A platform for Embodied Conversational Agents based on Distributed Logic Programming*

Anton Eliëns, Zhisheng Huang and Cees Visser
Intelligent Multimedia Group
Vrije Universiteit, Amsterdam, Netherlands
{eliens,huang,ctv}@cs.vu.nl

Abstract

In this paper we will outline the requirements for a software platform supporting embodied conversational agents. These requirements encompass computational concerns as well as presentation facilities, providing a suitably rich environment for applications deploying conversational agents.

We will then propose a platform based on the distributed logic programming language DLP and X3D, the extensible Web3D format. Three case studies will be described, illustrating the potential of the DLP+X3D platform: a multi-user game with autonomous players, avatars commenting on rich media presentations, and a gesture scripting language for humanoids based on dynamic logic.

In conclusion, we will discuss related work and review the evaluation criteria that pertain to the deployment of embodied conversational agents in rich media 3D environments.

1 Introduction

A variety of applications may benefit from deploying embodied conversational agents, either in the form of animated humanoid avatars or, more simply, as a 'talking head'. An interesting example is provided by *Signing Avatar*[1], a system that allows for translating arbitrary text in both spoken language and sign language for the deaf, presented by animated humanoid avatars. Here the use of animated avatars is essential to communicate with a particular group of users, using the sign language for the deaf.

Other applications of embodied conversational agents include e-commerce and social marketing, although in these cases it may not always be evident that animated avatars or faces actually do provide added value.

Another usage of embodied conversational agents may be observed in virtual environments such as Active Worlds[2], *blaxxun* Community[3] and Adobe Atmosphere[4]. Despite the rich literary background of such environments, including Neil Stephenson's Snow Crash, the functionality of such agents is usually rather shallow, due to the poor repertoire of gestures and movements on the one hand and the restricted computational model underlying these agents on the other hand. In effect, the definition of agent avatars in virtual environments generally relies on a proprietary scripting language which, as in the case of *blaxxun* Agents, offers only limited pattern matching and a fixed repertoire of built-in actions.

In contrast, the scripting language for *Signing Avatar* is based on the H-Anim[5] standard and allows for a precise definition of a complex repertoire of gestures, as exemplified by the sign language for the deaf. Nevertheless, also this scripting language is of a proprietary nature and does not allow for higher-order abstractions of semantically meaningful behavior.

In this paper, we propose an architecture and software platform for embodied conversational agents that provides the computational facilities for defining semantically meaningful behaviors and that allows for a rich presentational environment, in particular 3D virtual environments that may include streaming video, text and speech.

*http://www.cs.vu.nl/eliens/research/media/paper-platform.html
[1]http://www.signingavatar.com
[2]http://www.activeworlds.com
[3]http://www.blaxxun.com
[4]http://www.adobe.com/products/atmosphere
[5]http://www.hanim.org

structure The structure of this papers is as follows. In section 2, we will outline the requirements for a rich media software platform supporting embodied conversational agents. Then, in section 3, we will propose a platform based on the distributed logic programming language DLP and X3D. In section 4, three case studies will be described, illustrating the potential of the DLP+X3D platform. In section 5 we will review the evaluation criteria that pertain to the deployment of embodied conversational agents in rich media 3D environments, and in section 6 we will discuss related work. Finally, in section 7, we will give some conclusions and indications for future work.

2 Requirements

There is a wide range of presentation platforms for embodied conversational agents. On one end of the spectrum we have digitized video, as for example in the early versions of the Ananova[6] newsreader, On the other end of the spectrum we have rich media real-time 3D platforms, as in the aforementioned virtual environments.

For both type of systems, advanced authoring tools may be used to create content. In addition, however, for the latter type of systems also declarative means of modeling may be used as well as a programmatic interface to create dynamic content.

Based on our experiences, which are reported in section 4, we have a clear preference for web-based real-time 3D environments with a strong programmatic interface. In other words, this excludes systems that rely on offline rendering, but also systems that rely on the native graphics of a particular machine. Also, we rather program the dynamic behavior than create such behavior using advanced authoring tools. However, our approach allows for easily incorporating content produced by these tools. Our requirements with respect to a platform supporting such environments may be summarized as follows.

