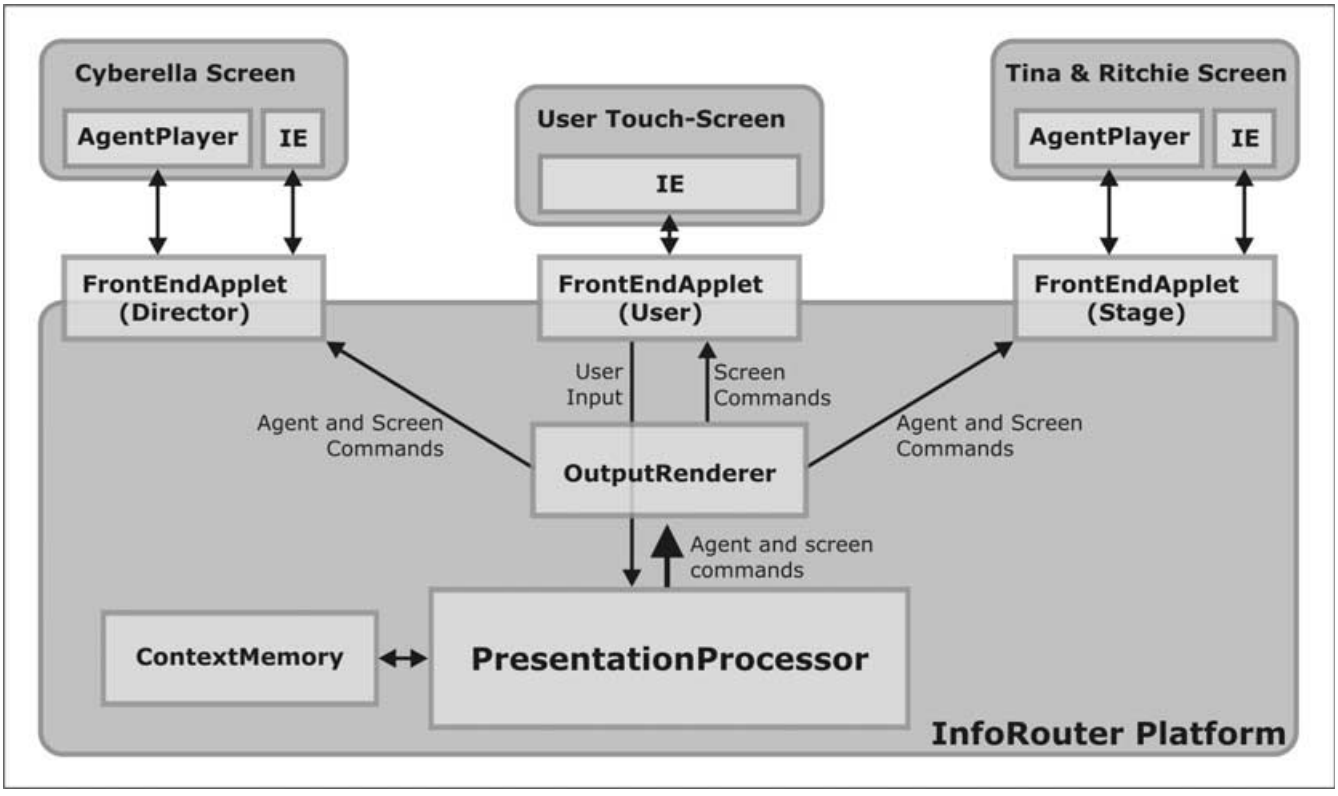

Fig. 2 The CrossTalk architecture

communication platform. The InfoRouter provides reliable messaging in a multi-blackboard paradigm.

CrossTalk's central module, the PresentationProcessor, is responsible for selecting and executing scenes. Executing scenes consists of forwarding commands for character and screen control to the OutputRenderer module on the one hand, and handling user and system events on the other. Internally, it runs a BDI-theoretic plan interpreter called JAM [9]. Similarities between CrossTalk's scene flow model and JAM's priority-based execution model allow a natural translation by the SceneMaker compiler. Also, because of JAM's high-level syntax, the domain model and the dialogue strategies for the plugged-in exhibit (Sect. 4) can easily be implemented by a human programmer.

The ContextMemory module stores the discourse history (user/system actions, played scenes and interactions) and situational context (e.g. day time or location) [10]. The context memory also creates log files for offline evaluation.

The OutputRenderer and FrontEndApplet modules implement the technical front end of CrossTalk. The OutputRenderer maps script commands used in authoring (e.g. gestures) to device-specific commands for character player, speech synthesis and screen control. Adding this abstraction layer to isolate output rendering devices makes exchanging these devices possible without changing the authoring language. Each screen has an instance of the FrontEndApplet running in an Internet browser. It displays screens and menus, runs the character player (Microsoft Agent[1]) and handles user input.

