Integrating VR-Authoring and Context Sensing: Towards the Creation of Context-Aware Stories

Dennis Erdmann, Klaus Dorfmueller-Ulhaas, and Elisabeth André

University of Augsburg, Multimedia Concepts and Applications, Eichleitnerstr. 30, 86159 Augsburg, Germany
{erdmann, dorfmueller-ulhaas, andre}@informatik.uni-augsburg.de

Abstract. Recent progress in the area of sensor technology has enabled the development of context-aware systems that are able to dynamically adapt their behaviours to the current situation and the individual user. In this paper, we present a framework for a new generation of context-sensitive stories that dynamically adapt to changing environmental conditions and user states. The framework combines approaches to interactive storytelling with work on context toolkits that foster the rapid prototyping of context-aware applications.

1 Introduction

Recently, a growing number of story telling environments are emerging that are embedded in the user's physical world and thus require more sophisticated mechanisms for acquiring context data. Unfortunately, the acquisition of context information is still cumbersome and requires a significant amount of technical expertise. Hardly any support is provided for the human author to specify context constraints in a declarative manner while abstracting from the details of the underlying context sensing mechanism. As a consequence, existing story telling systems that are embedded in the user's physical world usually rely on a limited set of dedicated context devices, such as a camera for tracking the user's motions.

In this paper, we present a framework that allows the author to specify context-sensitive stories at various levels of abstraction while providing specific support for the integration of context data that are not explicitly communicated by the user during the interaction. In this way, the author is able to specify at a high level of abstraction how a story changes considering user states and environmental information. For instance, a story may develop differently depending on where the user moves in the physical space, which objects he is grasping, the noise level of the environment, the time of the day etc.

In order to accomplish this goal, we need to provide the author with tools for the easy creation of audio-visual story elements. For instance, we may specify a story element that includes playing a complex animation sequence for presenting information to a user. In addition, tools have to be provided to specify how the flow of a story changes according to explicit user input and implicit contextual constraints. For instance, a different story element may be triggered depending on where the user is looking. Finally, we need to account for an easy way to integrate heterogeneous context devices that gather information on the environmental conditions as well as user states.

2 Related Work

Our work combines approaches from the area of interactive storytelling with work on frameworks for the rapid prototyping of context-aware systems. In the following, we will first review work done in the two areas and then report on first attempts to enhance interactive story telling by limited forms of context awareness.

2.1 Authoring Paradigms for Interactive Storytelling

Hardman and Bulterman [Bulterman2005] distinguish between four paradigms of authoring: *structure-based*, *timeline-based*, *graph-based*, and *script-based* authoring.

Structure-based authoring is based on an abstract representation of the story. Media elements are structured in the manner they get activated. Commonly a presentation or a story is subdivided into substories and usually administrated using a document hierarchy or a document tree. An example of a structure-based authoring approach includes the narrative prose generator by Callaway and Lester [Callaway2002].

The *timeline-based* paradigm is based on an explicit ordering of media assets along a time axis. An example of a timeline-based approach includes the work by Pinhanez and Bobick [Pinhanez1998]. The basic idea is to associate actions and states of the user as well as of the system with temporal intervals. Programming interactive environments is then accomplished by establishing temporal constraints between these intervals.

The *graph-based* approach relies on directed graphs to describe the flow of an interactive multimedia presentation. An example for a graph-based approach includes the work by Gebhard and colleagues [Gebhard2003] who presented a toolkit called SceneMaker for authoring interactive performances with embodied conversational agents. First, the scene flow is modelled by means of a cascaded finite state machine in which states refer to scenes with characters. Second, the content is provided which has been either pre-scripted by a author or is generated automatically on-the-fly using a plan-based approach.