- *declarative language* – for agent support
- *multiple threads of control* – for multiple shared objects
- *distributed communication* – networking capabilities (TCP/IP)

As indicated in section 4, an environment meeting these requirements may be used to create agent-based multi-user virtual environments. We have a definite preference for a logic-based declarative language that provides direct support for intelligent agents. As we have shown in Huang et al. (2002), both autonomous agents and shared objects may be realized using agent-technology. The platform must, furthermore, provide sufficient networking capabilities, so that changes to shared objects in a world can be propagated among multiple users. As a remark, in our platform we have used the X3D/VRML External Authoring Interface (EAI) to control the 3D environments programmatically.

scripting behavior Now, although a platform as described above offers powerful computational capabilities, this is clearly not enough to create embodied conversational agents with a rich repertoire of gestures. On top of this platform, we need a suitably expressive scripting language for defining gestures and driving the behavior of our humanoid agent avatars.

In section 4.3, an outline is given of a gesture scripting language for humanoids based on the H-Anim 1.1 standard. The design of the scripting language was motivated by the requirements listed below.

- *convenience* – for non-professional authors
- *compositional semantics* – combining operations
- *re-definability* – for high-level specification of actions
- *parametrization* – for the adaptation of actions
- *interaction* – with a (virtual) environment

As described in section 4.3, we developed a scripting language STEP that, to our belief, meets these requirements (Huang et al. (2002b)). STEP is based on dynamic logic and allows for arbitrary abstractions using the primitives and composition operators provided by our logic (Harel (1984)).

[6] http://www.ananova.com

3 The DLP+X3D platform

In Huang et al. (2002), we have described a platform for virtual environments based on agent technology. In effect, our platform is the result of merging VRML with the distributed logic programming language DLP, using the VRML External Authoring Interface. This approach allows for a clear separation of concerns, modeling 3D content on the one hand and determining the dynamic behavior on the other hand. As a remark, recently we have adopted X3D as our 3D format. The VRML profile of X3D[7] is an XML encoding of VRML97.

The language DLP is a distributed object-oriented extension of Prolog (Eliëns (1992)). It supports multiple inheritance, non-logical instance variables and multi-threaded objects (to allow for distributed backtracking). Object methods are collections of clauses. Method invocation is dealt with as communication by rendez-vous, for which synchronization conditions may be specified in so-called *accept* statements. The current implementation of DLP is built on top of Java.

To effect an interaction between the 3D content and the behavioral component written in DLP, we need to deal with two issues:

- control points: *get/set* – position, rotation, viewpoint
- *event-handling* – asynchronous accept

We will explain each of these issues separately below. In addition, we will indicate how multi-user environments may be realized with our technology.

control points The control points are actually nodes in the VRML scenegraph that act as handles which may be used to manipulate the scenegraph. In effect, these handles are exactly the nodes that may act as the source or target of event-routing in the 3D scene. As an example, look at the code fragment below, which gives a DLP rule to determine whether a soccer player must shoot:

```
findHowToReact(Agent,Ball,Goal,shooting) :-
   get(Agent,position,sfvec3f(X,Y,Z)),
   get(Ball,position,sfvec3f(Xb,Yb,Zb)),
   get(Goal,position,sfvec3f(Xg,Yg,Zg)),
   distance(sfvec3f(X,Y,Z),sfvec3f(Xb,Yb,Zb),DistB),
   distance(sfvec3f(X,Y,Z),sfvec3f(Xg,Yg,Zg),DistG),
   DistB =< kickableDistance,
   DistG =< kickableGoalDistance.
```

This rule will only succeed when the actual distance of the player to the goal and to the ball satisfies particular conditions. In addition to observing the state of the 3D scene using the *get* predicate, changes to the scene may be effected using the *set* predicate.