3 The SceneMaker authoring tool

The SceneMaker tool allows non-expert users to write content (scenes) and define the narrative structure (scene flow) on a level both abstract and intuitive. In an offline process SceneMaker takes content and narrative structure to produce the plan-based program that runs the CrossTalk system.

3.1 Authoring content

Authoring in CrossTalk is based on the concept of *scenes*. From the point of view of the system, scenes are pieces of user-edited contiguous dialogue[2]. From the point of view of the author, a scene is usually a coherent and closed unit regarding either a message, agent characterization or a humorous punchline. Technically, we call such scenes *atomic scenes*.

When defining the scene flow the author can define larger units that consist of linked atomic scenes. Such larger units are called *compound scenes*. Compound scenes can themselves become part of higher-level compound scenes. The resulting subsumption hierarchy is a

[1] http://www.microsoft.com/msagent

[2] Single utterances can also be scenes, not being a dialogue in the strict sense.

natural concept in authoring, comparable to paragraphs-sections-chapters in written text or scenes-acts in theater plays.

To enable the user to write a number of variations of an atomic scene, CrossTalk allows the definition of *scene groups*. A scene group contains a set of atomic scenes that are equivalent insofar that any of these scenes can be played when the scene group is activated in the course of scene flow traversal.

Authoring in CrossTalk is like writing a screen- or theater play where the writer creates text and stage directions. Directions include character actions that draw on a large repertoire (about 35 actions per character). These actions, mainly gestures, were empirically derived from TV material [11].

A simple authoring syntax is meant to appeal to all non-programmers, including professional playwrights. The author writes the protagonists' utterances and includes stage directions in the form of bracketed commands (Fig. 3). The script can be written in any text processing software. Possible actions come in four categories: gestures (G), facial expressions (F), posture shifts (P) and actions (A). Gestures were taken from a catalogue derived from analyzing a German TV show with manual gesture annotation [11–13]. The two analyzed speakers were found to have a shared lexicon of 69 conversational gestures. The most frequently used ones were modeled for CrossTalk by a professional graphics and animation designer. Some facial expressions were added based on the needs of the script (e.g. smile, skeptical). Posture shifts (P) are used to visually separate role (salesperson/customer) and meta-role (actor) of Tina and Ritchie: when they are 'themselves as actors' they have a relaxed body posture whereas when they are playacting they straighten up and look more tense. This is in accord with observations in psychotherapy where body posture was found to indicate separate topics or points of view in a conversation [14]. Other actions (A) comprise turning the head, breathing, and other idle-time actions like putting on glasses, yawning, etc.

CrossTalk's agents offer a large repertoire of about 35 actions each. For Tina and Ritchie each action was modeled twice, once for each of the two basic body postures. This opens a wide range of possibilities to visually support the conversation. Other systems are more focused and restricted in their approach to gestural modeling. For instance, the REA agent [3] uses seven gestures and the Jack Presenter [2] as well as the Papous Storyteller agent [15] use four gestures each.

Scene:	OFF-Chat stage-direction
...	...
Ritchie:	[TINA AS_LookLeft] Ok, if you are interested leave me your number. [V_LookToCy]
Tina:	Well, <Pau=300> ok. [RITCHIE V_LookToActor] Sounds ... great. [AS_Glasses] I'll think about it.
Cyberella:	[GS_Chide] My agent will contact you.
Ritchie:	Yeah. Sure. [GS_DoubtShrug] All right.

Fig. 3 Pre-scripted scene

3.2 Authoring narrative structure

Having written the content the author can define the narrative structure by linking the scenes in a graph called *scene flow*. Technically, we use cascaded Finite State Machines (FSMs) to represent the scene flow. A cascaded FSM consists of nodes and edges (transitions). Scenes can be attached to both nodes and edges. As mentioned above, scenes can be either pre-scripted or automatically generated. Additionally, nodes and edges can trigger system commands that, e.g. access global variables or the context memory. This extends scripting possibilities as it allows making scenes user-adaptive (Sect. 3.4).

3.2.1 Scene nodes

A node represents a state in a performance. It can have a scene attached that is invoked by the system at runtime. There are two types of nodes:

1. *Scenenode*: represents a state in which a pre-scripted scene is performed.
2. *Supernode*: represents a state in which a pre-scripted or an automatically generated scene is performed. Supernodes may contain sub-nodes of type supernode or scenenode. One of these sub-nodes must be declared the starting node. Edges connected to a supernode will be inherited by all sub-nodes.

At runtime, a node is called *running*, if the attached scene is currently performed, and terminated as soon as the attached scene is finished.