Finally, *script-based* approaches allow the user to specify multimedia presentations in a procedural manner. A prominent example includes the scripting language Lingo included in Macromedia Director or the authoring language LUA which has been used in a number of commercial game applications (such as World of Warcraft). Furthermore, a number of dedicated mark-up language have been developed to direct the verbal and non-verbal behaviour of animated agents. Usually, the approaches focus on the specification of the behaviours of single agents or a group of agents. There is hardly any approach that explicitly deals with reactive behaviours. An exception is the work by Nischt and colleagues [Nischt2006] who present a new mark-up language that relies on a reactive model which is able to handle interaction-rich scenarios with highly realistic 3D characters. Based on the reactive model, a scenario has been implemented that relies on eye tracking technology to monitor the user's focus of attention and adapt presentations accordingly. While their approach facilitates the creation of interactive stories, no mechanism is provided to easily integrate context information, however.

2.2 Context Aware Systems

The approaches discussed above allow the author to specify interactive stories at a high level of abstraction. However, they do not provide the author with the possibility to easily access context information. Context factors may refer to user states, for example his or her emotions, but also to environmental factors, such as temperature or luminosity. Context aware systems are able to dynamically adapt their behaviour to changing circumstances without explicit user interaction by continuously sensing and interpreting data from the user's environmental context.

Work done in the area of context-sensitive systems aims at the development of tools to foster rapid prototyping of context-aware applications. A prominent example includes the context toolkit by Dey [Salber1999] which is based on the widget principle. On the analogy of GUI widgets, context widgets provide a declarative interface for sensor hardware by abstracting from the details of context sensing. The context widget principle enables the easy exchange of widgets that provide the same kind of context information. For instance, we may exchange an RFID widget with a camera widget to get information on the position of objects in the physical environment. Furthermore, there is support for interpreting and aggregating context data.

2.3 Storytelling in Mixed Reality

There are also first approaches to create stories within a physical space which are of high interest to our own work because of the need to integrate context information. An example includes the approach by Romero and colleagues [Romero2004] who presented a hypermedia model that includes references to media elements, objects and relations between locations in physical and virtual worlds. Cavazza and colleagues [Charles2004] developed a mixed reality story telling environment which puts the user both in the role of an actor and a spectator by inserting the users video image in a virtual world that is populated by synthetic agents. The user interacts via natural language and gestures that are mirrored in the user's video image. The DART system of MacIntyre et al. [MacIntyre2004] has been developed to enable designers to work with Augmented Reality (AR). Many additional behaviours and actions are incorporated into Macromedia Director in order to access sensor data for Augmented and Virtual Reality, and to easily set up a new AR application. As a result, designers can work in the way they are accustomed to by using the timeline paradigm of Macromedia Director and predefined AR functionality provided by the DART framework. Unfortunately, the DART framework is limited to the performance and to interfaces of Macromedia's Director. Additionally, context is only provided by tracking sensors whereas a general concept for processing context information is not given.

2.4 Concluding Remarks

It is important to note that many approaches do not just rely on one authoring paradigm. For example, the graph-based approach by [Gebhard2003] may be combined with a planning approach that includes elements of structure-based authoring. For our purpose, the combination of a graph-based approach with a scripting approach seems most appropriate since it allows for a flexible way of integrating context information.

No authoring support is provided that enables both the specification of the content and flow of an interactive story as well as the easy integration of heterogeneous context devices. Our work aims at closing this gap by combining work done in the area of interactive story telling and work done in the area of context toolkits.