event handling Our approach also allows for changes in the scene that are not a direct result of setting attributes from the logic component. Therefore we need some way to intercept events. In the example below, we have specified an observer object that has knowledge of, that is inherits from, an object that contains particular actions.

```
:- object observer : [actions].
var slide = anonymous, level = 0, projector = nil.

observer(X) :-
   projector := X,
   repeat,
     accept( id, level, update, touched),
   fail.

id(V) :-  slide := V.
level(V) :- level := V.
touched(V) :- projector←touched(V).
update(V) :- act(V,slide,level).
:- end_object observer.
```

[7] http://www.web3d.org/fs_x3d.htm

The constructor sets the non-logical variable *projector* and enters a repeat loop to accept any of the incoming events for respectively *id*, *level*, *update* and *touched*. Each event has a value, that is available as a parameter when the corresponding method is called on the acceptance of the event. To receive events, the *observer* object must be installed as the listener for these particular events.

The events come from the 3D scene. For example, the *touched* event results from mouse clicks on a particular object in the scene. On accepting an event, the corresponding method or clause is activated, resulting in either changing the value of a non-logical instance variable, invoking a method, or delegating the call to another object.

An observer of this kind is used in the system described in section 4.2, to start a comment (dialog) on the occurrence of a particular slide.

multi-user virtual environments The DLP platform offers direct support for autonomous agents, as for example the players in the soccer game described in section 4.1. However, our agent technology may also be used to realize multi-user virtual environments. In particular, we distinguish between the following types of agents:

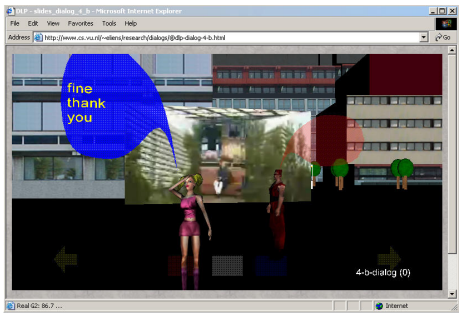
- *object agents* – control single shared objects (*pilot at server, drone at client*)
- *user agent* – controls users' avatar (*pilot at user side, drone at server or clients*)
- *autonomous agents* – like football player, with own avatar (*pilot at server, drone at clients*)

The terminology, *pilot* and *drone*, stems from the *Living Worlds* proposal, to indicate the difference between the original object (*pilot*) and the objects that replicate that object (*drones*). It extends the taxonomy of agents introduced in Huang et al. (2000) to allow for shared objects.

4 Case studies

To illustrate the potential of our DLP+X3D platform, we will briefly sketch three case studies, respectively a multi-user soccer game with autonomous player agents (fig. a), the use of dialogs in VR presentations (fig. b), and a scripting language for specifying gestures and movements for humanoids (fig. c).

(a) *soccer game*



(b) *dialog in context*

4.1 Multi-user soccer

The various aspects of our multi-user soccer game are reported in more detail in Huang et al. (2002). We chose the soccer game as a demonstrator or target application because it provided us with a number of challenges, as indicated below.

- *multiple (human) users* – may join during the game
- *multiple agents* – to participate in the game (e.g. goalkeeper)
- *reactivity* – players (users and agents) have to react quickly
- *cooperation/competition* – requires intelligent communication
- *dynamic behavior* – sufficiently complex (dynamic) 3D scenes

The soccer player agents each have a simple cognitive loop based on the extended BDI model described in Huang et al. (2001), which may be summarized as *sense*, *think*, *act*. [8]

Furthermore, to allow multiple users to join the game, taking the place of autonomous agent players if necessary, we designed a special purpose *Agent Communication Language* (ACL) to deal with the communication necessary to keep the world updated, as previously outlined in the discussion on multi-user virtual environments. The message format is, schematically: *Action, Type, Parameters.*

- *register game*: [register, game_name, from(Host)]
- *tell new player*: [tell, new_player, user(Host,Name)]
- *tell kick ball*: [tell, kick_ball, [user(Host,Name), force(X,Y,Z)]]
- *ask game score*: [ask, game_score, user(Host,Name)]

Other message types include:

> *register accept, register wait, tell position, tell rotation, tell game score, reply game score, unregister game, reply unregister game, player gone ...*

It is interesting to note that we may significantly optimize on the communication load if we introduce additional compound messages such as *run & trace ball*, to replace a sequence of primitive messages.