3.2.2 Scene transitions

In our scene flow definition, edges define the transition between states. In the context of the scene flow, they represent the transition between scenes, the processing of user interaction, or events. Like nodes they may have pre-scripted scenes attached which will be invoked when the edge is traversed. There are three types of edges (Fig. 4):

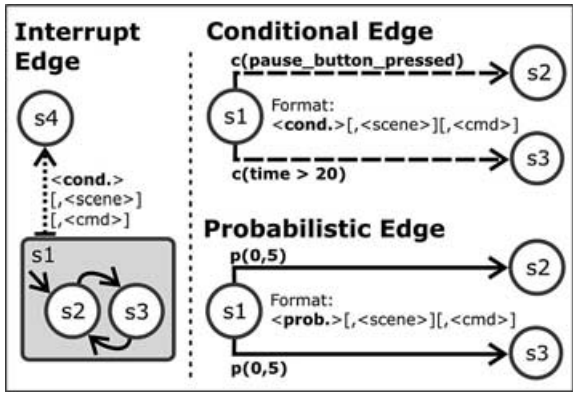

Fig. 4 Edge types

1. *Interrupt edge*: used for the handling of interruptive events, like pressing the pause button on a CD player. These edges have the task to directly interrupt a running node. Currently, this edge type supports temporal constraints like '20 seconds passed' or conditions like 'user has pressed red button'. If an edge points to its own source node, and if this node is a supernode, the author can specify to start over (i.e. restart interrupted scene) or jump to the last executed sub-node (i.e. resume interrupted scene).
2. *Conditional edge*: only when the node is terminated all conditional edges are checked in the order of author-specified priorities. This edge type supports the same constraints/conditions like the interrupt edge. If all conditions fail, the probabilistic edges are checked.
3. *Probabilistic edge*: these will be checked when the node is terminated and all conditional edges fail. In fact, probabilistic edges are ε-transitions tagged with a probability to support random branching in the scene flow.

Interrupt edges and conditional edges are mainly used for scripting the user interaction as described in the next section whereas probabilistic edges are used as a design feature for making the performance more variable.

3.3 Scene flow in CrossTalk

In this section we show how the scene flow is used to make CrossTalk run. Figure 5 shows a small excerpt from the CrossTalk scene flow. The system starts in OFF mode. This mode is modeled as a supernode with no scene attached (upper box). The sub-node *idle0*, declared starting node, is processed first. Having performed the attached scene, node *idle1* or node *idle2* are processed with probability 0.5 each. If a visitor arrives, the currently processed node is interrupted by traversing the interrupt edge *visitor_detected* to supernode ON, playing the attached scene *intro*. Then, Cyberella asks whether the user wants to provide parameters for the CarSales exhibit (see Sect. 4.3.2). This yes/no question is handled with two conditional edges: $c(yes)$ and $c(no)$. If the user does not answer within a certain amount of time (20 seconds), a third conditional edge $c(t > 20)$ is triggered, playing scene *feedback-reminder*.

During the CarSales performance the user can give feedback by pushing buttons. Multiple interrupt edges *c(feedback)* handle these button events, interrupting the performance temporarily by playing the respective scenes. Should the visitor leave, the interrupt edge *visitor_gone* immediately stops all ongoing activities in ON mode and enters the OFF supernode. Figure 6 shows the scene flow described above in XML syntax that we use as input for the SceneMaker tool.

The interactions mentioned above are part of CrossTalk's repertoire of menu-based interaction patterns:

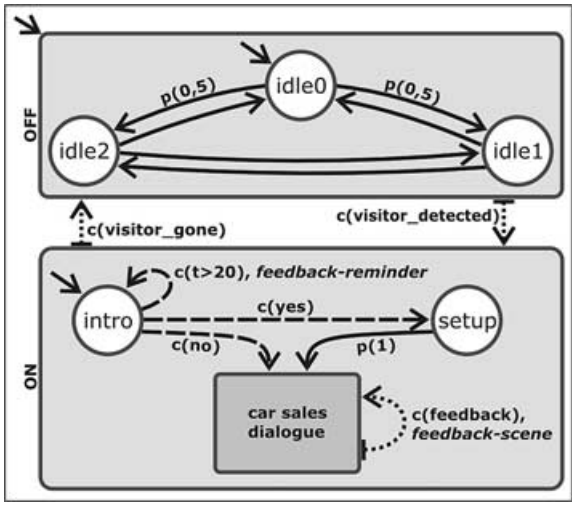

Fig. 5 Simplified scene flow in the CrossTalk scenario

```
<scene-flow start="OFF" name="simpleCrossTalk">
 <super-node name="OFF" sub-start="node0">
  <i-edge target="ON">
   <conditions>
    <event name="visitor_detected" />
   </conditions>
  </i-edge>
  <scene-node name="node0">
   <scene name="idle0" />
    <p-edge target="node1" prob="0.5" />
    <p-edge target="node2" prob="0.5" />
   </scene-node>
  ...
 </super-node>
 <super-node name="ON" sub-start="node3">
  <i-edge target="OFF">
   <conditions>
    <event name="visitor_gone" />
   </conditions>
  </i-edge>
  <scene-node name="node3">
   <scene name="intro" />
   <c-edge target="node4">
    <conditions>
     <event name="yes" />
    </conditions>
   </C-edge>
   <c-edge target="demo">
    <conditions>
     <event name="no" />
    </conditions>
   </c-edge>
   <c-edge target="node3">
    <scene name="feedback-reminder" />
    <conditions>
     <time-greater-than value="20" />
    </conditions>
   </c-edge>
  </scene-node>
  <scene-node name="node4">
   <scene name="setup" />
   <p-edge target="demo" prob="1" />
  </scene-node>
  <super-node name="demo">
   <automatic-scene name="car-sales-dialogue" />
   <i-edge target="demo">
    <scene name="feedback-scene" />
    <conditions>
     <event name="feedback" />
    </conditions>
   </i-edge>
  </super-node>
 </super-node>
</scene-flow>
```