3 Overview of the ACOSAS System

The development of ACOSAS (**A** **CO**ntext **S**ensing **A**uthoring **S**ystem) was driven by our plans to focus on interactive story telling in highly dynamic and unpredictable articifical and natural environments. ACOSAS was designed to access context information from the user's physical environment and to fuse real with virtual context. The bulk of ACOSAS functionality comes from a collection of context data provided by different sensors in the environment which can be accessed at any time. Additionally, since authors often do not have working sensors in physical environments or alternatively want to simulate future sensor technology during the prototype stages, ACOSAS enables "Wizard of Oz" simulation similar to Dow et al. [Dow2005]. In order to provide informal content, ACOSAS enables us to import 3D content from commercial design tools, such as 3D Studio Max and Blender, and provides methods to access this data. Most of ACOSAS is implemented in an interpreted scripting language (LUA), and can be modified by authors and storytellers to suit their needs, e.g., to support a new sensor device. Due to performance improvements needed, we decided to develop ACOSAS instead of using Macromedia Director, the defacto standard for multimedia content creation. Additionally, we are not limited to the restrictive programming interface of Director. As a consequence, we gain higher flexibility when integrating novel research ideas using a low level programming language such as C++. Furthermore, we can easily support a manifold of sensors and new output devices with stereoscopic rendering and asymmetric viewing frustums. Advantages in regard to latency of transmitted sensor and tracking data can be measured. With respect to augmented reality applications, the ACOSAS system should allow authors to specify complex relationships between the physical and the real world.

3.1 System Design

The ACOSAS framework as depicted in Figure 1 contains a story engine executing an authoring file, a library with predefined actions, and a server collecting the data of real and simulated sensors. Client modules implemented in JAVA and

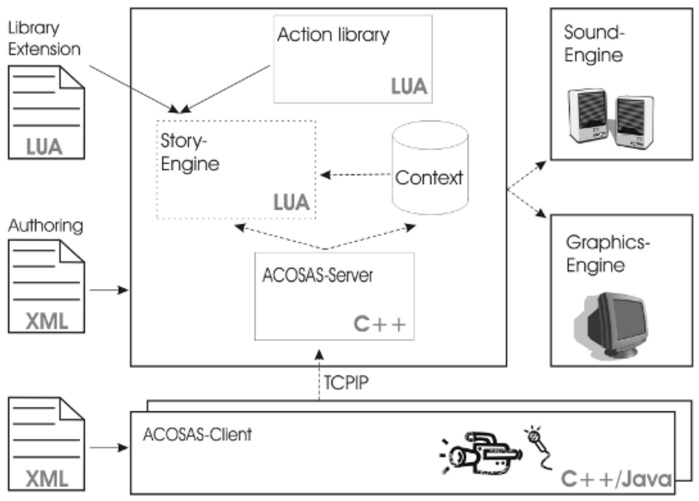


Fig. 1. Overview of the ACOSAS system. The core of ACOSAS is given by the interpretation of a graph-based authoring *XML*-file. The story engine converts this story content into a queue of invoked actions. Actions as defined by the action library are used to retrieve context data or to apply dynamic changes of the scene. In case, provided actions do not satisfy the needs of the author, additional scripted LUA functions can extend the functionality of ACOSAS. Last but not least, supporting a new sensor device requires the implementation of a client and the specification of client and event properties.

C++ are available and can be adjusted to easily support new sensor devices. Furthermore, the ACOSAS framework includes a sound and a graphics engine for rendering the audio-visual scene while the story engine is responsible for the synchronisation of the graphical animations and sound replay. The ACOSAS system additionally includes a database to store context information deliverable on request during runtime. Alternatively, for immediate and continuous requests the ACOSAS server is able to invoke a callback function defined by the author.

3.2 Context Modules

We make use of a client-server architecture in order to permit multiple clients to access sensor data concurrently that are gathered by a context server. This approach fosters not only the re-use of sensors, but also relieves the basic ACOSAS system from time-consuming operations.

ACOSAS-Client: ACOSAS clients collect contextual information recognized by specific sensor hardware and forward it to the context server using the local area network (TCP-protocol). To enable access to heterogeneous context devices, ACOSAS includes three basic modules to acquire context information: (1) a Java module to connect Java-based or Java-prepared hardware to the system, (2) a

C++ module with analogous functions and (3) a standalone client mainly used for non-changeable console programs, debugging, and Wizard of Oz simulation.