4.2 Dialogs in Virtual Environments

Desktop VR is an excellent medium for presenting information, for example in class, in particular when rich media or 3D content is involved. At VU, we have been using *presentational VR* for quite some time, and recently we included dialogs using balloons (and possibly avatars) to display the text commenting on a particular presentation, Eliëns et al. (2002). See figure (b) for an example displaying a virtual environment of the VU, a propaganda movie for attracting students, and two avatars commenting on the scene. The avatars and their text are programmed as annotations to a particular scene as described below.

Each presentation is organized as a sequence of slides, and dependent on the slides (or level within the slide) a dialog may be selected and displayed. See the *observer* fragment in section 3.

Our annotation for dialog text in slides looks as follows:

```
<phrase right="how~are~you">
<phrase left="fine~thank~you"/>
<phrase right="what do~you think~of studying ..."/>
...
<phrase left="So,~what~are you?"/>
<phrase right="an ~agent" style="[a(e)=1]"/>
<phrase left="I always~wanted to be~an agent" style="[a(e)=1]"/>
```

In figure (b), you see the left avatar (named *cutie*) step forward and deliver her phrase. This dialog continues until *cutie* remarks that she *always wanted to be an agent*. The dialog is a somewhat ironic comment on the contents of the movie displayed, which is meant to introduce the VU to potential students.[9]

Furthermore, there are a number of style parameters to be dealt with to decide for example whether the avatars or persona are visible, where to place the dialogs balloons on the display, as well as the color and transparancy of the balloons. To this end, we have included a *style* attribute in the *phrase* tag, to allow for setting any of the style parameters.

Apart from phrases, we also allow for gestures, taken from the built-in repertoire of the avatars. In section 4.3 we discuss how to extend the repertoire of gestures, using a gesture specification language.

Both phrases and gestures are compiled into DLP code and loaded when the annotated version of the presentation VR is started.

[8] The users' avatars do not have any cognitive model, but act directly on the users' keyboard and mouse input.

[9] Clearly, our approach is reminiscent to the notorious *Agneta & Frida* characters developed in the Persona project. See http://www.sics.se/humle/projects/persona/web

4.3 STEP – a scripting language for embodied agents

Given the use of humanoid avatars to comment on the contents of a presentation, we may wish to enrich the repertoire of gestures and movements to be able, for example, to include gestural comments or even instructions by gestures.

Recently, we have started working on a scripting language for humanoids based on dynamic logic. The STEP scripting language consists of basic actions, composite operators and interaction operators (to deal with the environment in which the movements and actions take place).

The basic actions of STEP consist of:

- *move* – `move(Agent,BodyPart,Direction,Duration)`
- *turn* – `turn(Agent,BodyPart,Direction,Duration)`

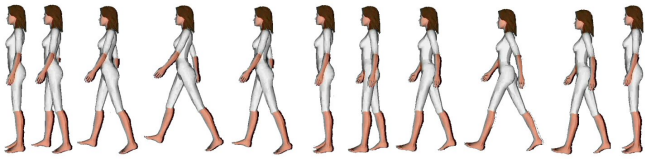
These basic actions are translated into operations on the control points as specified by the H-Anim 1.1 standard.

As composite operators we provide sequential and parallel composition, as well as *choice* and *repeat*. These composite operators take both basic actions and user-defined actions as parameters.