Fig. 6 Example scene flow specification

- Yes/No question and Yes question[3]
- User feedback choice (applause, boo, question)
- Parameter choices (role, personality and issues).

Each of these interactions is initiated by an edge in the scene flow where an attached system command causes the respective menu to appear on the user touch screen. Handling the user input can be modeled using various events and handling mechanisms:

1. *Request and wait*: an agent asks the user a question and the system waits until the user answers.
2. *Time-out events*: waiting for user input, the system regains the initiative after a defined period of time.
3. *Interrupts*: user feedback during the CarSales performance causes the system to seamlessly integrate a generic scene. Or, if the visitor leaves, the system interrupts all current activities and switches from ON to OFF.
4. *Concurrent event handling*: user feedback during the CarSales performance does not interrupt the performance but influences the agents' behavior long-term by modifying the context.

These events and handling mechanisms are implemented using the conditional and interrupt edge types. Request and wait is realized with conditional edges, one for each answer. Time-out events are produced using interrupt edges that act as observer demons and access system time. Both interrupts and concurrent event handling are realized in a similar manner. The latter does not interrupt the current performance but modifies the context instead to achieve long-term changes in character behavior.

3.4 Context in CrossTalk

By tailoring the performers' reactions to the user's previous behavior we can convey the impression that our agents understand the user while at the same time making the user feel like having an impact on the presentation in a way that is subtle and long-term.

Context-dependent branching facilities enable the author to select scenes based on variables like character mood, situation (daytime, location etc.) or discourse history (played scenes, utterances, user input). The author can also integrate pieces of context, e.g. the current time, into the spoken utterances of the characters using access functions in the content script.

Besides collecting information in a discourse history we analyze the data using measures like frequency and density. These measures are used to infer user stereotypes that can be exploited by the author.

[3] A Yes question occurs when the user has been presented with a question or choice but did not answer within a pre-defined time span (time-out). Then, Cyberella will ask "Are you still there?" and a "Yes" button appears. This we call a Yes question.

3.4.1 Context data

The authoring toolkit provides access functions to the context memory, allowing us to retrieve the following data:

- elapsed time (duration)
- feedbacks (positive, negative, question)
- interactions (type)

Feedbacks can be differentiated by addressee (Tina or Ritchie). For both feedbacks and interactions it is possible to request the number of occurrences and the number of time-outs. With this data, we are able to infer a user stereotype at runtime that can be used for conditional authoring. The range of stereotypes includes: critical user (many negative feedbacks), active user (many interactions), passive user (few interactions), liking/disliking for agent X (many positive/negative feedbacks for agent X), lazy user (frequent interaction time-out), tenacious user (long sessions).

Besides the memory data and user stereotypes, we also suggest a range of measures that can be computed from the original data:

- feedback density (total/positive/negative feedback over time)
- average response time (for each interaction type)

The access function values can be inserted into pre-scripted scenes by means of placeholders (so agents can talk about the number of interactions etc.). They can also be used as conditions to trigger pre-scripted scenes as described in the following section.

3.4.2 Context for scene selection

For the human author who tries to create generic dialogue pieces that nevertheless sound spontaneous and coherent, context knowledge allows him/her to react to unusual patterns. For example, a positive response after a series of negative responses could trigger a scene with a side remark like "It was about time you said something nice!". Here are some other examples (X is either Tina or Ritchie, C is Cyberella) where the user gives feedback during a performance by Tina and Ritchie:

- Liking for X causes X to say: "Oh, a real fan!"
- Disliking for X causes X to make nervous gestures or not to react at all
- Liking for X and negative feedback causes X to say: "Oops, just a little accident"
- Lazy user causes C to say: "You don't like talking, do you?"
- High feedback density causes C to say: "Why don't you just let it roll for a while" or she ignores feedback.
- The user stereotype controls the amount of menu options. If the user is active he is provided with more frequent choices. If not, not.

The concept of Liking triggers whole pre-scripted scenes (off-duty mode) where the agents argue about the worth

of 'popularity measured by empirical evidence' using data from context memory.

3.4.3 Context for evaluation

At several occasions hundreds of people have used the system, and interacted with it, often following certain patterns. This data can be fed back into the system to expand its possibilities and allow first, tentative evaluations. Therefore, we log discourse history data for further offline analysis.