ACOSAS-Server: The counterpart of the clients, the server, waits for context registration on the pre-specified TCP-port. When noticing that a new context client tries to connect to the system, the server checks all parameters and verifies the data type. In case no error has occurred, the client will receive an acknowledgement. Context data transmitted by several context clients are processed by the server following the context widget principle introduced by Dey [Salber1999]. After receiving data from the context clients, the context server updates the information in the corresponding context widget. Information stored in context widgets may be further interpreted and aggregated. For instance, temperature and luminosity data may be transformed into higher level weather descriptions, such as "sunny hot day".

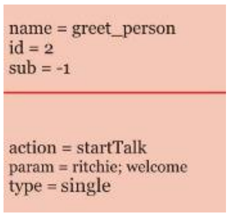
4 Authoring Context-Sensitive Stories

Following Gebhard and colleagues [Gebhard2003], we model the flow of a reactive story by means of cascaded finite state machines. The nodes of the finite state machines correspond to story elements while the edges represent transitions between story elements. We distinguish between the following kinds of transition:

– *Spontaneous transitions* do not underlie any constraints and may always be performed. They have the lowest priority of all transitions.
– *Conditional transitions* are performed only if the corresponding conditions are satisfied. In case the condition is not satisfied or information to verify the condition is missing, no transition is performed.
– *Context-sensitive transitions* are performed only if the corresponding context conditions are satisfied. In case the required sensory information is missing, no transition is made. Context-sensitive transitions have the highest priority.

In order to illustrate our concepts, we first present a single story node with a conditional transition, and then give an example of a small story including all three transition types.

Figure 2 shows a graphical representation of a single story node specified by the author in XML notation. To categorize the story node, the declaration includes a name. The name of a story node and its parameters define the type of the story node. A unique identifier id enables references to the story node. Using the sublevel identifier sub, we may group story nodes and thus define modules for a part of a story. The action tag is used to denote a function either specified in the action library or given by an additional LUA file, written by the author of the story (compare file Library Extension shown in Fig. 1). Parameters of this function are specified by the param tag while parameters are named references to geometry, sound or animation data referred in the header of the authoring XML file. A type tag is introduced to specify the type of transition tracing. In case of single, the story node is visited only once, in contrast to the statement multiple.



```
<storynode name="greet_person"
    id="2" sub="-1"
    action="startTalk"
    type="single">
    <param>ritchie</param>
    <param>welcome</param>
    <transition destination="3"
                type="condition"
                name="none"
                condition="end"
                value="none"/>
</storynode>
```

Fig. 2. A story node in the ACOSAS-System contains properties for naming and grouping story nodes, a story action to invoke, and transitions defined in child tags of the XML-node. Left: Graphical representation of a story node. Right: The corresponding XML file.

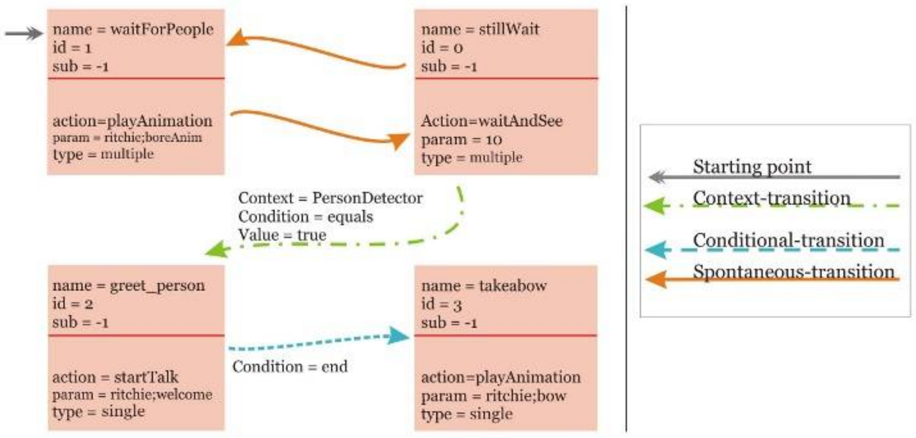


Fig. 3. Story scheme - example of a valid story flow using the three available transitions. (Orange/lined arrow = spontaneous transitions; blue/dotted arrow = conditional transitions; green/dotted-lined arrow = context transitions).