Each action is defined using the *script*, by specifying an action list containing the (possibly compound) actions of which that particular action consists. As an example, look at the definition of *walking* below.

```
script(walk(Agent), ActionList) :-
ActionList = [
  parallel([turn(Agent,r_shoulder,back_down2,fast),
         turn(Agent,r_hip,front_down2,fast),
         turn(Agent,l_shoulder,front_down2,fast),
         turn(Agent,l_hip,back_down2,fast)]),
  parallel([turn(Agent,l_shoulder,back_down2,fast),
         turn(Agent,l_hip,front_down2,fast),
         turn(Agent,r_shoulder,front_down2,fast),
         turn(Agent,r_hip,back_down2,fast)])
], !.
```

Notice that the *Agent* that is to perform the movement is given as a parameter. (Identifiers starting with a capital act as a logical parameter or variable in Prolog and DLP.)



(c) *walking humanoid*

Interaction operators are needed to conditionally perform actions or to effect changes within the environment by executing some command. Our interaction operators include: *test*, *execution*, *conditional* and *until*.

Potentially, an action may result in many parallel activities. To control the number of threads used for an action, we have created a scheduler that assigns activities to a thread from a thread pool consisting of a fixed number of threads.

As a demonstrator for STEP, we envisage to create an instructional VR for *Tai Chi*, the Chinese art of movement.

XML encoding Since we do not wish to force the average user to learn DLP to be able to define scripts in STEP, we are also developing X-STEP, an XML encoding for STEP. We use *seq* and *par* tags as found in SMIL[10], as well as *gesture* tags with appropriate attributes for speed, direction and body parts involved. As an example, look at the X-STEP specification of the *walk* action.

[10] http://www.w3.org/AudioVideo

```
<action type="walk(Agent)">
<seq>
 <par speed="fast">
  <gesture type="turn" actor="Agent" part="r_shoulder" dir="back_down2"/>
  ...
 </par>
 <par speed="fast">
  ...
  <gesture type="turn" actor="Agent" part="r_hip" dir="back_down2"/>
 </par>
</seq>
</action>
```

Similar as with the specification of dialog phrases, such a specification is translated into the corresponding DLP code, which is loaded with the scene it belongs to. For X-STEP we have developed an XSLT stylesheet, using the Saxon[11] package, that transforms an X-STEP specification into DLP. We plan to incorporate XML-processing capabilities in DLP, so that such specifications can be loaded dynamically.

5 Evaluation criteria

The primary criterium against which to evaluate applications that involve embodied conversational agents is whether the application becomes more effective by using such agents. Effective, in terms of communication with the user. Evidently, for the *Signing Avatar* application this seems to be quite obvious. For other applications, for example negotiation in e-commerce, this question might be more difficult to answer.

As concerns the embedding of conversationl agents in VR, we might make a distinction between *presentational VR*, *instructional VR* and *educational VR*. An example of educational VR is described in Johnson et al. (2002). No mention of agents was made in the latter reference though. In instructional VR, explaining for example the use of a machine, the appearance of a conversational agent seems to be quite natural. In presentational VR, however, the appearance of such agents might be considered as no more than a gimmick.

Considering the use of agents in applications in general, we must make a distinction between *information agents*, *presentation agents* and *conversational agents*. Although the boundaries between these categories are not clearcut, there seems to be an increasing degree of interactivity with the user.

From a system perspective, we might be interested in what range of agent categories the system covers. Does it provide support for managing information and possibly information retrieval? Another issue in this regard could be whether the system is built around open standards, such as XML and X3D, to allow for the incorporation of a variety of content.

Last but not least, from a user perspective, what seems to matter most is the naturalness of the (conversational) agents. This is determined by the graphical quality, as well as contextual parameters, that is how well the agent is embedded in its environment. More important even are emotive parameters, that is the mood and style (in gestures and possibly speech) with which the agents manifest themselves. In other words, the properties that determine whether an agent is (really) convincing.

6 Related work

There is an enormous amount of research dealing with virtual environments that are in one way or another inhabited by embodied agents. By way of comparison, we will discuss a limited number of related research projects.

As systems that have a comparable scope we may mention Broll (1996) and DIVE[12], that both have a client-server architecture for realizing virtual environments. Our DLP+X3D platform distinguishes itself from these by providing a uniform programmatic interface, uniform in the sense of being based on DLP throughout.