Evaluation is a burning issue in the animated agents community. Interactive systems have the great advantage that the interaction implicitly contains data about the system's effect on users. Measuring the time spent with the system alone allows tentative conclusions to be drawn about the attractiveness of the system. Using CrossTalk as a platform for arbitrary scenarios that are automatically generated, this kind of evaluation would allow to compare different character designs, different versions of a scenario, or whole dialogue generation engines – whether one is more interesting than the other.

4 The CarSales exhibit

4.1 Introduction

CrossTalk provides a virtual stage for the promotion of exhibits. An exhibit is a program that implements an embodied agents scenario, e.g. simulated sales dialogues, interactive tutoring, storytelling systems or interactive games. The promotion of such standalone applications, usually done by human personnel, can now be taken over by Cyberella, a presentation agent in a fully automated exhibition environment. These exhibits can be exchanged, they can be 'plugged' into the system, if they provide an interface to CrossTalk's PresentationProcessor.

CrossTalk's current exhibit is *CarSales*, a program that automatically generates car sales dialogues according to the personal interests of a user. Two embodied agents, Tina and Ritchie, are salesperson and customer talking about a car. The dialogue is not just a simple enumeration of facts. Instead, information about the car is presented in tactical question-answer games where the customer asks questions addressing issues of user interest. The salesperson either provides the requested information or uses tactical answers like evasions depending on personality. The idea is to inform and entertain the visitor similar to infomercials common on TV. The dialectic discourse becomes entertaining because the user can configure the agents' personalities. This kind of entertaining information presentation guided by user interests is a step towards customized e-commercials.

In the remainder of this section we describe CarSales on a technical level: its domain model, the dialogue strategies, and the types of parameters that can be configured by the user to influence the course and style of the ensuing conversation.

4.2 Domain model

Automatic scene generation requires a representation of the subject-matter domain. Part of the domain model is a database containing information about cars. All questions and answers that relate to the product are grounded in the contents of this database. It determines to a large extent what the virtual actors can talk about. Another part of the domain model relates the car attributes to the value dimensions of potential customers.

We use 13 attributes to characterize a car: price, horsepower, maximum speed, consumption, airbags, anti-lock brakes, broad tires, spaciousness of luggage compartment, spaciousness of interior, power windows, leather seats, portion of recyclable materials, and catalytic converter.

The value dimensions for the CarSales domain were adopted from a study of the German car market that suggests that price, operational costs, safety, comfort, prestige, sportiness, family and environmental friendliness are the most relevant [16]. The domain model characterizes a car attribute in two ways. First, how relevant it is for a certain value dimension: low, medium or high. For example, the 'consumption' attribute's relevance for the value dimension 'operational costs' is high. Secondly, an evaluation function determines the valence of an attribute's value in a value dimension: positive or negative. For example, a consumption of '10 liters per 100 kilometers' is negative in the 'operational costs' dimension. Figure 7 shows an excerpt of our domain model. The table entries show the attributes' relevance and samples of the evaluation function. If the relevance of an attribute for a value dimension is low the respective entry is left empty.

4.3 Dialogue generation

This section gives a concise description of our plan-based approach to automatic dialogue generation [5]. We first identify typical dialogue moves in a domain. A dialogue move represents a communicative act such as requesting some information (request), answering a question (confirm, inform), or giving feedback (agree). We then define dialogue strategies that combine these dialogue moves based on the structure of typical dialogues in the domain. A sales dialogue, for example, starts with a greeting phase, followed by the customer's request for information about a specific product. Subsequently, a question-answer game between the customer and the salesperson develops where various aspects of the product are discussed. Finally, the customer communicates a purchase decision and, in a closing phase, the communication is finished.

value dimension / attribute (value)	operational costs	prestige	environmental friendliness	safety
consumption (10 litres / 100km)	high, negative		high, negative	
maximum speed (210 km/h)		high positive		high negative
broad tires (yes)	low negative	medium positive		medium positive
airbags (yes)		low positive		high positive

Fig. 7 Excerpt from the domain model

4.3.1 Dialogue moves and strategies

Dialogue strategies are encoded as plan operators that can be processed by the PresentationProcessor (Sect. 2.3). Each plan operator has a goal and a precondition defining the dialogue context in which it can be applied and a body containing a procedural specification where other dialogue strategies can be triggered. When processing a plan a tree structure, the *plan tree*, is incrementally built where nodes represent dialogue strategies.

Figure 8 is an example of a plan operator for the dialogue strategy 'QuestionAnswer: Boolean'. The customer requests information about a boolean attribute by saying, for instance, 'Does it have airbags?'. Depending on the attribute value, the salesperson will confirm or disconfirm this. Finally, a new dialogue strategy is triggered in which both actors discuss this new piece of information.