Figure 3 shows the representation of an XML defined story which illustrates the categories of valid transitions. This story begins with a character (compare parameter reference ritchie) who appears to be bored because nobody is close to the screen. After playing the corresponding animation file, the virtual character performs a "wait and see"-action for ten seconds. After looking around, it falls back to the "being bored"-state if still no person to interact with it is close to the setup. In case a sensor (e.g. an optical sensor) detects a person in front of the screen, the specified condition is activated and the character welcomes the detected person. In case of an end flag which is emitted if a story node and each of its sub nodes is processed, the character takes a bow.

5 Flexibility of the ACOSAS System

In the following, we briefly sketch how an experienced author may specify a story by enhancing the basic repertoire of story actions and transition predicates. To facilitate the creation of a story, ACOSAS provides the author with a basic repertoire of VR actions, which can be found in nearly every Authoring application. It includes methods for manipulating the virtual camera position and rotation, methods for modifying objects included in the VR, concerning position, rotation and animation, and methods for modifying global scene properties (e.g. light position). To create a new story, the author has the possibility to define a new LUA scripted method using the library extension file (cf. Fig. 1).

For illustration, let us have a look at a simple LUA script. Let us assume that we have access to a sensor which detects the environmental noise level. In case the environmental noise is high, the user may have problems to understand a pure verbal presentation. Therefore, the system would give a higher preference to graphics. Figure 4 shows the LUA-implementation of this behaviour.

```
function chooseModality()
if(contextPresent("NoiseLevel")) then
  noiseLevel = context["NoiseLevel"].value
  if(noiseLevel == "high") then
    preferModality("Graphics")
endif
end function
```

Fig. 4. Example of a LUA scripted story action in order to extend the basic functionality of the ACOSAS-system. chooseModality() verifies if the context, detecting the noise level, is present or not. In case this information is present, the system decides on a presentation modality depending on the noise level.

Furthermore, we have the option of defining new context-related actions that are triggered by sensor data (Callback functions). For example, an action `setVirtualUserPosition` as shown in Fig. 5 is triggered whenever the specified sensor detects a movement of the user and adjusts the position of the camera in the VR scene.

```
function setVirtualUserPosition(parameter)
    local userPosX = parameter[1]
    local userPosY = parameter[2]
    -- knowing that the tracking system provides the real position of the user
    -- with origin centred to the screen the story engine calculates the
    -- camera movement directly from the retrieved tracking values.
    Cam.Pos.x = Cam.Pos.x + (scale * userPosX)
    Cam.Pos.y = Cam.Pos.y + (scale * userPosY)
end
```

Fig. 5. Example of context triggered action. Using a context module specifying `setVirtualUserPosition` as their *context method* this code is executed each time new context data is available. Keeping the example simple no rotation is integrated.

6 Application Scenarios

In order to evaluate the ACOSAS approach, we have designed and implemented a couple of application scenarios relying on different kinds of sensor hardware. For the purpose of this paper, we only describe simplified versions of the applications to demonstrate how context information influences the flow of a story.

6.1 VR Tourist Guide

The VR Tourist Guide Ritchie provides the user with information on the city of Augsburg as a preparation for a real visit on the same day. As soon as a user has approached the screen, Ritchie engages in small talk before starting with the actual tour. The first stop is the Mayor's Hall. Ritchie adapts his presentation to the number of persons, the gender distribution as well as the current weather conditions. For example, if the weather is unpleasant, he keeps the presentation of the architectural features rather short and suggests to move inside the building instead. This application is realized using the following context modules:

– *person recognition:* A system which recognizes whether a person is standing in front of the screen using a camera positioned at the ceiling.
– *gender recognition:* This module finds out the gender distribution of the audience by employing techniques from [Vogt2006].
– *temperature module:* A system that acquires the present temperature value. In case a hardware temperature sensor is not available, the temperature is taken from an internet web service.