The Parlevink[13] group at the Dutch University of Twente has done active research in applications of

[11] http://saxon.sourceforge.com
[12] http://www.sics.se/dive
[13] http://parlevink.cs.utwente.nl

virtual environments with agents. Their focus is, however, more on language processing, whereas our focus may be characterized as providing innovative technology.

Both Tarau (1999) and Davison (2001) deal with incorporating logic programming within VRML-based scenes, the former using the External Authoring Interface, and the latter inline logic scripts. Whereas our platform is based on distributed objects, Jinni[14] deploys a distributed blackboard to effect multi-user synchronisation.

Our scripting language may be compared to the scripting facilities offered by Alice[15], which are built on top of Python. Also, *Signing Avatar* has a powerful scripting language. However, we wish to state that our scripting language is based on dynamic logic, and has powerful abstraction capabilities and support for parallelism.

Finally, we seem to share a number of interests with the VHML[16] community, which is developing a suite of markup languages for expressing humanoid behavior. We see this activity as complementary to ours, since our research proceeds from technical feasibility, that is how we can capture the semantics of humanoid gestures and movements within our dynamic logic, which is implemented on top of DLP.

7 Conclusions and future research

In summary, we may state that our DLP+X3D platform is a powerful, flexible and high-level platform for developing VR applications with embodied agents. It offers a clean separation of modeling and programming concerns. On the negative side, we should mention that this separation may also make development more complex and, of course, that there is a (small) performance penalty due to the overhead incurred by using the External Authoring Interface.

Where our system is currently lacking, clearly, is adequate computational models underlying humanoid behavior, including gestures, speech and emotive characteristics. The VHML effort seems to have a rich offering that we need to digest in order to improve our system in this respect.

Our choice to adopt open standards, such as XML-based X3D, seems to be benificial, in that it allows us to profit from the work that is being done in other communities, so that we can enrich our platform with the functionality needed to create convincing embodied agents in a meaningful context.

References

Broll W. (1996), VRML and the Web: A basis for Multi-user Virtual Environments on the Internet. In Proceedings of WebNet96, H. Maurer (ed.), AACE, Charlottesville, VA (1996), 51-56.

Davison A. (2001), Enhancing VRML97 Scripting, Euromedia'2001, Valencia, Spain, April 18-20. available from: http://fivedots.coe.psu.ac.th/∼ad

Eliëns A. (1992), DLP – A language for Distributed Logic Programming, Wiley

Eliëns A., Huang Z., Visser C. (2002), Presentational VR – *What is the secret of the slides?*, in preparation

H-Anim 1.1 , http://H-Anim.org/Specifications/H-Anim1.1

Harel D. (1984), Dynamic Logic. In: Handbook of Philosophical Logic, Vol. II, D. Reidel Publishing Company, 1984, pp. 497-604

Huang Z., Eliëns A., van Ballegooij A., De Bra P. (2000), A Taxonomy of Web Agents, IEEE Proceedings of the First International Workshop on Web Agent Systems and Applications (WASA '2000), 2000.

Huang Z., Eliëns A., Visser C. (2001), Programmability of Intelligent Agent Avatars, Proceedings of the Agent'01 Workshop on Embodied Agents, June 2001, Montreal, Canada

Huang Z., Eliëns A., Visser C. (2002), 3D Agent-based Virtual Communities. In: Proc. Int. Web3D Symposium, Wagner W. and Beitler M.(eds), ACM Press, pp. 137-144

Huang Z., Eliëns A., Visser C. (2002b), STEP – A scripting language for embodied agents. In preparation

Johnson A., Moher T., Cho Y-J., Lin Y.J., Haas D., and Kim J. (2002), Augmenting Elementary School Education with VR, IEEE Computer Graphics and Applications, March/April

Tarau P. (1999), Jinni: Intelligent Mobile Agent Programming at the Intersection of Java and Prolog, Proc. of PAAM'99, London, UK, April, see also http://www.binnetcorp.com/Jinni

[14] http://www.binnetcorp.com/Jinni
[15] http://www.alice.org
[16] http://www.vhml.org