A single dialogue contribution, a dialogue *turn*, is modeled by the DialogueMove plan, independent of concrete surface realizations. The DialogueMove plan selects a multimodal utterance from the database according to context (e.g. focused attribute or value dimension) and user-selected parameters (e.g. the agent's politeness). The utterances contain text and gestures, including listening actions of interlocutors. In CarSales, a large number of utterances were written by a human author using the same authoring syntax and gesture repertoire as for the pre-scripted CrossTalk scenes (Sect. 3.1).

```
Plan {
NAME : "QuestionAnswer:Boolean"
GOAL : PERFORM QuestionAnswer
                $product $attribute;
PRECONDITION :
  FACT type $attribute "Boolean";
  FACT role "customer" $actor1;
  FACT role "salesperson" $actor2;
BODY :
  PERFORM DialogueMove $actor1 "requestIf"
                       $attribute;
  ASSIGN $value (getValue $product $attribute);
  OR {
    TEST (== $value "true");
    PERFORM DialogueMove $actor2 "confirm";
  }{
    PERFORM DialogueMove $actor2 "disconfirm";
  };
  PERFORM DiscussValue $product $attribute;
}
```

Fig. 8 Dialogue strategy for the car sales domain

4.3.2 User-selected parameters

In CarSales, the visitor can assign roles (salesperson and customer) and personalities (polite vs. impolite, good-humored vs. bad-tempered) to the virtual actors. These personalities are enacted by Tina and Ritchie, i.e. they belong to their roles as salesperson or customer (ON mode). They have no influence on their *meta-role*, i.e. Tina and Ritchie as actors (OFF mode). The visitor can also select the value dimensions that interest him/her.

These parameters influence the course and style of the ensuing conversation. The course of the dialogue is influenced by constraining the selection of dialogue strategies. For example, depending on their mood the two actors use different degrees of criticism (customer) and enthusiasm (salesperson)when discussing the car's properties. The style of the dialogue is determined by different surface realizations for the same dialogue move as shown in Fig. 9. This leads to clearly noticeable variations in the performance.

If multiple utterances for the same parameter value exist, one of them is randomly selected, avoiding those most recently used.

4.3.3 Benefits of the approach

Using a plan-based approach to automatic dialogue generation has several benefits. The product database can easily be extended to increase the actors' 'vocabulary'. The only additional effort is modeling the relevance and evaluation function for each new attribute on some value dimensions. Another benefit is that a small number of dialogue strategies, we use less than 20 in CarSales, suffice

dialogue move	parameter value	multimodal utterance
disconfirm	good-humoured	1. [V_Shake] I'm afraid not. 2. Not really. [V_Shake] 3. No.
	bad-tempered	1. [PS_Hips] No! 2. [PS_Fold] Not at all. 3. Certainly not.

Fig. 9 Multimodal utterances for dialogue move 'disconfirm'

to generate a broad variation of dialogues. The specification of dialogue strategies and moves on the one hand, and the multimodal utterances on the other hand, means separating the language-independent structure of the dialogue from its language-dependent content. Therefore, making CarSales multi-lingual was done by simply translating the multimodal utterances.

The most powerful argument, however, is the fact that we can combine pre-scripted and generated scenes at runtime to establish links between the role and the meta-role of an actor (Sect. 2.2.1), e.g. by inserting unexpected out-of-character *intermezzi* in the generated dialogue or by simulating a rehearsal (i.e. inserting a piece of a generated dialogue, within pre-scripted scenes [17]).

5 Related work

In CrossTalk, embodied agents are presented as virtual actors giving interactive performances. This can be considered a playful illustration of the 'computers as theatre' paradigm introduced by Brenda Laurel [18]. During the last few years, an increasing number of attempts have been made to develop systems for *interactive drama* within which the user experiences a story from a first person perspective [19]. There are three problems in interactive drama: controlling narrative structure, integrating user and autonomous agents, providing scripting languages.

Interactive drama systems usually have synthetic agents that the user interacts with. The fundamental problem there is that user and agents can make decisions that endanger the narrative structure intended by an author. The solution is usually to define 'freeplay' areas, where user and agents can freely interact, represented by nodes in a story graph [20]. Transitions in the graph can then be used to control the development of the plot.

This is either realized by local evaluation functions for transition selection or by a so-called plot or drama manager with global control [21–23]. The Oz drama manager [21] selects actions based on an evaluation of a sequence of plot points in terms of dramatic quality. In [20], transitions are triggered by events or user/agent actions. In a system for *interactive pedagogical drama* [24] an evaluation function models pedagogical progress. In the system by [25], where the story graph is called *StoryNet*, an evaluation function checks a list of conditions, usually tasks that must be completed. In traversing transitions, pre-scripted scenes are played to set up the next node's situation.