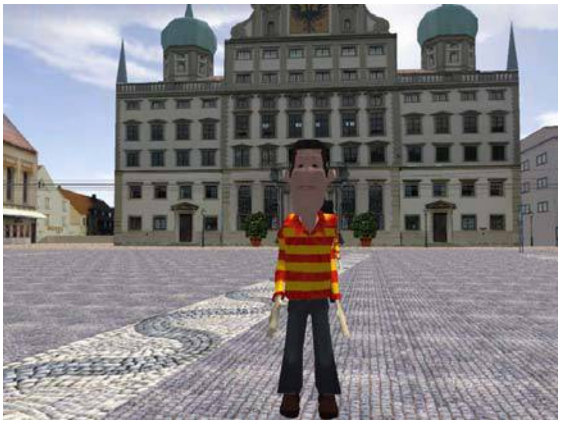


Fig. 6. A screen shot of the VR-Tourist guide application showing the guide Ritchie standing on the town hall square of Augsburg

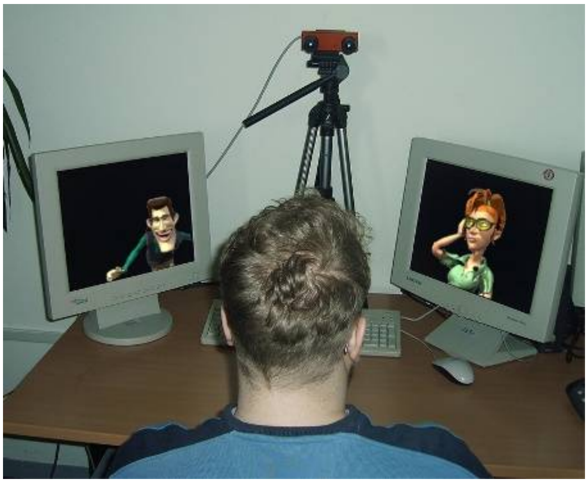


Fig. 7. The constellation of the second application showing two displays with different characters and the head tracker. Depending on which display the user is focusing the story continues differently.

6.2 Guiding Presentations by Tracking the User's Attention

The second application serves to demonstrate how the user's head movements may be used to influence the continuation of a story.

The setup of this application (cf. Fig. 7) consists of two displays and one head tracker to recognize the head position and rotation of a person sitting in front of the monitors [Morency2002]. In this application the head tracker is the only context provider.

The character Ritchie who appears on the left screen aims at informing the user about the newest developments in information technologies while Tina who is displayed on the right screen is rather interested in informing the user about the latest gossip. Depending on which screen the user is looking at, the corresponding character starts talking. If the user shifts her attention to the other screen, the presenter will have to give the turn to the other character.

6.3 Augsburg City Run

Augsburg City Run is an immersive game which enables the user to navigate through a crowd via movements of his head. While the user moves in front of a 3D projection screen, he is wearing shutter glasses enabling a 3D impression (cf. Fig. 8). Virtual characters autonomously move through the inner city of Augsburg. The user's task in this game is to catch a specific pedestrian, but to avoid hitting others in a crowd within a close time limit. The player controls the continuation of his journey only by moving his upper body.

An optical tracking system recognizes markers attached to the player's shutter-glasses and is thus able to capture the player's movements. Context data in this

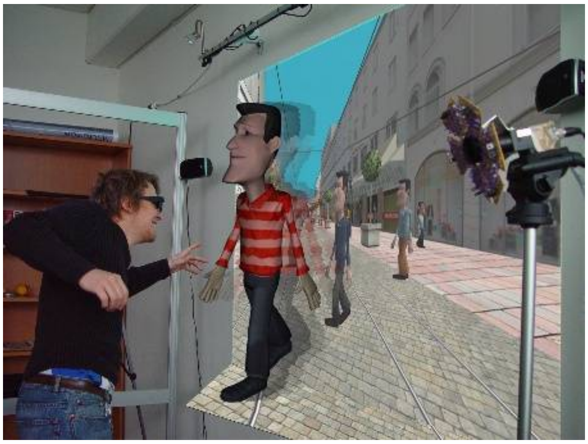


Fig. 8. The application Augsburg City Run. The user interacts with the VR-Environment only by head movement.

application include information on the user's head position. The system transmits positional changes in order to perform an adaption of the viewing perspective.