IMPROV is one of the first authoring systems for interactive characters [26]. It allows authors to write action scripts and decision rules. Both determine the overall behavior of an actor based on local decisions. There is no explicit representation of narrative structure as in CrossTalk. In SCREAM (SCRipting Emotion-based Agent Minds) [27] characters are scripted at the level of local action-reaction pairs. Story structure depends on these rules together with the author's definition of an agent's initial goals, beliefs, and attitudes. IMPROV and most other systems focus on the freeplay areas in their research, i.e. the nodes in the story graph. Within these a player interacts with characters in complex ways to influence how the story unfolds. Transitions between story nodes are rarely modeled within a formal representation of the story graph. Instead, story graphs are usually hard-coded by programmers and cannot be authored or modified easily. By using cascaded finite state machines (FSMs) the scene flow can be explicitly modeled by a human author which is simple to maintain, at the same time allowing complex transitions conditions. Cascaded FSMs are similar to Badler's parallel transition networks (PatNets) [28]. PatNets are used for the autonomous control of gaze and hand movement in animated agents. Although this is a lower level of control compared to scene selection, PatNets could in theory also be used for this purpose. In fact, cascaded FSM are inferior to PatNets in terms of expressivity because they do not allow the parallel execution of two actions (scenes). However, cascaded FSM are especially designed to allow simple scene flow management by providing hierarchical structures for a basically sequential process. Adding parallel structures would interfere with the simplicity of maintenance.

CrossTalk provides authors with tools that make editing the scene flow easy and transparent. The scene flow can be seen as an extension of story graphs. On top of this, CrossTalk offers a screenplay-like authoring language to specify the virtual characters verbal and nonverbal behavior as opposed to other systems that rely on formal mark-up languages like MPML [29] and VHML[4], both XML-based. These languages were

[4] http://www.vhml.org

designed for augmenting web pages with animated presentation agents similar to DFKI's WebPersona [30]. They do not have a context memory and only rudimentary support to define the scene flow, e.g. by using hyperlinks to jump to other parts in the script.

6 Summary

In this article we have described the CrossTalk framework for staging virtual character exhibitions in public spaces. Part of the overall framework are the SceneMaker authoring tool and the CarSales exhibit. CrossTalk extends the commonplace human-screen interaction to an interaction triangle. The physical presence of the agents is established through the separation of screens and intensified by inter-agent conversations across screens. CrossTalk utilizes a combination of both automatically generated and pre-scripted scenes, and deploys a context memory to adapt to the user and the environment. A number of principles have been introduced to enhance the lifelikeness of the virtual characters, including multiple layers of illusion (role and meta-role), letting characters be aware of situational context, alluding to cultural references, and most of all, creating the illusion that the whole installation is under full control of virtual characters. CrossTalk is generic with respect to dialogue content, narrative structure, and the promoted exhibit through the SceneMaker authoring tool.

SceneMaker separates narrative structure and content. Narrative structure is explicitly represented using cascaded finite state machines with typed edges for conditional branching. Non-experts can write multimodal content in a screenplay-like language drawing on a large repertoire of empirically motivated conversational gestures. Authors can use context memory access functions to make scenes user-adaptive. The SceneMaker tool also allows to seamlessly integrate automatically generated scenes as demonstrated by the CarSales exhibit.

The CarSales exhibit illustrates how plan-based dialogue generation can be used to create personalized infomercials. The demo can be configured by the user and is performed by two virtual actors. The performance can be influenced by user feedback in the form of applause, booing and asking for help. The generated CarSales performance does not relate to the agent's existence as 'actors'. Since the agents' roles in CarSales are independent of their meta-roles in CrossTalk as actors, exchanging the exhibit is easily possible.

CrossTalk's approach allows to plug arbitrary character-based exhibits into the system, making it a generic framework for staging embodied agents technology.

Acknowledgements The work presented in this paper is a joint effort with contributions from our colleagues Stephan Baldes, Markus Schmitt, and Thomas Schleiff. We would also like to thank our graphics designer Peter Rist for providing us with the virtual actors Cyberella, Tina, and Ritchie. CrossTalk has been built upon contributions from the MIAU project funded by the German Ministry for Education and Research and from the EU-funded IST projects NECA, SAFIRA, and MAGICSTER.