The impression of immersion is achieved by combining the optical tracking system with the navigation control and a special 3D projection system that adapts itself dynamically to the viewpoint of the user in front of the projection screen. This feeling is enhanced by 3D sound effects.

7 Conclusion

In this paper, we have presented a framework that allows an author to easily integrate context constraints that may relate both user states as well as environmental information into a story. In this way, contextual circumstances continuously influence a story at runtime. ACOSAS fosters the development of a new generation of story telling systems that allow for richer user interactions by taking into account the user's environmental context. Our approach bears many similarities with Gebhard's scene maker, but enhances it by mechanisms to easily access context information. Finally, we have presented a number of applications that have been developed using the ACOSAS framework.

References

[Bulterman2005] D. Bulterman, L. Hardman, *Structured Multimedia Authoring*, ACM Trans. Multimedia Comput. Commun. Appl. (2005) 89 - 109

[Charles2004] F. Charles, M. Cavazza, S. J. Mead, O. Martin, A. Nandi, X. Marichal, *Compelling experiences in mixed reality interactive storytelling*, Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology (2004) 32 - 40

[Dow2005] S. Dow, J. Lee, C. Oezbek, B. MacIntyre, J. D. Bolter, M. Gandy, *Wizard of Oz interfaces for mixed reality applications*, CHI '05: CHI '05 extended abstracts on Human factors in computing systems, ACM Press, New York, USA, 2005, 1339 - 1342

[Gebhard2003] P. Gebhard, M. Kipp, M. Klesen, T. Rist, *Authoring scenes for adaptive, interactive performances*, AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems (2003) 725 - 732

[Callaway2002] C. B. Callaway, J. C. Lester, *Narrative prose generation*, Artif. Intell., 139(2), Elsevier Science Publishers Ltd., 2002, 213 - 252

[MacIntyre2004] B. MacIntyre,M. Gandy,S. Dow, J. D. Bolter, *DART: a toolkit for rapid design exploration of augmented reality experiences* In Proceedings of the 17th Annual ACM Symposium on User interface Software and Technology (Santa Fe, NM, USA, October 24 - 27, 2004). UIST '04. ACM Press, New York, NY, 197 - 206

[Morency2002] L.-P. Morency, T. Darrell, *Stereo Tracking using ICP and Normal Flow*, Proceedings Int. Conf. on Pattern Recognition, 2002.

[Nischt2006] M. Nischt, H. Prendinger, E. André and M. Ishizuka, *MPML3D: a Reactive Framework for the Multimodal Presentation Markup Language*, Intelligent Virtual Agents: 6th International Working Conference, IVA 2006, 2006.

[Pinhanez1998] C. Pinhanez and A. Bobick, *Human Action Detection Using PNF Propagation of Temporal Constraints*, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition , Santa Barbara, CA, June 1998.

[Romero2004] L. Romero, J. Santiago, N. Correia, *Contextual information access and storytelling in mixed reality using hypermedia*, Comput. Entertain. (2004) 12 - 12

[Salber1999] D. Salber, A. K. Dey, G. D. Abowd, *The context toolkit: aiding the development of context-enabled applications*, CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems (1999) 434 - 441

[Vogt2006] T. Vogt, E. André, *Improving Automatic Emotion Recognition from Speech via Gender Differentiaion*, Proc. Language Resources and Evaluation Conference (LREC 2006) , 2006