References

1. Rickel J, Johnson WL (1999) Animated agents for procedural training in virtual reality: perception, cognition, and motor control. Appl Artif Intell 13:343–382
2. Noma T, Zhao L, Badler N (2000) Design of a virtual human presenter. IEEE J Comput Graph Applic 20(4):79–85
3. Cassell J, Bickmore T, Billinghurst M, Campbell L, Chang K, Vilhjálmsson H, Yan H (1999) Embodiment in conversational interfaces: Rea. Proceedings CHI 99 Conference, pp 520–527
4. André E, Rist T (2000) Presenting through performing: on the use of multiple animated characters in knowledge-based presentation systems. Proceedings IUI'00, ACM Press, pp 1–8
5. André E, Rist T, van Mulken S, Klesen M, Baldes S (2000) The automated design of believable dialogues for animated presentation teams. In: Cassell J, Sullivan J, Prevost S, Churchill E (eds) Embodied Conversational Agents. MIT Press, Cambridge, pp 220–255
6. Aylett R (1999) Narrative in virtual environments – towards emergent narrative. Proceedings AAAI Symposium on Narrative Intelligence
7. Thomas F, Johnston O (1981) The illusion of life: Disney animation. Hyperion Press, New York
8. Blumberg B, Tomlinson B, Downie M (2001) Multiple conceptions of character-based interactive installations. Proceedings 19th Computer Graphics International Conference (CGI-01), IEEE Computer Society, pp 5–14
9. Huber M (2001) JAM: A BDI-theoretic mobile agent architecture. Proceedings Third Conference on Autonomous Agents, ACM Press, pp 236–243
10. Gebhard P, Kipp M, Klesen M, Rist T (2003) Authoring scenes for adaptive, interactive performances. Proceedings Second International Joint Conference on Autonomous Agents and Multiagent Systems
11. Kipp M (2001) From human gesture to synthetic action. Proceedings Workshop on Multimodal Communication and Context in Embodied Agents, held in conjunction with the Fifth International Conference on Autonomous Agents, pp 9–14
12. Kipp M (2001) Anvil – a generic annotation tool for multimodal dialogue. Proceedings 7th European Conference on Speech Communication and Technology (Eurospeech), pp 1367–1370
13. Kipp M (2001) Analyzing individual nonverbal behavior for synthetic character animation. In: Cavé C, Guaitella I, Santi S (eds) Oralité et Gestualité – Actes du colloque ORAGE, pp 240–244
14. Scheflen AE (1964) The significance of posture in communication systems. Psychiatry 26:316–331
15. Silva A, Vala M, Paiva A (2001) Papous: the virtual storyteller. In: . de Antonio A, Aylett R, Ballin D (eds) Intelligent Virtual AgentsVolume, Lecture Notes in AI 2190. Springer Verlag. 2001; 171–180
16. Spiegel-Dokumentation: Auto, Verkehr und Umwelt. Spiegel-Verlag, Hamburg
17. Baldes S, Gebhard P, Kipp M, Klesen M, Rist P, Rist T, Schmitt M (2002) The interactive CrossTalk installation: meta-theater with animated presentation agents. Proceedings PRICAI'02 Workshop on Lifelike Animated Agents
18. Laurel B (1993) Computers as theatre. Addison-Wesley, Reading MA
19. Murray JH (2000) Hamlet on the holodeck: The future of narrative in cyberspace. MIT Press, Cambridge, MA
20. Sgouros N (1999) Dynamic generation, management and resolution of interactive plots. Artif Intell 107(1):29–62

21. Weyhrauch P (1997) Guiding interactive drama. PhD.thesis, Carnegie Mellon University
22. Mateas M, Stern A (2000) Towards integrating plot and character for interactive drama. Working notes of the Social Intelligent Agents: The Human in the Loop Symposium. AAAI Press
23. Paiva A, Machado I, Prada R (2001) Heroes, villians, magicians, ...: dramatis personae in a virtual story creation environment. Proceedings 6th International Conference on Intelligent User Interfaces, pp 129–136
24. Marsella SC, Johnson WL, LaBore C (2000) Interactive pedagogical drama. Proceedings Fourth International Conference on Autonomous Agents. ACM Press, pp 301–308
25. Swartout W, Hill R, Gratch J, Johnson WL, Kyriakis C, LaBore C, Lindheim R, Marsella S, Miraglia D, Moore B, Morie J, Rickel J, Thiébaux M, Tuch L, Whitney R, Douglas J (2001) Toward the holodeck: integrating graphics, sound, character and story. Proceedings Fifth International Conference on Autonomous Agents. ACM Press, pp 409–416
26. Perlin K, Goldberg A (1996) Improv: a system for scripting interactive actors in virtual worlds. Comput Graph 30:205–216
27. Prendinger H, Ishizuka M (2002) SCREAM: Scripting emotion-based agent minds. Proceedings AAMAS'02. ACM Press, pp 350–351
28. Badler N, Webber B, Becket W, Geib C, Moore M, Pelachaud C, Reich B, Stone M (1995) Planning and parallel transition networks: animation's new frontiers. In: Shin SY, Kunii TL (eds) Computer Graphics and Applications. World Scientific, New York, pp 101–117
29. Ishizuka M, Tsutsui T, Saeyor S, Dohi H, Zong Y, Prendinger H (2000) MPML: A multimodal presentation markup language with character control functions. Proceedings Workshop on Achieving Human-like Behavior in Interactive Animated Agents, held in conjunction with the Fourth International Conference on Autonomous Agents, pp 50–54
30. André E, Rist T, Müller M (1997) WebPersona: a life-like presentation agent for the world wide web. Proceedings IJCAI'97 Workshop on Animated Interface Agents: Making them Intelligent